Intuitive Exploration of Multivariate Data

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt
von

Daniel Paurat
aus
Duisburg

Bonn, 2017

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Thomas Gärtner
2. Gutachter: Prof. Dr. Stefan Wrobel
Tag der Promotion: 15.03.2017
Erscheinungsjahr: 2017

Daniel Paurat

University of Bonn
Department of Computer Science III

and

Fraunhofer Institute for Intelligent Analysis
and Information Systems IAIS

Declaration

I, Daniel Paurat, confirm that this work is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at the point of their use. A full list of the references employed has been included.

Acknowledgements

I would like to express my sincere gratitude to Thomas Gärtner for the trust and for encouraging me to start this work, for the constant support throughout it, countless discussions and for being a great supervisor in general. Henrik Grosskreutz and Mario Boley for guiding me through the first year. Roman Garnett, Tamás Horváth, Ulf Brefeld and Kristian Kersting for all the quality discussions. Michael Kamp for innumerable coffee and theory crafting sessions; I really enjoyed our talks a lot. The other PhD students at Fraunhofer institute from Thomas' and Kristian's research groups, namely Pascal Welke, Olana Missura, Dino Oglic, Katrin Ullrich, Anja Pilz, Fabian Hadiji, Babak Ahmadi and Mirwaes Wahabzada. Not to forget my awesome office mates Sandy Moons and Marion Neumann. All of you guys and some of the really inspiring students that I met over the last couple of years provided an exquisite environment for me to pursue my thesis. It was a great time for me and I certainly would ~~drink~~ work with all of you again!

I would also like to thank my parents Roland and Ute Paurat for the trust they granted me. My partner Stephanie Höppner for the freedom and the constant stream of motivation, as well as our son Jakob for providing me the final reason to complete this work. Finally, I would like to thank all of my friends, who repeatedly had to endure my talks on data analysis, with a special reference to Sebastian Ginzel for the (sometimes much needed) discussions and talks. I owe all of you guys my deepest gratitude.

Part of this work was supported by the German Science Foundation (DFG) under the reference numbers 'GA 1615/1-1' and 'GA 1615/1-2'.

Abstract

Approaching a dataset with an analysis question is usually not a trivial process. Apart from integrating, cleaning and pre-processing the data, typical issues are to generate and validate hypotheses, to understand which algorithms to apply, to estimate parameter settings and to interpret intermediate analysis results. To this end, it is often helpful to explore the data at first in order to find and understand its main characteristics, the driving influences, structures and relations among the data records, as well as revealing outliers. *Exploratory data analysis*, a term coined by John W. Tukey (Tukey, 1977), is a loose set of methods, mostly of graphical nature, to summarize and understand the main characteristics of the data at hand. This work extends the set of exploratory data analysis methods by proposing several new methods that support the analyst in his, or her task of understanding the data. Over the course of this thesis, two conceptually different approaches are investigated.

The first approach studies pattern mining algorithms, a family of methods that find and report hypotheses which describe interesting sub-populations of the dataset to the analyst, where the interestingness is measured by different quality functions. As the results of pattern mining methods are interpretable by a human expert, these algorithms are often utilized to study a dataset in an exploratory way. Note that many pattern mining algorithms address the problem of finding a small set of diverse high quality patterns. To this end, this work introduces two new algorithms, one for relevant and one for Δ-relevant subgroup discovery. In addition an algorithmic framework for sampling patterns according to different pattern quality measures is introduced. The second approach towards exploratory data analysis leaves the discovery of interesting sub-populations to the analyst and enables him, or her to study a two dimensional projection of the data and interact with it. A scatter plot visualization of the projected data lets the analyst observe the data collection as a whole and visually uncover interesting structures. Manipulating the locations of individual data records within the plot further enables the analyst to alter the projection angle and to actively steer the projection. This way relations among the data records can be set, or discovered and aspects of the data's underlying distribution can be explored in a visual manner. Finding the according projections is not trivial and throughout this thesis three novel approaches are proposed to do so.

The thesis concludes with a synthesis of both approaches. Classical pattern mining algorithms often aim at reducing the output of patterns to a small set of highly interesting and diverse patterns. However, by discarding most of the patterns, a trade-off has to be made between ruling out potentially insightful patterns and possibly drowning the analyst in results. Combining interactive visual exploration techniques with pattern discovery, on the other hand, excels on working with larger pattern collections, as the underlying pattern-distribution emerges more clearly. This way, the analyst does not only retain an overview on the underlying structure of the dataset, but can also survey the relations among the interesting aspects of the dataset.

Contents

List of Algorithm Acronyms

BSD	–	Bitset based subgroup discovery
cKPCA	–	Constrained kernel principal component analysis
ClosedSD	–	Closed subgroup discovery
Closed$^+$SD	–	Closed subgroup discovery on the positive labeled portion of the data
CN2-SD	–	Clark and Niblett's CN2 algorithm for subgroup discovery
DP-subgroup	–	Depth pruning subgroup discovery
FP-growth	–	Frequent pattern growth
ICA	–	Independent component analysis
IMR	–	Inductive minimum representative construction
Isomap	–	Isometric mapping
KPCA	–	Kernel principal component analysis
LCM	–	Linear time closed itemset miner
LLE	–	Locally linear embedding
LSP	–	Least squared error projection
MDS	–	Multidimensional scaling
MLE	–	Most likely embedding
PCA	–	Principal component analysis
PP	–	Projection Pursuit
RelevantSD	–	Relevant subgroup discovery
SVD	–	Singular value decomposition

List of Tables

List of Figures

1. Introduction

"Our information age more often feels like an era of information overload. Excess amounts of information are overwhelming; raw data becomes useful only when we apply methods of deriving insight from it."

(Scott Murray, Interactive Data Visualization for the Web, 2013)

1.1. Background and Motivation

Collecting data has become omnipresent. Online retailers collect and evaluate enormous amounts of data to give better product recommendations to their users and thus increase their sales. Companies log and assess information along their processes to optimize their workflows, biologists utilize machine learning techniques on microarrays to discover relations between genes and diseases, and governments collect and analyze communication traces to identify potential threads. Along with the presence of inexpensive and available processing power, storage capacity and communication bandwidth researchers and companies are collecting more and more data with the goal of extracting valuable insights from it. The list of use-cases for discovering knowledge from databases is long and all of them share the hope that the quality of the extracted insights increases with the amount of collected data. Together with the growing processing and storing power, the size of the data collections is steadily increasing; in the amount of assembled data records, as well as the number of attributes that are being monitored.

With this overwhelming amount of data, there is an increasing need for efficient methods that help an analyst to develop an understanding of the data (Chakrabarti et al., 2008). Since there is usually no technique which directly extracts all the relevant insights, the analyst needs to explore the data in order to identify important variables and relations, detect anomalies, coin and test hypotheses and make algorithm and parameter choices.

Exploring the data, can help the analyst to understand the data's underlying structure, ask the right questions and ultimately uncover the desired insights. In general, extracting knowledge from data is an iterative process that involves repetitive modelling and understanding of it (Shearer, 2000) with the ultimate goal to gain previously unknown and potentially useful information (Frawley et al., 1992). To this end exploratory data analysis techniques can help the user to understand the underlying structure and relations of the data. In this thesis two orthogonal approaches were investigated, which both focus on a presentation of the results that is interpretable by a human domain expert. This yields the benefit that an expert of the domain from which the data derives can be empowered to perform knowledge discovery tasks without possessing in-depth machine learning knowledge. The first of the here investigated approaches studies techniques that automatically deliver human-understandable descriptions of interesting partitionings of the dataset to the analyst. The contributions from this part of the thesis to the scientific community are mainly towards finding more concise local pattern descriptions in a more efficient way. The other approach to exploratory data analysis investigates techniques that let the analyst observe all data records of a dataset and their relations at a glance. This part of the thesis studies a novel area of interactive visual data analysis. The idea behind all here introduced approaches is to enable the analyst to browse and navigate a two dimensional Scatter plot projection of the whole dataset in a live-updating and interactive manner. Seeing related data records move in cohesion, while altering the perspective, enables the analyst to understand their connections, coin and test hypotheses and the grasp underlying structure of the data itself.

To guide the reader with a consistent example dataset, throughout this thesis a collection of cocktail recipes will be used. This collection is based on 1702 recipes which were retrieved from the website `http://webtender.com` and it comprises 334 different ingredients. The cocktail dataset is on the one hand complex enough to be interesting and contain non-trivial insights. On the other hand the domain is easily understandable and the reader directly has an intuition for the results. Depending on the task and the utilized algorithms, the cocktail recipe collection is pre-processed differently to form a suitable dataset.

1.2. Contributions

This thesis addresses the human in the loop of data analysis tasks and extends the research on human-understandable knowledge discovery and exploratory data analysis methods by investigating two complementary approaches and their synthesis. The first approach, discussed in Chapter 2, focuses on finding interesting and yet diverse local patterns efficiently. In an exploratory data analysis setting, the discovered patterns can be utilized to guide the analyst's attention towards interesting sub-populations of the data collection. Employing different pattern mining methods and interestingness measures helps to uncover various aspects within the data. With respect to the pattern mining community, the contributions of this dissertation are the following:

- Section 2.2 presents an efficient algorithm to solve the problem of listing relevant patterns, as introduced by Garriga et al. (2008). It is a variation of an algorithm, introduced by Boley and Grosskreutz (2009) to traverse the search space of all closed and positively labeled patterns, following a "shortest-description-length-first" search strategy. The here presented version of the algorithm is designed to exhibit a small memory footprint, by applying an iterative deepening search strategy and resulted in a publication at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (Grosskreutz and Paurat, 2011).

- A follow-up publication at the SIGKDD Conference on Knowledge Discovery and Data Mining (Grosskreutz et al., 2012), refined the above mentioned relevance to a stricter formulation. The so called Δ-relevance, as introduced in Section 2.3, allows to omit patterns that are redundant to already discovered patterns up to a certain threshold. Employing this pattern mining technique usually leads to a more diverse result set of discovered interesting patterns, which are introduced in Section 2.1.2.

- In contrast to finding a small result set of high-quality patterns, additionally the idea of sampling patterns with a probability proportional to different interestingness measures was investigated. This lead to a publication at the SIGKDD Conference on Knowledge Discovery and Data Mining (Boley et al., 2011). The investigated random pattern sampling procedure, as introduced in Section 2.4, can be adjusted to expose different sampling biases that are closely related to different interesting measures.

The second approach to exploratory data analysis that is studied in this dissertation investigates direct interaction with a scatter plot visualization of all data records from a data collection. The main idea behind all techniques, proposed in this work, is to interact with the scatter plot visualization by manually placing individual points of the plot to a desired location. This interaction serves as input to the underlying algorithm, which maps the original data to a two dimensional space that is visualized, to consider the the feedback and re-calculate the mapping accordingly. Interacting with a visualization of all data records and directly receiving response helps the analyst to find interesting sub-populations, craft and check hypotheses and ultimately develop an understanding of the relations among the data records and the underlying structure of the whole dataset. The main contributions to this area of research are the following:

- Section 3.2 introduces a way to interact with and control a (in this case two-dimensional) projection of a dataset, by "grasping and dragging" individual data points within a scatter plot visualization. This technique is not limited to a certain algorithm and can be utilized to express domain knowledge, or to uncover structural properties of the dataset. A first implementation of this interaction method employed the *least squared error projection* (LSP) algorithm which solely considers the control points' data and embedding locations and calculates a linear projection

with the least squared residual error. This lead to a publication at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (Paurat and Gärtner, 2013) and build the foundation of a tool for interactive visualization (InVis), which is also a part of the contributions of this dissertation.

- A drawback of the LSP method is that with no control points placed, the algorithm projects every data record to the origin of the embedding space. To overcome this problem and some other limitations of the LSP method, when dealing with sparse data, a probabilistic approach was investigated. The resulting embedding method, discussed in Section 3.3, considers a prior belief about the embedding and regards the placement of the control points as evidence. Section 3.3 also shows that for a certain parameter choice this *most likely embedding* (MLE) is equivalent to the LSP algorithm. Paurat et al. (2014) discuss the underlying technique in the context of interactive visualizations briefly, as an almost identical mathematical framework was proposed by Iwata et al. (2013).

- An alternative approach to overcome the limitations of LSP and to improve on the flexibility of the underlying embedding algorithm lead to two publications on knowledge based *constraint Kernel-PCA* (cKPCA). One publication at the NIPS Workshop on Spectral Learning (Paurat et al., 2013b), the other one at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (Oglic et al., 2014). cKPCA is an interactive version of a kernel-PCA, as introduced in Section 3.4, that can take several types of constraints into account. These constraints can e.g. be given in the form of desired locations within the embedding of individual points.

The last contribution of this dissertation shows a way of combining pattern discovery and interactive embeddings. Large amounts of patterns tend to overload the analyst with information. For this reason, many pattern mining techniques revolve around the task of finding a small and condensed set of highly interesting and diverse patterns. The combination of pattern discovery and interactive embeddings takes a different approach:

- In Chapter 4, a general procedure is introduced, which facilitates interactive embedding methods to empower the user to interactively explore and understand large amounts of discovered patterns. Exploring a pattern collection interactively, helps the user to keep an overview on general topics among the patterns and allows dive into regions of interest on demand. Following this approach resulted in a publication at the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (Paurat et al., 2014).

1.3. Previously Published Work

As just stated in Section 1.2 on the contributions of this dissertation, parts of it have already been published in conference and workshop proceedings of the international conference of the Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining (ACM SIGKDD), the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) and the international conference on Neural Information Processing Systems (NIPS). In detail that is:

1. Mario Boley, Claudio Lucchese, Daniel Paurat, and Thomas Gärtner. Direct local pattern sampling by efficient two–step random procedures. In *Proceedings of the 17th annual ACM SIGKDD Conferences on Knowledge Discovery and Data Mining*, 2011

2. Henrik Grosskreutz and Daniel Paurat. Fast and memory–efficient discovery of the top–k relevant subgroups in a reduced candidate space. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2011

3. Henrik Grosskreutz, Daniel Paurat, and Stefan Rüping. An enhanced relevance criterion for more concise supervised pattern discovery. In *Proceedings of the 18th annual ACM SIGKDD Conferences on Knowledge Discovery and Data Mining*, 2012

4. Daniel Paurat and Thomas Gärtner. Invis: A tool for interactive visual data analysis. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2013

5. Daniel Paurat, Dino Oglic, and Thomas Gärtner. Supervised PCA for interactive data analysis. In *Proceedings of the 2nd NIPS Workshop on Spectral Learning*, 2013

6. Dino Oglic, Daniel Paurat, and Thomas Gärtner. Interactive knowledge–based kernel pca. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2014

7. Daniel Paurat, Roman Garnett, and Thomas Gärtner. Interactive exploration of larger pattern collections: A case study on a cocktail dataset. In *Proceedings of the 2nd ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, 2014

1.4. Outline

This section connects the chapters and sections by providing an outline through the thesis. Chapter 2 tackles different local pattern mining techniques that can be useful to an analyst in an exploratory data mining setting. Starting with the preliminaries and introducing the formal notation the Sections 2.2, 2.3 and 2.4 deal with efficient listing and sampling of local patterns. All of these techniques automatically find interesting descriptions of partitionings of the dataset, guiding the analysts attention towards statistically outstanding sub-populations of the data distribution. Sections 2.2 and 2.3 investigate relevant and Δ-relevant pattern mining methods. These techniques aim at efficiently listing the top non-redundant, concise and interesting subgroup descriptions of a labeled transactional database. Section 2.4 introduces a fast way to sample local patterns according to different interestingness measures. The chapter then concludes with a summary and discussion of the techniques.

Having investigated techniques that automatically find and deliver interesting aspects of the dataset, Chapter 3 changes the focus and studies methods that enable the analyst to observe and interact with a projection of the whole dataset. Being able to observe all data records and their relations at once and to directly interact with them, lets the analyst study the dataset "from a bird's eyes perspective" and discover interesting aspects on his own. This way the analyst can decide for himself which partitionings are of interest. The chapter starts again by introducing the preliminaries to these techniques and then continues to introduce three different algorithms that project the data into a lower dimensional space and allow the analyst to directly alter the projection. For the purpose of interactive visual analysis this lower dimensional space is the 2d plane. Here, the analyst can actively browse the whole dataset in a visual way, see some of the relations among the data records and understand the underlying structure of the dataset. Navigating the projection is done by relocating individual data records within the visualization in a "drag and drop" like manner. Selecting and relocating such a "control point" triggers the underlying embedding algorithm to consider the analysts feedback and shift the projection angle accordingly. This work introduces three different interactive embedding techniques that utilize the placement of control points to alter the projection of the data. Section 3.2 introduces a straight forward approach to interact with a projection via control points. From the here presented methods this is the fastest and most scalable algorithm. However, the approach is limited in several ways. For instance, with no control points given, the embedding collapses to the origin. Another limitation comes when dealing with sparse data. In this case, poorly chosen, or too few, control points can lead to a degenerated embedding that doesn't reveal any interesting aspect of the data. To overcome some of the limitations, in Section 3.3 a probabilistic version of an interactive embedding algorithm presented. Without any control points placed, it is able to start with a prior belief about a "good" projection of the data. Note that a similar idea has been published independently by Iwata et al. (2013). Although it does not focus on the interaction with the embedding, the underlying mathematics are largely alike. Section 3.4 studies an alternative approach to overcome the limitations

of the initial approach to interact with the embedded data. It presents an interactive version of a kernel PCA. This way, the embedding is not limited to linear projections any more, the initialization problem is solved and sparse datasets do not degenerate, as the variance among the data records is naturally taken into account. However, these benefits come at the price of computational complexity. The chapter concludes again with a summary and discussion.

The ideas and methods, presented in the Chapters 2 and 3 represent two very different approaches to exploratory data analysis and on how to find, study and understand the driving aspects of a dataset. Chapter 4 presents a natural way of combining these approaches, by interactively and visually analysing large pattern collections. To do so, the mined patterns have to be represented as vectors.

The final Chapter 5 concludes with a general discussion on this thesis, open issues and further research areas that might possibly emerge from this work.

2. Local Pattern Discovery

In an exploratory data analysis setting, the analyst tries to find interesting aspects of the data. This can be done, for instance, by studying and understanding the underlying distribution from which the data derives. Pattern mining can be of help here, as it automatically finds human interpretable descriptions of interesting partitionings of the data. In an exploratory setting, these patterns can be used to guide the analyst's attention towards interesting aspects of the dataset. This chapter investigates how to efficiently find interesting and yet diverse local patterns. To this end, two fundamentally different pattern mining approaches are studied. The first one considers the space of all possible patterns that are defined on a dataset and reports a small set of highly interesting, yet non-redundant, ones. One way of avoiding redundancy in a pattern discovery scenario, is to focus on the so called relevant patterns. As a contribution, this work proposes a novel algorithm for listing the top relevant patterns. The algorithm is faster and possesses a smaller memory footprint than it's competitors. In addition, the notion of relevance is re-considered to allow for some slack, which in terms yield a less redundant and more condensed result set. Another pattern mining approach that avoids redundancy in a natural way, is to randomly sample them. This work introduces a sampling procedure that can be adjusted to draw random patterns with a probability proportional to different interestingness measures. This way, the analyst can explore patterns with a certain bias towards an interestingness measure, but is not strictly bound to the "top" ones. In an exploratory setting, this leaves room to discover patterns that are highly attractive to the analyst, but are not considered interesting in terms of the measure.

2.1. Preliminaries

The following section gives an introduction to the notation of pattern mining. It provides the definition of a pattern that is used throughout this work, introduces basic concepts and notions like, e.g., extention and support, the true- and false positives of a pattern for a labeled dataset, and presents several measures of interest for a given pattern.

2.1.1. Patterns

For the task of pattern mining, we assume a database $\mathcal{D}$ of m data records, $d_1, \ldots, d_m$, with each data record being described by a set of n binary attributes, or features, $(a_1(d_i), \ldots, a_n(d_i)) \in \{0,1\}^n$. A **pattern** p is a subset of the attribute set, i.e. $p \subseteq \{a_1, \ldots, a_n\}$, with each single a_i of the pattern being referred to as item. For a given database $\mathcal{D}$, a data record d satisfies a pattern p if $a(d) = 1$ for all $a \in p$, that is, patterns are interpreted conjunctively. The cardinality of a pattern $|p|$ denotes the number of contained items, meaning the number of attributes a for which $a(d) = 1$. Sometimes also the notation $a_{i_1} \& \ldots \& a_{i_k}$ is used instead of $\{a_{i_1}, \ldots, a_{i_k}\}$, omitting the items for which $a(d) = 0$. In addition, this work also refers to a pattern as a **subgroup description** or **subgroup**. This comes due to the fact that parts of the here presented research were done in the specific pattern mining area of subgroup discovery. Subgroup descriptions are "regular" patterns in terms of their representation. The difference is that in order to measure the interestingness of a subgroup, an additional feature, the label, is considered. Hence, subgroup discovery algorithms explicitly consider labeled datasets and perform the task of finding frequent patterns that exhibit an unusual label distribution in comparison to the overall label distribution.

The expression $\mathcal{D}[p]$, also referred to as the extention of p, describes the set of data records $d \in \mathcal{D}$ of a database $\mathcal{D}$ that satisfy the pattern p. The **support** of p, denoted by $\text{supp}(\mathcal{D}, p)$ or short $\text{supp}(p)$, is the cardinality of the set of data records that are described by $\mathcal{D}[p]$. Considering the cocktail dataset, a pattern could for instance be *Vodka & Orange juice*. It describes the extention of all cocktails which contain *Vodka* and *Orange juice* at the same time. The Table 2.1 below shows an excerpt of five cocktails represented as transactions from our exemplary cocktail database.

Id	Name	Itemset
1	Caipirinha	Cachaça & Lime & Sugar
2	Mojito	Light rum & Lime & Mint & Soda water & Sugar
3	Piña Colada	Coconut milk & Light rum & Pineapple
4	Screwdriver	Vodka & Orange juice
5	Tequila Sunrise	Grenadine & Orange juice & Tequila
⋮	⋮	⋮

Table 2.1.: An itemset representation of five well known cocktails. The listed ingredients indicate their presence in the cocktail.

Additionally, the data records can be associated with a label. For the sake of simplicity, here only binary labels are considered, though in practice not all pattern mining methods are restricted to that. Formally, the label is a special attribute $class(d)$ that has the range $\{\oplus, \ominus\}$.

For such a binary labeled dataset the **true positives** (TP) and the **false positives** (FP) of a pattern p can be defined, with respect to the database $\mathcal{D}$, as

$$TP(\mathcal{D}, p) := \{d \in \mathcal{D}[p] \mid class(d) = \oplus\} \quad \text{and}$$

$$FP(\mathcal{D}, p) := \{d \in \mathcal{D}[p] \mid class(d) = \ominus\}.$$

The cardinalities of the sets $|TP(\mathcal{D}, p)|$ and $|FP(\mathcal{D}, p)|$ are denoted by $\text{supp}^+(p)$ and respectively $\text{supp}^-(p)$.

To stay in the previously given example of the cocktail dataset, the pattern *Vodka* & *Orange juice* describes the extention of all cocktails in our database that contain both ingredients *Vodka* and *Orange juice* at the same time. In our dataset there are 95 cocktails which support this pattern. Considering a label that indicates whether a cocktail is creamy or not, only 5 out of the 95 cocktails are labeled as creamy. These five data records are the true positives of the pattern, the other 90 form the false positives.

2.1.2. Interestingness of a Pattern

The interestingness of a pattern p in the context of a database $\mathcal{D}$ is measured by a **quality function** $q(\mathcal{D}, p)$ that assigns a real-valued quality to p. As patterns can be of interest for different reasons, there are diverse measures to determine the interestingness of a pattern. One common interestingness measure is the frequency of a pattern. It is defined as the share of data records that are supported by the pattern over the amount of all data records

$$\text{freq}(\mathcal{D}, p) = \frac{\text{supp}(\mathcal{D}, p)}{|\mathcal{D}|}.$$

Another prominent measure that does also not consider labels is the so called **lift** of a pattern. It compares the observed support of a pattern to its expected support if all the attributes were statistically independent. For instance, when considering the cocktail dataset, 452 out of the 1702 cocktails contain *Vodka* and 249 *Orange juice*. That is, 26.5% of the cocktails contain *Vodka* and 14.6% *Orange juice*. If we assume that the occurrence of these ingredients is independent, we would expect to observe that 3.8% of all cocktails contain both ingredients at the same time. However, in the data we observe 5.6% (95 cocktails) containing both ingredients. As this is almost 1.5 times our expected value, the pattern *Vodka* & *Orange juice* is lift-wise considered as interesting. More formally, the lift of the pattern p on the dataset $\mathcal{D}$ is defined as

$$\text{lift}(\mathcal{D}, p) = \frac{\text{freq}(\mathcal{D}, p)}{\prod_{p_i \in p} \text{freq}(\mathcal{D}, p_i)}.$$

Other prominent measures of interest additionally consider labels that are assigned to the data records. To see how this can be of value, let us consider a label for the cocktail dataset that indicates whether a cocktail is creamy, or not. For our previous example pattern *Vodka & Orange juice*, only 5 out of 95 cocktails which supported the pattern were creamy. Compared to the overall distribution, where 368 out of 1702 cocktails are labeled as creamy, the pattern under-represents the creamy cocktails. The shift of how much the label distribution of a pattern's extention deviates from the overall label distribution can be utilized as an interestingness measure. However, note that considering solely the ratio between $|\oplus|$ and $|\ominus|$ can be very sensitive to the amount of supporting data records. If, e.g., another pattern p' would only be supported by a single data record that also happens to be labeled as creamy, the ratio would be 1 out of 1. Although the ratio indicates a highly interesting pattern, the extention that the pattern describes would be much too small to be interesting to the analyst. To account for this drawback, the ratio is usually weighted by a function of the frequency of the pattern. This yields a quality measure which promotes patterns that expose a larger extention and have an unusual label distribution. Some of the most common quality functions for binary labeled data that capture this are of the form

$$q(\mathcal{D}, p) = |\mathcal{D}[p]|^{\alpha} \cdot \left(\frac{|TP(\mathcal{D}, p)|}{|\mathcal{D}[p]|} - \frac{|TP(\mathcal{D}, \varnothing)|}{|\mathcal{D}|} \right), \tag{2.1}$$

where α is a constant such that $0 < \alpha \leq 1$, and $TP(\mathcal{D}, \varnothing)$ simply denotes the set of all positively labeled data records in the dataset. The family of quality functions characterized by this equation includes some of the most popular quality functions: for $a = 1$, it is order equivalent to the Piatetsky-Shapiro quality function (Klösgen, 1996) and the weighted relative accuracy (WRACC) (Lavrač et al., 2004), while for $a = 0.5$ it corresponds to the binomial test quality function (Klösgen, 1996).

2.1.3. Listing Patterns

This section gives a brief introduction to the algorithmic approach of enumerating interesting patterns in an efficient way. Many pattern mining techniques find the interesting patterns by traversing the space of all possible patterns over the given attributes in a systematic way. Hereby the patterns are generated one after another and the quality of the pattern is measured (and reported).

Traversing the Pattern Space

Many pattern mining algorithms consider the set of all possible patterns over a set of attributes, ordered in a general to specific manner, as search space and traverse it in order to find the interesting patterns. Figure 2.1 displays this search space for a dataset over the three attributes a_1, a_2 and a_3. Here the patterns are connected by an edge, if they are in a super-/subset relation and differ by only one item. The patterns in this lattice are arrange in the so called *general to specific order*. As each item of a pattern

constitutes a constraints, the empty set, a pattern with no items, is the most general one. It does not exhibit any constraint and thus is supported by all data records of the data set. The opposite constitutes the set of all items. Usually there are no, or only very few, data records that satisfy all possible constraints and contribute to the support set of the most specific pattern. Note that for any amount of attributes the general to specific ordered search space always starts and ends in a single node, the empty set and the set of all attributes.

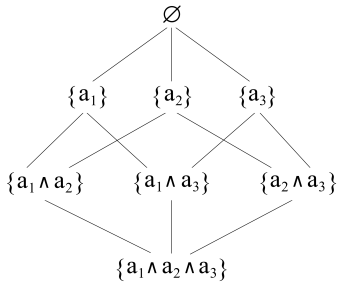


Figure 2.1.: All patterns that can be formed from the attributes a_1, a_2 and a_3. The connections denote the super-/sub-set relation between the patterns. Starting from the empty pattern, with each step down in the lattice, the patterns become more specific.

Traversing this pattern-lattice in a systematic way is the basis for a whole family of pattern mining algorithms. Most of them can be seen as an instance, or a refinement of the MIDOS algorithm (Wrobel, 1997). The difference among pattern mining algorithms can often be found in the traversal strategy. Depending on the objective of the algorithm, they usually start at one of the single-node endpoints and perform a breadth-, or a depth-first-search on the graph. In addition, there are several methods that let the algorithms terminate faster, or have a smaller memory footprint. One of these techniques, for instance, is that the algorithms avoid multiple visits of the same node. More complicated refinements involve an iterative deepening traversal strategy, or the application of a heuristic in order to perform a greedy beam-search. Another common technique which will be introduced in the following section is the so called pruning. In a scenario where only a fix amount of the highest quality patterns are desired, pruning can help to drastically reduce the runtime of an algorithm by excluding vast parts of the search space from the search.

Frequency and Optimistic Estimator Pruning

Consider again our cocktail dataset. In this dataset, there are 334 different ingredients and each of them may occur in a pattern, or not. This means that there are 2^{334} different patterns possible, a number with hundred and one digits. Usually it is not feasible to test all of these patterns exhaustively for their interestingness, however, there are ways to deal with this massive amount of patterns. Many interestingness measures are anti-monotone towards specializations of a pattern. This means that

specializing a pattern can only lower, of retain the interestingness. As a consequence for these interestingness measures, any specialization of a non-interesting pattern is also not interesting. State-of-the-art top-k pattern mining algorithm do not traverse the whole space of candidate patterns explicitly, but apply pruning to reduce the number of patterns effectively visited (Atzmüller and Lemmerich, 2009; Grosskreutz et al., 2008; Nijssen et al., 2009). The use of such techniques results in a dramatic reduction of the execution time and is an indispensable tool for fast exhaustive subgroup discovery and pattern mining algorithms in general (Atzmüller and Lemmerich, 2009; Grosskreutz et al., 2008; Morishita and Sese, 2000).

Consider a scenario, where the analyst is only interested in the top-k most frequent patterns of a dataset. In this case, not all patterns have to be tested for their frequency. The support, or frequency, of a pattern is a quality measure that is in an anti-monotone relation to the description length of the pattern. This can easily be understood, if each item of a pattern is interpreted as a constraint on the data records that support the pattern. The more constraints a pattern exhibits, the less data records are able to fulfill all of them. This means that specializing a pattern, by augmenting it with a new item, can only retain or lower the original support, but never increase it ($\forall\ p' \sqsupseteq p \Rightarrow \text{supp}(\mathcal{D}, p') \leq \text{supp}(\mathcal{D}, p)$). Hence, if a pattern does not exhibit a certain minimum support, none of its specializations will. When searching for the k most frequent patterns, the quality of the k^{th} best pattern found so far can be utilized as a minimum frequency threshold. While the algorithm finds better patterns, this threshold increases dynamically and the part of the search space with patterns that can potentially be among the k best ones shrinks. By dynamically increasing this threshold, the pattern space left to explore for the algorithm can be pruned Combined with a quality-bound on all specializations of patterns, a dynamically increasing threshold allows to ignore large parts of the search space, as it is guaranteed that all specializations of already ruled out patterns do not possess the desired minimum support.

A closely related concept is that of an **optimistic estimate** (Grosskreutz et al., 2008). An optimistic estimator is a function that provides a bound on the quality of a pattern and of all its specializations. Formally, an optimistic estimator for a quality measure q is a function oe that maps a database $\mathcal{D}$ and a pattern p to a real value such that for all $\mathcal{D}$, p and specializations $p' \sqsupseteq p$, it holds that $oe(\mathcal{D}, p) \geq q(\mathcal{D}, p')$. Note that pattern mining algorithms, which traverse the lattice of all patterns, scale exponentially with the amount of attributes. Utilizing an optimistic estimate pruning technique remedies this effect in practice to a certain extend, as it improves the expected runtime performance drastically (Grosskreutz et al., 2008; Morishita and Sese, 2000).

Reporting only the k best patterns also has the benefit that the resulting set of interesting patterns has a convenient size for a human analyst. However, the k reported patterns should not all revolve around the same aspect of the dataset. A good result set should

not only contain few highly interesting patterns, but also diverse ones. A common way to promote diversity in the result set is to avoid redundancy among the patterns via a closure system.

2.1.4. Avoiding Redundancy via a Closure System

A common mathematical framework that pattern mining methods employ to avoid redundancy among the reported patterns is that of a **closure operator**. A closure operator Γ on a set $\mathcal{S}$ is a function $\Gamma : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ from the power set of $\mathcal{S}$ into the power set of $\mathcal{S}$ that has to satisfy the following three properties for any two sets $s, s' \in \mathcal{P}(\mathcal{S})$:

1. $s \subseteq \Gamma(s)$ (extensivity)
2. $s \subseteq s' \Rightarrow \Gamma(s) \subseteq \Gamma(s')$ (monotonicity)
3. $\Gamma(s) = \Gamma(\Gamma(s))$ (idempotence)

The **closed sets** are the fixpoints of a closure operator on a dataset (Pasquier et al., 1999). *Closed pattern mining* algorithms find the (usually top-k) closed patterns of a dataset. The above definition allows for different realizations of the closure operator. One that has a broad application in pattern mining considers patterns of maximal description length in their support equivalence class as closed. Note, that a closure may have different generator sets, but that there is only one unique set of maximal description length.[1] This instantiation of the closure operator will be used throughout this work. This means that a pattern p is closed if and only if there is no $p' \supset p$ that is supported by the same data records. Using this concept of the closure operator, classic closed pattern mining algorithms report only the (unique) pattern of maximal description length for a support equivalence class. An example of this closure operator is illustrated on the following dataset:

Id	a_1	a_2	a_3	a_4
1	1	1	1	1
2	1	1	1	0
3	1	1	1	0
4	1	1	0	0
5	1	1	0	0

Table 2.2.: A dataset to illustrate closed sets.

[1] If two patterns p_1 and p_2 possess exactly the same extention, then $p_1 \cup p_2$ also possesses this extention. (For illustration, have a look at the patterns $\{a_1\}$ and $\{a_2\}$ in Table 2.2.) The union of all patterns of a support equivalence class $P_\cup$ is of maximal length. There is also only one pattern of maximal description length, because if there was another different pattern P' of the same support equivalence class with the same (maximal) length, P' would have been part of $P_\cup$ in the first place. As $P_\cup$ is the union of all patterns of that support equivalence class and $P' \neq P_\cup$, $P' \subset P_\cup$ and hence $|P_\cup| > |P'|$.

Consider the patterns $\{a_1\}$, $\{a_2\}$ and $\{a_1 \wedge a_2\}$. All of them are supported by exactly the same data records with the Ids $\{1, 2, 3, 4, 5\}$, hence, they are in the same *support equivalence class*. For these three patterns the one with the longest description length, the pattern $\{a_1 \wedge a_2\}$, is chosen as their representative. The tile that it spans in the dataset is highlighted in Table 2.2 with a dark blue outline. The other closed patterns of this example dataset are the pattern $\{a_1 \wedge a_2 \wedge a_3\}$ and the pattern $\{a_1 \wedge a_2 \wedge a_3 \wedge a_4\}$. Any of the 16 possible patterns over the attributes a_1 to a_4 belongs to one of the support equivalence classes, described by those four patterns.

Why is avoiding redundancy among the reported patterns interesting? As stated earlier, there is a huge space of candidate patterns that are potentially interesting to the analyst. Even after eliminating all obviously not interesting patterns, the remaining collection is usually still vast. As a human analyst is only capable of reviewing a small amount of output patterns, most pattern mining algorithms try to deliver a compact set of high quality patterns. For this small collection of patterns it is of great value, if the contained patterns represent different concepts of the underlying distribution. A good way to promote diversity is to discard patterns with redundant information.

There are many fast implementations of algorithms that find the frequent closed patterns of a dataset. Of particular interest to us are IMR and LCM (Boley and Grosskreutz, 2009; Uno et al., 2004), as they essentially perform depth first and breadth first search on the space of the closed patterns. It is also notable that implementation of LCM by Uno et al. (2004) was the winner of the FIMI contest (Bayardo et al., 2004) and is known among the pattern mining community for it's fast execution.

2.2. Relevant Patterns

The theory of relevance (Lavrač and Gamberger, 2005; Lavrač et al., 1999) aims at eliminating irrelevant patterns, respectively subgroups. Similar to the closed patterns, this remedies some of the redundancy among the finally reported patterns.

2.2.1. A Definition of Relevance

In order to be able to apply the theory of relevance to pattern mining approaches, the patterns have to be closed and the data records have to possess a binary label. Given this, a closed pattern p_{irr} is considered **irrelevant** if it is **dominated** (or **covered**) by another closed pattern p. More formal, a closed pattern p_{irr} is considered irrelevant if and only if a different closed pattern p exists in database $\mathcal{D}$ with

$$\text{i)} \quad TP(\mathcal{D}, p_{irr}) \subseteq TP(\mathcal{D}, p) \text{ and} \tag{2.2}$$

$$\text{ii)} \quad FP(\mathcal{D}, p) \subseteq FP(\mathcal{D}, p_{irr}).$$

All patterns that are not dominated are considered **relevant**. Applying the theory of relevance to pattern mining yields a set of finally reported patterns that are all closed and relevant. This means that for each of the not reported patterns there is a dominating relevant pattern in the result which is potentially of more value. The following Table 2.3 shows an example of the domination relation for the pattern p_{irr} on a toy dataset of only four labeled data records.

Row-Id	Label	p_{irr}	p_a	p_b	p_c	p_d	p_e	p_f	p_g	p_h	p_i
1	$\oplus$		×		×					×	×
2	$\oplus$	×	×	×	×	×			×	×	
3	$\ominus$	×			×	×	×	×	×	×	
4	$\ominus$							×	×	×	
			patterns that dominate p_{irr}				patterns that do not dominate p_{irr}				

Table 2.3.: A dataset of four labeled data records that exemplifies the dominance relation. The ×-symbol indicates that the pattern supports the data record with the according Row-Id.

Here, the ×-symbol indicates that a data record, identified by its Row-Id, supports a pattern. In the sense of the above defined dominance relation the pattern p_{irr} is dominated by the patterns p_a, p_b, p_c and p_d. All of these patterns cover all positively labeled instances of p_{irr} (and possibly more), while covering at maximum all negatively labeled instances of p_{irr}. Note that it is possible for two patterns to dominate each other (see p_{irr} and p_d), however only in the case that they have an identical extention. In this case, they share a common unique representative, the earlier introduced closure (See Section 2.1.4). On the other hand, the patterns p_e, p_f, p_g, p_h and p_i do not dominate the pattern p_{irr}. Some of these patterns do not cover all of the positively labeled instances that support p_{irr}, others cover a superset of the negatively labeled data records. Note that all patterns $p_a \dots p_h$ have an overlap in their support set with the one of p_{irr}. The patterns p_i and p_{irr}, however, have no overlap in their support sets. Here p_i can be seen as a representative for all patterns that possess this property. Meaning, that p_i and p_{irr} cannot dominate each other, as they are set-wise not comparable.

2.2.2. A Reformulation of Relevance

As shown by Garriga et al. (2008), the notation of relevance, as stated in equation 2.2, can be restated by using the closure operator over a set of attributes A, only on the positively labeled data records

$$\Gamma^+(p) := \{a \in A \mid \forall d \in TP(\mathcal{D}, p) : a(d) = 1\}.$$

Γ^+ is a closure operator, as introduced in Section 2.1.4, meaning that it is a function defined on the power-set of attributes $\mathcal{P}(\{a_1, \ldots, a_n\})$ such that for all patterns $p, p' \in \mathcal{P}(\{a_1, \ldots, a_n\})$, (i) $p \subseteq \Gamma(p)$ (extensivity), (ii) $p \subseteq p' \Rightarrow \Gamma(p) \subseteq \Gamma(p')$ (monotonicity), and (iii) $\Gamma(p) = \Gamma(\Gamma(p))$ (idempotence) holds. The fixpoints of Γ^+, i.e. the patterns for which $p = \Gamma^+(p)$, will further on be referred to as the **closed-on-the-positives**. The main result of Garriga's research for mining the relevant patterns in an efficient way is the following:

Proposition 1 *The space of relevant patterns consists of all patterns p_{rel} satisfying the following:*

(i) p_{rel} is closed-on-the-positives, and

(ii) there is no generalization $p \subsetneq p_{rel}$ that is closed-on-the-positives such that $|FP(\mathcal{D}, p)| = |FP(\mathcal{D}, p_{rel})|$.

This connection between relevancy and closure operators is particularly interesting because closure operators have extensively been studied in the area of closed pattern mining (Boley and Grosskreutz, 2009; Klösgen, 1996; Pasquier et al., 1999; Uno et al., 2004). However, unlike here, traditional closed pattern mining algorithms do not account for the label of the data.

2.2.3. Listing all Relevant Patterns

The publication of Garriga et al. (2008) is the first that proposes an approach to solve the relevant pattern discovery task. Making use of Proposition 1, Garriga et al. (2008) have proposed a simple two-step approach to find the relevant patterns:

1. Find and store all closed-on-the-positives
2. Remove all dominated closed-on-the-positives using Proposition 1

In the following, this subgroup discovery approach will be referred to as Closed$^+$SD. The search space considered by this algorithm — the closed-on-the-positives — is a subset of all closed patterns, thus it operates on a potentially exponentially smaller candidate space than all earlier approaches. The downside is that it does not account for optimistic estimate pruning, and that it has very high memory requirements, as the whole set of closed-on-the-positives has to be stored.

2.2.4. Efficient Listing of the Top-k Relevant Patterns

In the following, an algorithm is derived that possesses a memory-efficient way to test the relevance of a newly visited pattern while traversing the pattern space. For many datasets it is infeasible to store all closed-on-the-positive patterns in memory and then apply Proposition 1 to test for relevance. Instead, the following observation leads to a solution:

Proposition 2 *Let $\mathcal{D}$ be a dataset, $q(\cdot)$ be a quality function of the form of Equation 2.1 and θ some real value. Then, the relevance of any closed-on-the-positive p with $q(p) \geq \theta$ can be computed from the set of all generalizations of p with $q(p) \geq \theta$:*

$$G^* = \{p_{gen} \subsetneq p \,|\, p_{gen} \text{ is relevant in } \mathcal{D} \text{ and } q(\mathcal{D}, p_{gen}) \geq \theta\}$$

In particular, p is irrelevant if and only if there is a relevant pattern p_{gen} in G^* with the same negative support.[2]

To prove the correctness of Proposition 2, we first present two lemmas:

Lemma 3 *If a closed-on-the-positive p_{irr} is irrelevant, i.e. if there is a generalization $p \subsetneq p_{irr}$ closed on the positives with the same negative support as p_{irr}, then there is also at least one relevant generalization $p_{rel} \subsetneq p_{irr}$ with the same negative support.*

Proof Let N be the set of all closed-on-the-positives generalizations of p_{irr} with the same negative support as p. There must be at least one p_{rel} in N such that none of the patterns in N is a generalization of p_{rel}. From Proposition 1, we can conclude that p_{rel} must be relevant and dominates p_{irr}. □

Lemma 4 *If a relevant pattern p_{rel} dominates another pattern p_{irr}, then p_{rel} has higher quality than p_{irr}.*

Proof As a pattern can only be dominated by its generalizations and because support is antimonotonic, we have that $|\mathcal{D}[p_{rel}]| \geq |\mathcal{D}[p_{irr}]|$. Thus, to show that p_{rel} has higher quality, it is sufficient to show $|TP(\mathcal{D}, p_{rel})|/|\mathcal{D}[p_{rel}]| > |TP(\mathcal{D}, p_{irr})|/|\mathcal{D}[p_{irr}]|$. Because p_{rel} has to be a generalization of p_{irr} and because of the anti-monotonicity property, we can conclude that p_{rel} and p_{irr} have the same number of false positives; let F denote this number. Using F, we can restate the above inequality as $|TP(\mathcal{D}, p_{rel})|/(|TP(\mathcal{D}, p_{rel})| + F) > |TP(\mathcal{D}, p_{irr})|/(|TP(\mathcal{D}, p_{irr})| + F)$. All that remains to show is thus that $|TP(\mathcal{D}, p_{rel})| > |TP(\mathcal{D}, p_{irr})|$. By definition of relevance, $|TP(\mathcal{D}, p_{rel})| \geq |TP(\mathcal{D}, p_{irr})|$, and because p_{rel} and p_{irr} are different and closed on the positives, the inequality must be strict, which completes the proof. □

Based upon these lemmas, it is straightforward to prove Proposition 2:

Proof We first show that if p is irrelevant, then there is a generalization in G^* with the same negative support. From Lemma 3 we know that if p is irrelevant, then there is at least one *relevant* generalization of p with same negative support dominating p. Let p_{gen} be such a generalization. Lemma 4 implies that $q(\mathcal{D}, p_{gen}) \geq q(\mathcal{D}, p) \geq \theta$, hence p_{gen} is a member of the set G^*.
It remains to show that if p is relevant, then there is no generalization in G^* with same negative support. This follows directly from Proposition 1. □

[2] Only the support is needed, as p_{gen} is a generalization of p, which includes the extention of p.

Proposition 2 tells us that we can perform the relevance check based only on the top-k relevant patterns visited so far: Applying an iterative deepening traversal strategy of the space of all patterns in a general to specific order ensures that a pattern p is only visited, once all of it's generalizations have been visited first; so if the quality of the newly visited pattern p exceeds that of the k^{th}-best pattern visited so far, then the set of the best k relevant patterns visited includes all generalizations of p with higher quality; hence, we can check the relevance of p. On the other hand, if the quality of p is lower than that of the k^{th}-best pattern visited, then we don't care about its relevance anyways.

This leads us to the relevant subgroup discovery Algorithm 1, further on referred to as RelevantSD. The main program is responsible for the iterative deepening. The actual work is done in the procedure `findSubgroupsWithDepthLimit`, which traverses the space of the closed-on-the-positives in a depth-first fashion using a *stack* data structure. Thereby, it ignores found (closed-on-the-positive) patterns that are longer than the length limit, and avoids multiple visits of the same node using a standard technique like the *prefix-preserving property test* (Uno et al., 2004). Moreover, the function applies standard optimistic estimate pruning and dynamic quality threshold adjustment. The relevance check is done in line 6, relying on Proposition 2.

Let us now turn to the complexity of the algorithm and analyse it. To do so, let n denote the number of attributes in the dataset and m the number of records. Given that the maximum recursion depth is n, the maximum size of the result queue is k, and every pattern has length $\mathcal{O}(n)$, the algorithm has to store at maximum n patterns of length n, plus the k results (also of length n). This results in a memory complexity for the RelevantSD algorithm in the order of $\mathcal{O}(n^2 + kn)$.

For the runtime complexity, the following observations have to be considered and put together: For every node visited, the algorithm computes the quality, tests for relevance and considers at most n augmentations. The quality computation can be done in $\mathcal{O}(nm)$, while the relevance check can be done in $\mathcal{O}(kn)$. The computation of the successors in Line 5 involves the execution of n closure computations, each in $\mathcal{O}(nm)$, which amounts to $\mathcal{O}(n^2m)$. Altogether, the cost-per-node is thus $\mathcal{O}(n^2m + kn)$. Finally, the number of nodes considered is obviously bounded by $\mathcal{O}(|\mathcal{C}_p|\, n)$, where $\mathcal{C}_p$ is the set of closed-on-the-positives and the factor n is caused by the iterative deepening approach. So all put together, the RelevantSD algorithm has a time complexity of $\mathcal{O}(|\mathcal{C}_p| \cdot [n^3m + n^2k])$.

Table 2.4 compares the runtime and space complexity of the RelevantSD algorithm with that of Closed$^+$SD, classical depth first search subgroup discovery with pruning (DP-subgroup) and closed subgroup discovery algorithms (ClosedSD). Although these algorithms solve a different and simpler task, it is interesting to observe they do not have lower complexities. The expression $\mathcal{S}$ used in the table denotes set of all subgroup descriptions, while $\mathcal{C}$ denote the set of closed subgroups.

Algorithm 1 Iterative Deepening Top-k RelevantSD

Input : An integer k, a database $\mathcal{D}$ over attributes $\{f_1; ...; f_n\}$,
a quality function $q()$ with an optimistic estimator $oe()$
Output : The top-k relevant subgroups

main:

1. **let** *result* = queue of maximum capacity k (initially empty)
2. **let** $\theta = 0$
3. **for** *limit* = 1 to n **do**
4. `findSubgroupsWithDepthLimit`(*result*, *limit*, θ)
5. **end for**
6. **return** *result*

procedure `findSubgroupsWithDepthLimit`(*result*, *limit*, θ):

1. **let** stack = new stack initialized with $\varnothing$ as initial pattern
2. **while** *stack* not empty **do**
3. **let** *next* = pop from *stack*
4. **if** $|next| \leq limit$ and $oe(next) > \theta$ **then**
5. add all direct specializations of *next* to *stack* (avoiding multiple visits)
6. **if** q($next$) > θ and is not dominated by any $p' \in result$ **then**
7. add *next* to *result*
8. update θ to $min(\ \{q(p) \mid \forall p \in result\}\ )$
9. **end if**
10. **end if**
11. **end while**

Let us consider some of the properties of these algorithms in more detail, starting with the memory complexity: Except for Closed$^+$SD, all approaches can employ depth-first-search and thus have moderate memory requirements. Closed$^+$SD, on the other hand, has to collect all closed-on-the-positives, each of which has a description of length n. Please note that no pruning is applied, meaning that $n|\mathcal{C}_p|$ is not a loose upper bound for number of nodes stored in memory, but the exact number — which is why the Θ-notation in Table 2.4 is used. As the number of closed-on-the-positives can be exponential in n, this approach can quickly become infeasible.

As for the runtime, let us compare the complexity of RelevantSD with that of classic, and respectively closed subgroup discovery algorithms. Probably the most important difference is that the algorithms operate on different spaces. While the time complexity of RelevantSD is higher by a linear factor (resp. quadratic, compared to classic subgroup discovery), the search space, i.e. the closed-on-the-positives $\mathcal{C}_p$, can be exponentially smaller than the one considered by the other approaches (i.e. $\mathcal{C}$, respectively its superset $\mathcal{S}$). The exponential difference in the size of the search space can easily be seen for

Algorithm	Memory	Runtime	Pruning
DP-subgroup	$\mathcal{O}(n^2 + kn)$	$\mathcal{O}(\lvert\mathcal{S}\rvert \cdot nm)$	yes
ClosedSD	$\mathcal{O}(n^2 + kn)$	$\mathcal{O}(\lvert\mathcal{C}\rvert \cdot n^2 m)$	yes
Closed$^+$SD	$\Theta(n\,\lvert\mathcal{C}_p\rvert)$	$\Theta(\lvert\mathcal{C}_p\rvert \cdot n^2 m)$	no
RelevantSD	$\mathcal{O}(n^2 + kn)$	$\mathcal{O}(\lvert\mathcal{C}_p\rvert \cdot [n^3 m + n^2 k])$	yes

Table 2.4.: Runtime and memory complexity of different pattern discovery approaches, compared to the here proposed RelevantSD algorithm.

datasets that are constructed as follows: We define a binary dataset $\mathcal{D}_n = d_1, \ldots, d_n, d_{n+1}$ with $n+1$ data records over n attributes $a_1, \ldots, a_n$ and a label. The first n data records are constructed as

$$a_j(d_i) = \begin{cases} 0, & \text{if } i = j \\ 1, & \text{otherwise} \end{cases} \qquad \text{and } \text{Label}(d_i) = \ominus$$

The dataset is then augmented with an additional entry d_{n+1}, which contains solely 1-entries and a positive class label $(1, \ldots, 1, \oplus)$. In this family of datasets, every pattern is closed. The total number of closed patterns is thus 2^n, while there is only one closed-on-the-positives, namely $\{a_1 \ldots a_n\}$. The below Table 2.5 illustrates such a construction for four attributes.

Id	a_1	a_2	a_3	a_4	Label
1	0	1	1	1	$\ominus$
2	1	0	1	1	$\ominus$
3	1	1	0	1	$\ominus$
4	1	1	1	0	$\ominus$
5	1	1	1	1	$\oplus$

Table 2.5.: A dataset with exponentially fewer closed-on-the-positive patterns than regular-closed ones.

Finally, compared to Closed$^+$SD, we see that in worst-case the iterative deepening approach causes an additional factor of n (the second term involving k is not much of a problem, as in practice k is relatively small). For large datasets, this disadvantage is however outweighed by the reduction of the memory footprint, which allows RelevantSD to work datasets that cannot be processed by Closed$^+$SD. Moreover, as the following section will show, in practice this worst-case seldom happens: on real datasets, due to the use of pruning, RelevantSD is mostly able to outperform Closed$^+$SD.

2.2.5. Evaluation

In this section, the new relevant subgroup discovery algorithm will be compared empirically with other existing algorithms. In particular, the following two questions were considered:

- How does the algorithm perform compared to Closed$^+$SD?
- How does the algorithm perform compared to classical and closed subgroup discovery algorithms?

This section does not investigate and quantify the advantage of the relevant subgroups over standard or closed subgroups, as the value of the relevance criterion on similar datasets has already been demonstrated by Garriga et al. (2008).

Implementation and Setup

RelevantSD was implemented in JAVA, without the usage of sophisticated data structures like fp-trees (Han et al., 2000) or bitsets (Lemmerich and Atzmüller, 2010). As minor optimization, during the iterative deepening the length limit is increased in a way that length limits for which no patterns exist are skipped (this is realized by keeping track, in every iteration, of the length of the shortest pattern expanded exceeding the current length limit).

In the following investigation, nine datasets from the UCI Machine Learning Repository (Asuncion and Newman, 2007) were used, which are presented along with their most important properties in Table 2.6. All numerical attributes where discretized using minimal entropy discretization. The experiments were run, using two quality functions: the binomial test quality function and the WRACC quality (Equation 2.1 with $\alpha = 0.5$ and $\alpha = 1$). For pruning, different optimistic estimators were used for different quality functions. For the WRACC quality, the optimistic estimate $|TP(\mathcal{D},p)|^2/|\mathcal{D}[p]| \cdot (1 - |TP(\mathcal{D},\varnothing)|/|\mathcal{D}|)$ from Grosskreutz et al. (2008) was used, while for the binomial test quality the function $\sqrt{|TP(\mathcal{D},p)|} \cdot (1 - |TP(\mathcal{D},\varnothing)|/|\mathcal{D}|)$ was used as optimistic estimator. These estimates were used in all implementations to make sure that the results are comparable. The experiments were run on a Core2Duo 2.4 GHz PC with 4 GB of RAM.

Comparison with Closed$^+$SD

This paragraph, draws a comparison between the algorithms RelevantSD and Closed$^+$SD. In order to abstract from the implementation, the number of visited nodes was compared, rather than the runtime or the exact amount of memory used.

First, Figure 2.2 shows the number of nodes considered by RelevantSD and by Closed$^+$SD. Here, the binomial test and the WRACC quality measures were used, for two values of k, namely 10 and 100. For the 'splice' dataset, the number of nodes considered by

Dataset	#data records	#features	target class
credit-g	1000	58	bad
lymph	148	50	mal_lymph
mushroom	8124	117	poisonous
nursery	12960	27	recommend
sick	3772	66	sick
soybean	638	133	brown-spot
splice	3190	287	EI
tic-tac-toe	958	27	positive
vote	435	48	republican

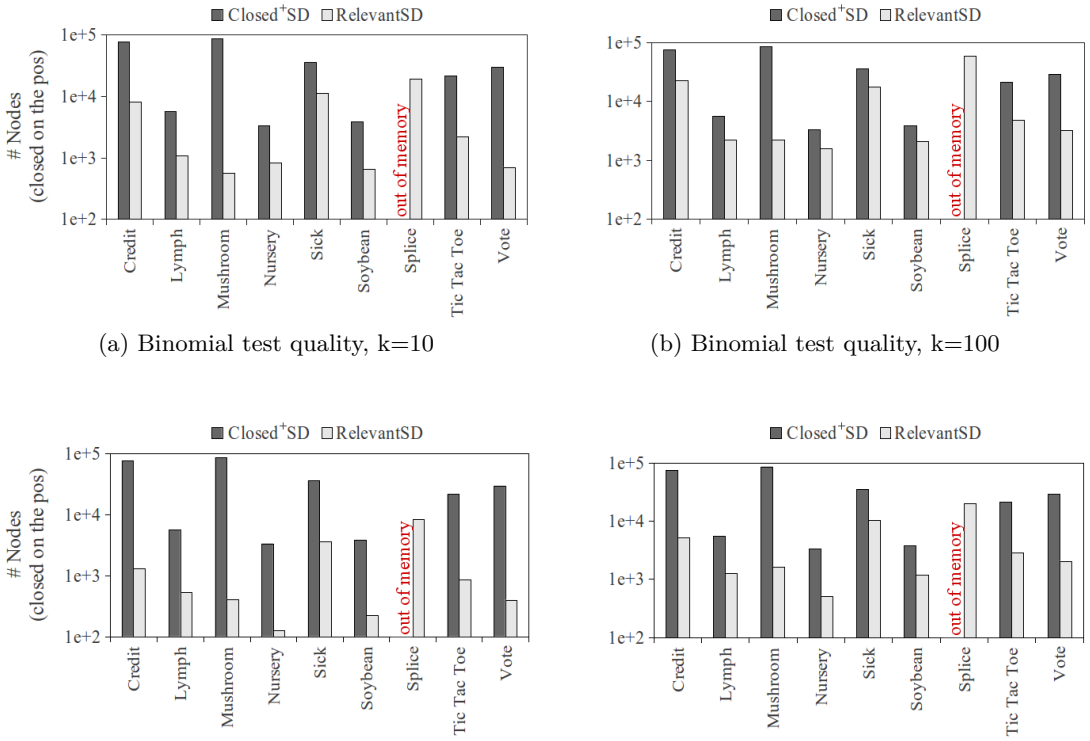
Table 2.6.: Datasets used in the evaluation.

Closed$^+$SD was almost 100 millions. As the algorithm Closed$^+$SD has to keep all visited nodes in memory, the computation failed, with an "out of memory" exception. This illustrates that the memory footprint of Closed$^+$SD can already for datasets of moderate size become prohibitive. The RelevantSD algorithm, on the other hand, has no need to keep all visited patterns in memory, and hence all computations succeeded. Moreover, in total the RelevantSD approach considers way less nodes than Closed$^+$SD.

Comparison with other subgroup miners

In this section, the RelevantSD algorithm is compared with subgroup miners that solve a different but related task, namely *classical* subgroup discovery and *closed* subgroup discovery. As representatives, the algorithms DP-subgroup (Grosskreutz et al., 2008) and the depth-first closed subgroup miner from Boley and Grosskreutz (2009) were used. The latter one is essentially an adaptation of LCM (Uno et al., 2004) to the task of subgroup discovery. Note that the results are also representative for approaches like the algorithms CN2-SD (Lavrač et al., 2004) and BSD (Lemmerich and Atzmüller, 2010), which both operate on the space of all patterns.

As the compared algorithms originate from different sources, written by different programmers in different languages, comparing the number of visited nodes during the traversal of the search space seems to be the most common indicator of the runtime. Figure 2.3 shows the number of nodes considered by different algorithms for k set to 10 and 100, and for the binomial test and the WRACC quality function respectively. Please note that for RelevantSD, all nodes are closed-on-the-positives, while for the closed subgroup discovery approach (ClosedSD) they are closed and for the classic approach (DP-subgroup) they are arbitrary subgroup descriptions. The results differ strongly depending on the characteristics of the data. For several datasets, using RelevantSD results in a decrease of the number of nodes considered. The difference to the classical subgroup miner DP-subgroup is particularly apparent, as it sometimes amounts to several orders



(a) Binomial test quality, k=10

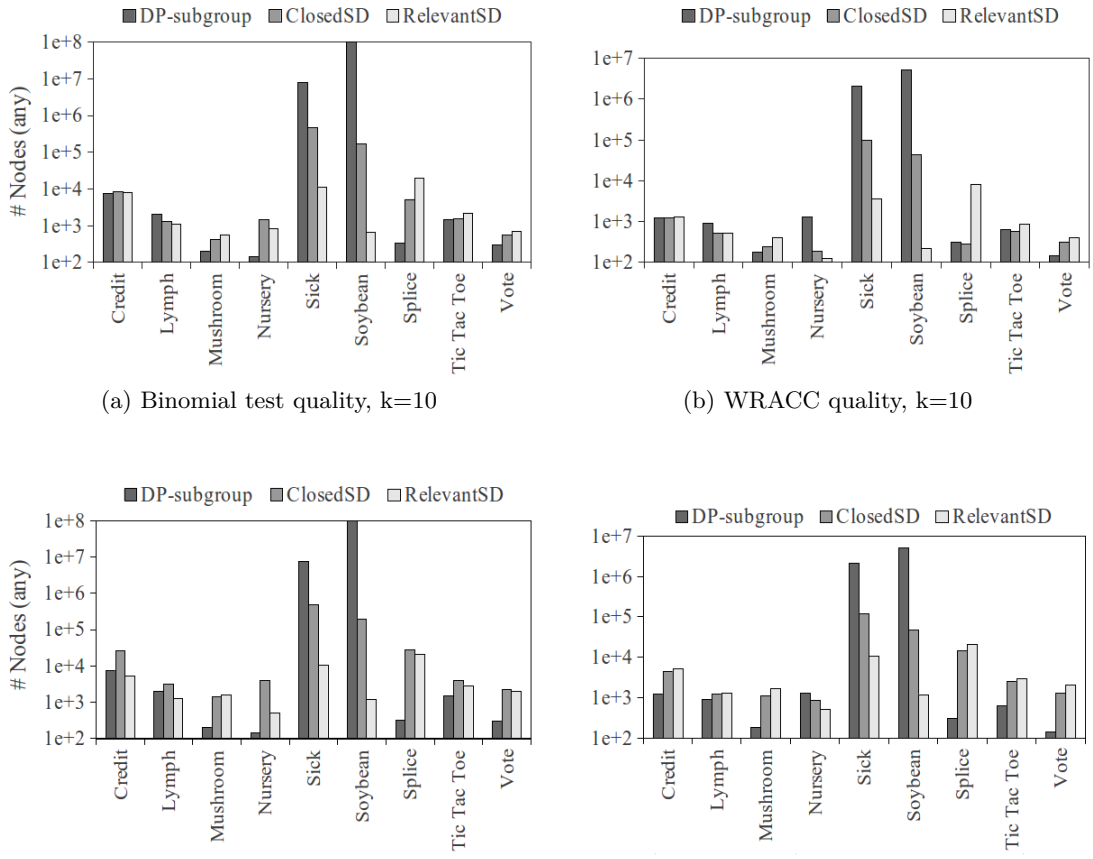
(b) Binomial test quality, k=100

(c) WRACC quality, k=10

(d) WRACC quality, k=100

Figure 2.2.: The number of nodes considered during relevant subgroup discovery for the RelevantSD and the Closed$^+$SD algorithm for different quality measures and k's.

of magnitude. There are, however, several datasets where the RelevantSD algorithm traverses more nodes than the classical approaches. Again, the effect is particularly considerable when compared with the classical subgroup miner. Beside the overhead caused by the multiple iterations, one reason for this effect is that the quality of the k^{th} pattern found differs for the different algorithms: for the relevant subgroup algorithm, the k-best quality tends to be lower, because this approach suppresses high-quality but irrelevant patterns. One could argue that it would be more fair to use a larger k-value for the non-relevant algorithms, as their output contains more redundancy. Overall, the RelevantSD algorithm can in practice compete with the other approaches. Although the costs-per-node are lower for classical subgroup discovery than for the other approaches, it does not compensate for the much larger number of nodes traversed, as the aggregated Table 2.7 shows.



(a) Binomial test quality, k=10

(b) WRACC quality, k=10

(c) Binomial test quality, k=100

(d) WRACC quality, k=100

Figure 2.3.: Number of nodes considered by (non-relevant) pattern mining algorithms for different quality measures and k values.

	DP-subgroup	ClosedSD	RelevantSD
total number of visited nodes	346'921'363	1'742'316	590'068
percentage vs. DP-subgroup	100%	0.5%	0.17%
total Runtime [sec]	2'717	286	118
percentage vs. DP-subgroup	100%	10.5%	4.4%

Binomial test quality function

	DP-subgroup	ClosedSD	RelevantSD
total number of visited nodes	15'873'969	459'434	120'967
percentage vs. DP-subgroup	100%	2.9%	0.76%
total Runtime [sec]	147	100	45
percentage vs. DP-subgroup	100%	68%	30%

WRACC quality function

Table 2.7.: Total number of nodes visited for the algorithms DP-subgroup, ClosedSD and RelevantSD, and percentage compared to DP-subgroup (k=10). The table is split by the different quality measures, binomial test quality and weighted relative accuracy.

2.3. Δ-Relevant Patterns

The theory of relevance provides a solid theoretical approach to the suppression of useless patterns, and has shown to be of great use in practical applications (see e.g. Lavrač and Gamberger (2005)). However, the stringent definition of dominance can in practice and especially for noisy data lead to unsatisfying results. The basic problem is depicted in the following Figure 2.4. Here, we have aligned (or projected) the instances in a two-dimensional space, and the green organically shaped area highlights a subset with a high share of positively labeled instances (marked as "+"). The figure also shows three patterns, visualized by rectangular boxes, that cover some of these instances. While it is clear that all patterns are correct and non-identical descriptions of parts of the target concept, they are highly correlated. The idea behind Δ-relevant pattern discovery is to sacrifice a small amount of precision in favor of a more condensed result set. In the here presented case, a single pattern would suffice to describe the target concept reasonably well.

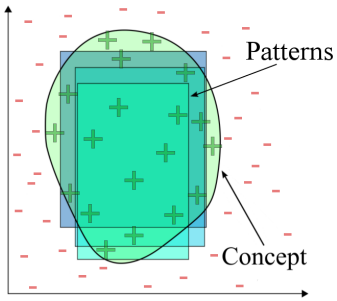
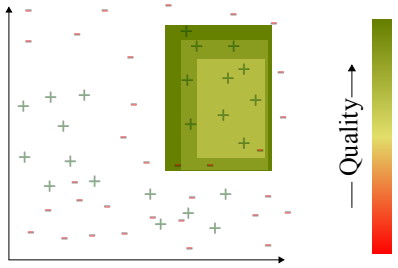
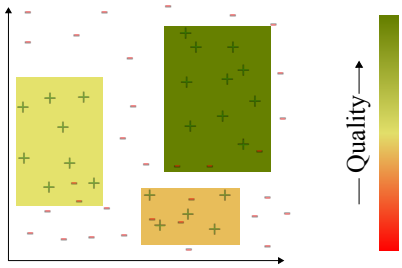


Figure 2.4.: A highly correlated set of patterns (boxes) that approximate a concept (area of positive labeled instances). If a small loss of precision is acceptable in order to avoid redundancy, only one of these patterns has to be reported.

Let us have a look at a short example why a criterion like Δ-relevance is useful. Let p_1 be a pattern which covers 1000 positive and 1 negative example. Further, let p_2 be a specialization of p_1 ($p_1 \sqsubset p_2$) that covers almost exactly the same examples, with the exception of one positive and the negative one. Obviously, neither of the patterns dominates each other, because although the true positives of p_2 are contained in the true positives of p_1, the one false positive of p_1 is missing for p_2 (intuitively speaking, p_1 is better than p_2 on the positives, but worse on the negatives). Although these patterns do not dominate each other, in practice, however, one common representative for both patterns would suffice. As redundant patterns occupy valuable space in the set of finally reported k patterns which could have been given to other patterns that bear more insights. This problem can be remedied by allowing for some controlled slack in the definition of relevance. Figure 2.5 tries to illustrate this.



(a) The relevant top-3 approach would deliver the three relevant patterns with the highest quality. However, it is possible that all these patterns revolve around the same concept.



(b) The Δ-relevant approach tries to ignore roughly redundant descriptions of the same concept. Thus, leaving room for other, diverse, patterns to enter the top-3 list.

Figure 2.5.: Condensing redundant patterns creates space for new patterns to enter the result list. These new patterns offer the potential to uncover yet unknown concepts.

Motivated by this example, in the following, a definition of dominance will be introduced that allows a share Δ of missing false positives in the dominated pattern. I.e. the dominated pattern may be better than the dominating one on the negatives, but only slightly so. Thus, in the above example p_2 would be dominated by p_1 for an appropriate choice of Δ.

2.3.1. Δ-Dominance

Recalling the definition of domination from Section 2.2 - equation 2.2, a closed pattern p_{irr} is dominated by the closed pattern p in database $\mathcal{D}$ if and only if

i) $TP(\mathcal{D}, p_{irr}) \subseteq TP(\mathcal{D}, p)$ and

ii) $FP(\mathcal{D}, p) \subseteq FP(\mathcal{D}, p_{irr})$.

In order to formulate the enhanced version of relevant pattern discovery, first the notation of dominance has to be generalized. To this end, the notation of Δ-domination is introduced, where Δ is a real-valued parameter between 0 and 1. The larger the value of Δ, the lower the requirements for a pattern to be dominated, and thus the more patterns are (potentially) dominated.

Definition (Δ**-dominance**) Let Δ be some value, $0 \leq \Delta < 1$. The pattern p_{irr} is Δ-dominated by the pattern p in database $\mathcal{D}$ iff.

i) $TP(\mathcal{D}, p_{irr}) \subseteq TP(\mathcal{D}, p)$ and

ii) $|FP(\mathcal{D}, p) \setminus FP(\mathcal{D}, p_{irr})| \leq \frac{\Delta}{1-\Delta} |\mathcal{D}[p_{irr}]|$

Intuitively, this definition says that p_{irr} is Δ-dominated if it supports the same or less positives than the dominating pattern, and the number of additional negatives is relatively small compared to the overall size of the pattern p_{irr}. Meaning, it allows a dominating pattern to have an arbitrary large growth in the covered positives, while limiting the additional negatives.

Note that also alternative definitions of the second constraint would be possible, e.g. using a constant bound on the number of additional false positives. Boley et al. (2009) studied a similar scenario, for strongly closed patterns. The main advantage of the relative definition is that it allows us to quantify the relation between the share of positives in the dominated pattern and in the dominating patterns:

Proposition 5 *If a pattern p_{irr} is Δ-dominated by a pattern p_{rel}, then the share of positives in p_{rel} is no lower than $(1-\Delta)$ times the share of positives in p_{irr}. Formally,*

$$\frac{|TP(\mathcal{D}, p_{rel})|}{|\mathcal{D}[p_{rel}]|} \geq (1-\Delta)\frac{|TP(\mathcal{D}, p_{irr})|}{|\mathcal{D}[p_{irr}]|}.$$

With this proposition, Δ becomes a parameter with a clearly defined semantic, namely the control of the trade-off of compression versus allowed loss of precision $\text{prec}(p) = {|TP(\mathcal{D}, p_{rel})|}/{|\mathcal{D}[p_{rel}]|}$. Hence, a user can easily choose an adequate value of Δ that is consistent with his practical application requirements. Note that the question of how much loss of precision is still considered as tolerable is an external decision, hence it is necessary to include a user-controllable parameter for this step.

Proof of Proposition 5. We know that the pattern p_{rel} has to be a generalization of p_{irr}, implying that $FP(\mathcal{D}, p_{rel}) \supseteq FP(\mathcal{D}, p_{irr})$. Further by definition $FP(\mathcal{D}, p_{irr})$ has an empty intersection with $FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})$. Knowing this, we can stat that the false positives of p_{rel} are the false positives of p_{irr} plus any additional false positives p_{rel} makes, leading us to $|FP(\mathcal{D}, p_{rel})| = |FP(\mathcal{D}, p_{irr})| + |FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})|$. Now we can state that

$$\frac{|TP(\mathcal{D}, p_{rel})|}{|\mathcal{D}[p_{rel}]|} = \frac{|TP(\mathcal{D}, p_{rel})|}{|TP(\mathcal{D}, p_{rel})| + |FP(\mathcal{D}, p_{rel})|}$$

$$= \frac{|TP(\mathcal{D}, p_{rel})|}{|TP(\mathcal{D}, p_{rel})| + |FP(\mathcal{D}, p_{irr})| + |FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})|}$$

From the first condition of relevance we know that $|TP(\mathcal{D}, p_{rel})| \geq |TP(\mathcal{D}, p_{irr})|$, so we can conclude that the above must be larger than, or equal to

$$\geq \quad \frac{|TP(\mathcal{D}, p_{irr})|}{|TP(\mathcal{D}, p_{rel})| + |FP(\mathcal{D}, p_{irr})| + |FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})|}.$$

Knowing that $|TP(\mathcal{D}, p_{rel})| + |FP(\mathcal{D}, p_{irr})| \geq |TP(\mathcal{D}, p_{irr})| + |FP(\mathcal{D}, p_{irr})| = |\mathcal{D}[p_{irr}]|$ we can conclude that the above must be larger than, or equal to

$$\geq \quad \frac{|TP(\mathcal{D}, p_{irr})|}{|\mathcal{D}[p_{irr}]| + |FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})|}$$

Moreover, from the second condition we have

$$|FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})| \leq \frac{\Delta}{1-\Delta} |\mathcal{D}[p_{irr}]|$$

which together from the earlier inequality implies the proposition. □

2.3.2. Properties of the Generalized Dominance

In this part, we will prove some important properties of the generalized definition of dominance that will be of importance later. As already mentioned, a higher value of Δ has the effect that potentially more patterns are dominated. In fact, the following implication holds:

Lemma 6 *If a pattern p Δ-dominates another pattern p_{irr}, then the pattern p Δ'-dominates p_{irr} for all $\Delta' > \Delta$.*

Proof This follows from the fact that the function $f(\Delta) = \frac{\Delta}{1-\Delta}$ is monotonically increasing for $\Delta < 1$.

In particular, every pattern dominated according to the classical definition of dominance is Δ-dominated for arbitrary Δ, as 0-dominance is equivalent to classical dominance. The definition of Δ-dominance implies additional properties, which are similar to those of the classical definition:

Lemma 7 *The following holds:*

1. *Every pattern p_{notCl} not closed on the positives is dominated by some pattern p which is closed on the positives; (and, hence, also Δ-dominated);*
2. *Let p_1, p_2 be different patterns closed on the positives. If p_1 is Δ-dominated by p_2, then p_2 is a generalization of p_1;*
3. *Every pattern that is closed on the positives and that is Δ-dominated by some pattern is also Δ-dominated by a closed-on-the-positive.*

Proof For **item 1**, let p_{notCl} be some pattern not closed on the positives. Then p_{notCl} is dominated by $p_{closed} = \Gamma^+(p_{notCl})$. This can be verified by checking the conditions of the definition of domination: 1. both patterns support the same set of positives (by definition), hence the first condition is satisfied. 2. p_{closed} supports a subset of the examples supported by p_{notCl}, hence the second condition is satisfied.

Item 2: By condition 1 of the definition of Δ-relevance, the TP supported by p_1 are all supported by p_2. Hence, the closure on the positives of p_1 is a superset of the pattern p_2. As p_1 is closed on the positives, p_1 must be a specialization of p_2.

Item 3: Assume that p_{irr} is dominated by some pattern p_{notCl} not closed on the positives. Then, it must also be dominated by $p_{closed} = \Gamma^+(p_{notCl})$: first, p_{closed} and p_{notCl} support the same positives. Second, by Definition 2.3.1 we know that $|FP(\mathcal{D}, p_{notCl}) \setminus FP(\mathcal{D}, p_{irr})| \leq {}^{\Delta}/_{(1-\Delta)} \cdot |\mathcal{D}[p_{irr}]|$. Moreover, the false positives of p_{notCl} are a subset of the false positives of p_{closed}, which implies $|FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{irr})| \leq {}^{\Delta}/_{(1-\Delta)} \cdot |\mathcal{D}[p_{irr}]|$ and completes the proof.

In the following, an example is presented that illustrates the above and to shows that dominance misses on one particular property, namely transitivity. Please consider the example dataset which is given in Table 2.8. It includes three attributes, a_1, a_2 and a_3, and the support set for each of these attributes as singleton pattern is a subset of the support set of the previous attributes ($\mathcal{D}[a_1] \supset \mathcal{D}[a_2] \supset \mathcal{D}[a_3]$).

Id	a_1	a_2	a_3	Label
1	1	1	1	$\oplus$
2	1	1	0	$\ominus$
3	1	1	0	$\oplus$
4	1	0	0	$\ominus$
5	1	0	0	$\oplus$
6	0	0	0	$\ominus$

Table 2.8.: A dataset which illustrates the lack of transitivity in the Δ-dominance relation.

The space of pattern descriptions of this example is visualized in Figure 2.6, together with the records that support the different patterns (positive records are rendered with a gray filling). The pattern $\{a_3\}$ (which is has the same extention as the pattern $\{a_1 \wedge a_2 \wedge a_3\}$) supports only one record, which is positive; the pattern $\{a_2\}$ (resp. $\{a_1 \wedge a_2\}$) supports three records, two of which are positive; and finally $\{a_1\}$ supports 5 records, three of which are positive.

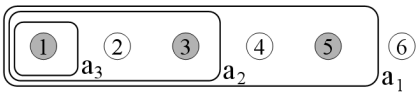


Figure 2.6.: The Δ-dominance relation is not transitive.

For $\Delta = 0$, Δ-domination does not differ from the classical definition of domination, so none of the above patterns are dominated. For $\Delta = 0.5$, however, $\{a_1\}$ Δ-dominates $\{a_1 \wedge a_2\}$, but $\{a_1\}$ does *not* 0.5-dominate $\{a_1 \wedge a_2 \wedge a_3\}$. As the true positives of $\{a_1 \wedge a_2\}$ (the data records 1 and 3) are also true positives of $\{a_1\}$, there exists 1 false positive of $\{a_1\}$ that is no false positive of $\{a_1 \wedge a_2\}$ (the data record 4), $\{a_1 \wedge a_2\}$ covers a total of 3 examples (1,2, and 3), and $1 \leq 3 \times 0.5$. Similarly, it is easy to verify that $\{a_1 \wedge a_2\}$ 0.5-dominates $\{a_1 \wedge a_2 \wedge a_3\}$. This demonstrates that the Δ-dominance relation is *not transitive*, which is still an open issue.

2.3.3. A Generalized Definition of Relevance

Now that we have defined Δ-dominance, we turn to the definition of Δ-relevance. It might seem at first that Δ-relevance can be defined in complete analogy to classical relevance (where a pattern is relevant if it is not dominated by any other pattern). This, however, is problematic due to the different characteristics of Δ-dominance, namely the missing transitivity. If we would define patterns to be Δ-relevant in the sense that they are not Δ-dominated by any other pattern, then (unlike in classical relevance) patterns could be suppressed, although the result set does not contain a Δ-dominating pattern. To avoid such situations, we base the definition of Δ-relevance on the concept of a covering of Δ-dominating patterns:

Definition (**covering**) Given a dataset $\mathcal{D}$, a set of patterns S is called a **covering** of Δ-relevant patterns for $\mathcal{D}$ iff.

- For every pattern p in $\mathcal{D}$, there is a pattern $p' \in S$ that Δ-dominates p.
- There are no two patterns $p_1, p_2 \in S$ such that p_1 Δ-dominates p_2.
- Every pattern in S is closed on the positives.

The first condition ensures that S is a covering, that is, every pattern has a Δ-relevant representative. The second requires it to be minimal. The third condition says that we want the covering to consist only of patterns closed on the positives, which we can use as representatives.

The practical significance of this definition is that in combination with Proposition 5 we can now guarantee that if one is willing to tolerate an indistinctness of at most Δ, it suffices to look only at the patterns within the covering. Part 1 of the definition guarantees that for all other patterns we can find a pattern in the covering which is almost as good.

While this definition seems to leave a high degree of freedom, in fact it specifies precisely a single set of patterns. These can be found iteratively, based on the specialization graph defined on the patterns closed-on-the-positives. This graph $G = (V, E)$ has, as vertex set V, the set of closed-on-the-positives, and its edge set consists of all (p, p') such that p' is a specialization of p (not necessarily a direct specialization). Based on this graph, the covering of Δ-relevant patterns can be constructed as described in Algorithm 2. In the following it will be shown that these construction rules correctly calculate a set of patterns *Cover* that is a covering.

Algorithm 2 Covering of Δ-relevant Patterns

Input : A value $0 \leq \Delta \leq 1$, a database $\mathcal{D}$ over attributes $\{f_1, ..., f_n\}$, a binary label from the set $\{\oplus, \ominus\}$ and a closure operator
Output : The cover of Δ-relevant subgroups

1. **let** G = the specialization graph of all patterns $\in \mathcal{D}$ that are closed-on-the-positives
2. **let** *Cover* := $\{c \in G \mid c$ has no generalization in $G\}$
3. **while** G is not empty **do**
4. G := the graph obtained from G by removing all Δ-dominated specializations of nodes in *Cover* (and the corresponding edges).
5. **let** *Cover* := *Cover* $\cup$ $\{c \in G \mid c$ has no generalization in $G\}$
6. **end while**
7. **return** *Cover*

Proof We first show inductively that all nodes in *Cover* must be part of the covering. (Base case:) The set of vertices selected in Line 2 have to be part of the covering, because (i) they are closed on the positives, and (ii) there isn't any closed-on-the-positive generalization of them. Thus, according to Lemma 7 they are not Δ-dominated by any other pattern and hence they must be part of the covering.

(Inductive step:) All nodes removed in 4 cannot be part of any covering. This follows directly from the second part of the definition of a covering together with the fact that the covering already includes a pattern that Δ-dominates them.

Moreover, all nodes added to the covering in Line 5 must be part of the covering. This follows from Lemma 7 and the facts that (i) the covering does not yet include a pattern which Δ-dominates them, and (ii) there isn't any Δ-dominating pattern which could alternatively be added to the covering, because for all generalizations we already know that they either cannot be part of the covering or are not Δ-dominating.

The fact that the algorithm ends follows from the facts that the (initial) vertex set is finite, and that in every loop the graph is replaced by a graph with strictly less vertexes (because G has no cycles).

Finally, *Cover* must be a covering because for every vertex resp. pattern, either it is in the covering or there is a dominating pattern in *Cover*. This directly follows from the construction of the algorithm for patterns that are closed on the positives. For all other patterns p_{notCl}, there are two different cases: Either their closure in the positives, $p_{closed} = \Gamma^+(p_{notCl})$ is a member of *Cover* (in this case they are obviously dominated), or their closure p_{closed} is dominated by some member p_{rel} of *Cover*. By statement 2 of Lemma 7, p_{rel} is a generalization of p_{closed}. Hence, the true positives of p_{rel} are a superset of the true positives of p_{closed}, resp. p_{notCl} and the first condition of the definition of Δ-dominance is satisfied. It remains to show that the second condition also holds. As p_{rel} dominates p_{closed}, hence by Definition 2.3.1 we have $|FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{closed})| \leq {}^{\Delta}/_{(1-\Delta)} \cdot |\mathcal{D}[p_{closed}]|$. Moreover, the false positives of p_{notCl} are a superset of the false positives of p_{closed}, and the support set of p_{notCl} is a superset of the support set of p_{closed}. This implies $|FP(\mathcal{D}, p_{rel}) \setminus FP(\mathcal{D}, p_{notCl})| \leq {}^{\Delta}/_{(1-\Delta)} \cdot |\mathcal{D}[p_{notCl}]|$. □

Based on this, we finally define Δ-relevance as follows:

Definition (Δ**-relevant pattern**) For a dataset $\mathcal{D}$, a subgroup, or pattern, is called Δ-relevant if and only if it is a member of $\mathcal{D}$'s unique covering.

2.3.4. A Caveat

It is interesting to notice that a property like Lemma 6 (dealing with Δ-dominance) does not hold for Δ-relevance: it is possible that a pattern becomes Δ-irrelevant for one value of Δ, but that for another $\Delta' > \Delta$, becomes Δ'-relevant again. This can lead to the situation that a covering for one value of Δ is actually smaller than that for a higher value Δ'. For example, consider the data set in Table 2.9. For $\Delta = 0.5$, the pattern $\{a_1 \wedge a_2\}$ dominates the patterns $\{a_1 \wedge a_2 \wedge a_3\}$ and $\{a_1 \wedge a_2 \wedge a_4\}$, but is not dominated by $\{a_1\}$. Hence, the relevant patterns for $\Delta = 0.5$ are $\{a_1\}$ and $\{a_1 \wedge a_2\}$. For $\Delta = 0.56$,

$\{a_1\}$ dominates $\{a_1 \wedge a_2\}$, but does dominate neither $\{a_1 \wedge a_2 \wedge a_3\}$ nor $\{a_1 \wedge a_2 \wedge a_4\}$, so there exists three relevant patterns, $\{a_1\}$, $\{a_1 \wedge a_2 \wedge a_3\}$, and $\{a_1 \wedge a_2 \wedge a_4\}$. Again, this has to do with the missing transitivity of dominance: $\{a_1 \wedge a_2\}$ does still dominate $\{a_1 \wedge a_2 \wedge a_3\}$ and $\{a_1 \wedge a_2 \wedge a_4\}$, but as $\{a_1 \wedge a_2\}$ is suppressed from the result set, it does not exclude its two specializations.

Id	a_1	a_2	a_3	a_4	Label
1	1	1	1	0	⊕
2	1	1	0	1	⊕
3	1	1	0	0	⊕
4	1	0	0	0	⊕
5	1	0	0	0	⊖
6	1	0	0	0	⊖
7	1	1	0	0	⊖
8	1	0	0	0	⊖
9	1	0	0	0	⊖
10	1	0	0	0	⊖

Table 2.9.: The size of the Δ-relevant pattern set is not monotonous in Δ

Figure 2.7 shows the graph with the dominance dependencies between these patterns together with the minimal Δ for which the dominance holds. The experiments, however, will demonstrate that this problem usually does not occur in practice.

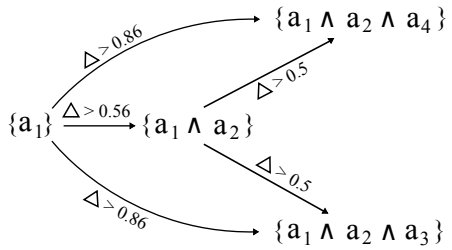


Figure 2.7.: The dominance graph of Example 2.9.

2.3.5. Listing Δ-Relevant Patterns

Algorithm 2 computes the unique covering of Δ-relevant patterns for a database. While the covering of Δ-relevant patterns can be much smaller than the whole set of (relevant) patterns, depending on the database this set can still be unacceptably large. As already mentioned in Section 2.1.3, one standard approach to deal with this issue is to rank the pattern according to some quality function and keep only the top-k patterns.

This section describes how the new concept of Δ-relevance can be combined with top-k approaches – more precisely, how it can be used to devise an improved pattern discovery algorithm. Using the generalized notion of relevance, the task of pattern discovery will be extended as follows:

Task (**top-k Δ-relevant pattern discovery**) Given a database $\mathcal{D}$, a quality function q, an integer $k > 0$ and a real value Δ, $0 \leq \Delta < 1$, find a set of patterns G of size k, such that

(i) all patterns in G are Δ-relevant, and

(ii) all Δ-relevant patterns not in G have a quality no higher than $\min_{p \in G} q(\mathcal{D}, p)$.

Δ-relevant pattern discovery can be performed following Algorithm 2, where for every discovered pattern its quality is calculated. In a subsequent step, all but the top-k patterns are then discarded. However, it is not necessary to compute the graph G beforehand, as this can be done dynamically during the execution of the algorithm. The basic trick is that Property 2 of Lemma 7 allows to search through the space of closed patterns in a general-to-specific order. If one is only interested in the top-k patterns, then large parts of the specialization graph (used in the construction in Sec. 2.3.3) can be pruned using optimistic estimators (see Section 2.1.3). Essentially, all vertices corresponding to patterns with an optimistic estimate below a minimum threshold can be ignored. Even if such a threshold is not specified beforehand, the threshold can dynamically be determined using the quality of the best k relevant patterns so far considered. Algorithm 1 from Section 2.2.4, which finds the top-k relevant patterns in an efficient way, can easily be extended to also find the top-k Δ-relevant patterns.

2.3.6. Evaluation

In this section, the effect of the new definition of relevance is studied from a practical perspective on several benchmark datasets. More precisely, the following two questions are investigated:

1. To what extent does the new definition reduce the number of patterns?

2. Does the use of the stronger relevance criterion improve the quality of the top-k patterns?

To answer these questions, again the datasets from the UCI repository (Asuncion and Newman, 2007), that are listed in Table 2.6, were used to study Δ-relevant pattern discovery in practice. To stay comparable, the datasets are the same that were already used for the evaluation of the unrefined ($\Delta = 0$) relevant pattern mining approach.

Reduction of the Number of Patterns

This section aims at showing that Δ-relevance significantly reduces the number of patterns that are found in a database. Boley and Grosskreutz (2009) have already shown that closed patterns very much reduce the number of patterns in comparison to all possible patterns, while Garriga et al. (2008) have shown that going from closed pattern to relevant patterns again very much reduces the amount of patterns found. Hence, it suffices to show that going from relevant patterns, i.e. Δ-relevant patterns for $\Delta = 0$, to Δ-relevant patterns for $\Delta > 0$ reduces the number of patterns.

Figure 2.8 shows how the number of Δ-relevant patterns reduces depending on Δ. It can be seen that for all datasets, the number of patterns reduces with increasing Δ.

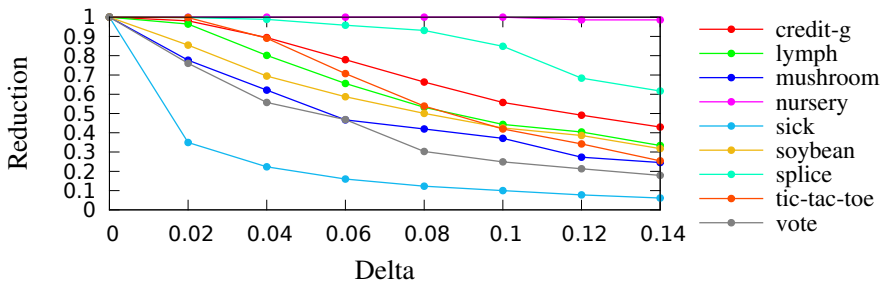


Figure 2.8.: Reduction of Δ-relevant patterns found (percentage, y-axis) depending on Δ (x-axis) for the Piatetsky-Shapiro quality measure.

Quality of the Patterns

Despite the reduction of the output space, the number of Δ-relevant patterns found can still be too large to handle. Therefore, a combination with a top-k approach is advisable. The benefit that comes with the usage of Δ-relevant patterns is that the top-k Δ-relevant patterns are less redundant and more "interesting" than classical top-k patterns.

In the following, first the redundancy of the top patterns will be examined by using a visualization technique introduced in van Leeuwen and Knobbe (2011). Thereafter, the effect of the Δ parameter will be investigated for building predictive classifiers from the top patterns. While optimizing accuracy is not the primary goal of the pattern discovery methods, proposed by Lavrač et al. (2004), it is a common approach to evaluate predictive accuracy in the absence of a better option to capture the "interestingness" of the patterns.

Redundancy

To illustrate how mining the Δ-relevant patterns affects redundancy, a visualization proposed by van Leeuwen and Knobbe (2011) is used. Here, the coverage of a set of patterns, i.e. the set of data records satisfied by the individual patterns, is visualized as a rectangular plot of black and white pixels. The plot has one row of pixels for every pattern. Similarly, there is one column for every data record. The pixel at location (x, y) is plotted in black if the x-th pattern is supported by data record y; else, the pixel is plotted in white. If a set of patterns is highly redundant, then the plot will reveal noticeable vertical patterns: the reason is that the coverage of the patterns, and hence the rows visualizing them, will be very similar. Table 2.10 shows an individual plot for the top-k patterns found using different approaches. On top, we show the plot for the top-20 classic subgroups; next, the plot for the top-20 closed patterns; thereafter, the top-20 relevant patterns and finally the top-20 $\Delta_{0.1}$-relevant patterns. In these plots, more vertical lines imply a higher degree of redundancy among the top-20 patterns, which indicates slightly more diversity among the Δ-relevant patterns.

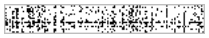
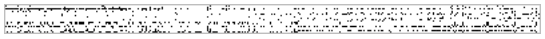
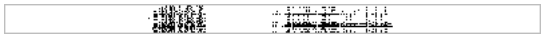
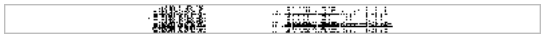
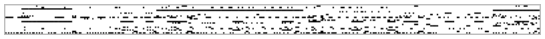
Dataset	Method	Redundancy figure
credit-g	classic	
	closed	
	relevant	
	Δ-relevant	
lymph	classic	
	closed	
	relevant	
	Δ-relevant	
mushroom	classic	
	closed	
	relevant	
	Δ-relevant	
sick	classic	
	closed	
	relevant	
	Δ-relevant	
soybean	classic	
	closed	
	relevant	
	Δ-relevant	
tic-tac-toe	classic	
	closed	
	relevant	
	Δ-relevant	
vote	classic	
	closed	
	relevant	
	Δ-relevant	

Table 2.10.: Studying the redundancy of the top-20 patterns using classic, closed, relevant and Δ-relevant patterns (for a Δ of 0.1) on five of the datasets. Depicted is, for the first 400 data records, whether it supports a pattern (black dot), or not. For the Δ-relevant patterns, vertical stripes are much less apparent, indicating less redundancy.

Predictive Accuracy

In the following experimental setup, a set of patterns is converted into a predictive model in the following way: for any pattern p the class probability $P_{class}(p) = {}^{|TP(\mathcal{D},p)|}/_{|\mathcal{D}[p]|}$ is computed. To any data record x that supports the pattern p, the predicted class probability $P_{class}(p)$ is assigned. In case x supports more than one pattern, the maximum P_{class} of all covering patterns is assigned to it. In the case where x is not covered by any pattern, the default probability is assigned to it. This allows to compute the Area under the Curve (AUC) of a set of patterns.

Figure 2.9 shows the AUC of the Top 10 Δ-relevant patterns for different values of Δ and the Piatetsky-Shapiro quality function. It can be seen that the AUC tends to increase for $\Delta > 0$ (for 5 datasets, there is a clear improvement; one dataset ("nursery") is completely unaffected; and finally, for two datasets ("lymph" and "vote"), the plot shows that the AUC decreases after reaching a maximum for a value of Δ around 0.05). The results are similar for the binomial quality function in Figure 2.10: again, the AUC tends to increase when Δ-relevant patterns are considered instead of classical relevant patterns.

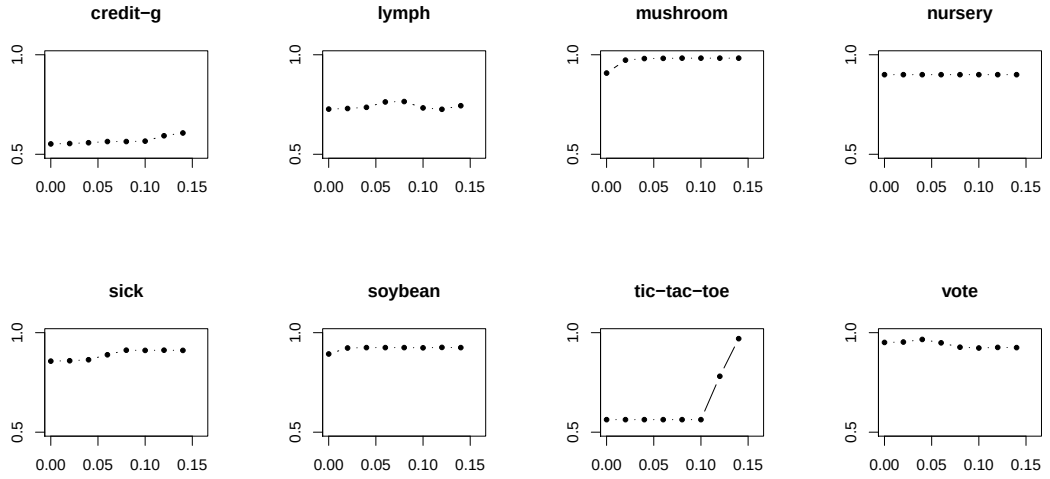


Figure 2.9.: AUC of the top−10 Δ-relevant patterns (y-axis) depending on Δ (x-axis) for the Piatetsky-Shapiro quality measure.

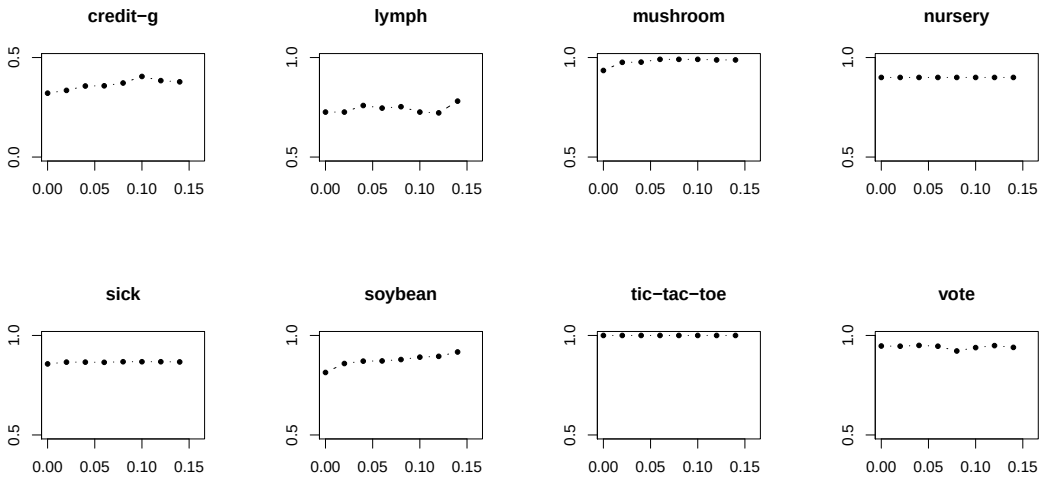


Figure 2.10.: AUC of the top−10 Δ-relevant patterns (y-axis) depending on Δ (x-axis) for the Binomial test quality measure.

2.4. Sampling Interesting Patterns

An alternative approach to systematically listing interesting patterns (like e.g, frequent set mining or optimistic-estimator-based subgroup discovery) is to sample from the space of all possible patterns over the given attribute set. In the following section, a general pattern random sampling framework will be introduced which allows the analyst to draw patterns with a probability that is proportional to a specified interestingness-measure. The framework allows for different kinds of interestingness-measures.

2.4.1. Motivation for Sampling Patterns

Pattern sampling can be of great use if the listing strategy of the mining algorithm does not coincide with the desired measure of interestingness. For instance, an analyst could be interested in patterns that discriminate between two target labels on a dataset. The Fischer score is a measure which captures this well. However, the analyst might choose to utilize a pattern mining algorithm that lists the patterns according to another measure of interest, e.g. the frequency of occurrence. This would not be unreasonable, as the frequent patterns often tell the analyst something about the most dominant attribute combinations of the dataset at hand, thus providing a basic intuition for it. Unfortunately, as many frequent pattern mining algorithms traverse the lattice of all patterns in a general-to-specific order, the most frequent patterns would be listed first and the most interesting ones could be listed last and the analyst might have to wait a long time. With a bit of bad luck, the analyst might have to wait a long time. This can, for instance, occur in datasets with a heavy label imbalance. A pattern that

describes exactly the data records of the minority class would be highly discriminative, but infrequent. A good example of this can be observed e.g. on the *primary-tumor* dataset. Here the most interesting patterns (according to the Fisher score) are among the least frequent ones. Frequent pattern mining algorithm like e.g. Apriori, FP-growth or LCM (Agrawal et al., 1996; Han et al., 2000; Uno et al., 2004) would list literally millions of less interesting patterns, before finally delivering the "nuggets" at the very end.

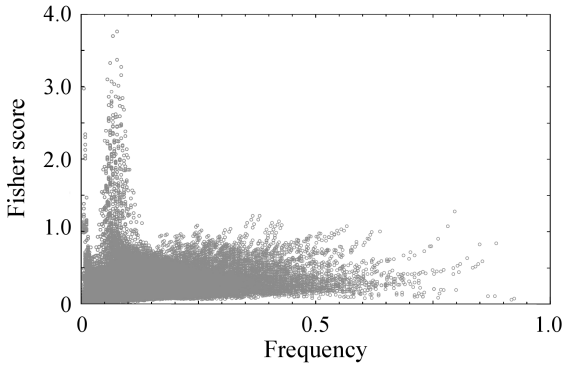


Figure 2.11.: All patterns of *primary-tumor* dataset plotted with their frequency (x-axis) against their Fisher score (y-axis).

Another motivation is that even though pattern mining algorithms strive for speed, their execution time can still take hours or days, depending on the task. This motivates the introduction of algorithms that only sample a representative set of patterns without explicitly searching in the pattern space. Such algorithms exist in the literature (Al Hasan and Zaki, 2009; Boley et al., 2010; Chaoji et al., 2008) but they provide either no control over the distribution of their output or only asymptotic control by simulating a stochastic process on the pattern space using the Markov chain Monte Carlo method (MCMC). In addition to only offering approximate sampling, MCMC methods have a scalability problem: the number of required process simulation steps is often large and, even more critical, individual simulation steps typically involve expensive support counting operations. Hence, these algorithms are often infeasible for large input datasets. In the following, a pattern generation framework is introduced that samples patterns exactly proportional to a probability distribution and directly, i.e., without simulating time-consuming stochastic processes.

2.4.2. Sampling According to Frequency

Before going into technical details, some additional notations have to be introduced. For a finite set X we denote by $\mathcal{P}(X)$ its power set and by $u(X)$ the uniform probability distribution on X. Moreover, for positive weights $w\colon X \to \mathbb{R}^+$ let $w(X)$ denote the distribution on X arising from normalizing w by $x \mapsto w(x)/\sum_{x' \in X} w(x)$. For simplicity the dataset $\mathcal{D}$ is assumed to be binary over some finite attribute ground set $A = \{a_1, \ldots, a_n\}$.

A naive approach for sampling a pattern according to a distribution π is to generate a list of all patterns $p_1, \ldots, p_N$ that have a chance of occurring ($\pi(p) > 0$), draw an $x \in [0,1]$ uniformly at random, and then return the unique set p_k with $\sum_{i=1}^{k-1} \pi(p_i) \leq x < \sum_{i=1}^{k} \pi(p_i)$. However, this exhaustive enumeration of all possible patterns is precisely what we want to avoid. That is, we are interested in a *non-enumerative* sampling algorithm. In the following, Algorithm 3 for frequency-based pattern sampling is introduced. It is inspired by the elementary procedures used by Karp et al. (1989). The key idea for randomly drawing a pattern proportional to its frequency of occurrence, i.e., $\pi = q_{\text{freq}}(\mathcal{P}(A))$, is that random experiments are good in reproducing frequent events. Namely, if we look at a pattern that is supported by a random data record we are likely to observe a pattern that is supported by many data records altogether. This intuition leads the following fast and simple two-step non-enumerative sampling routine:

- First select a data record of the input dataset randomly with a probability that is proportional to the size of its power set,
- then return a uniformly sampled subset of that data record.

The random set, which results from combining those two steps exhibits indeed the desired distribution, as the following proof shows:

Proof Let $Z = \sum_{p \subseteq A} |\mathcal{D}[p]|$ be the normalizing constant and $\mathbf{d}$ denote the random data record that is drawn in step 2 of Algorithm 3. For the probability distribution of the returned random set $\mathbf{r}$ we have

$$\begin{aligned}
\mathbb{P}[\mathbf{r} = r] &= \sum_{d \in \mathcal{D}} \mathbb{P}[\mathbf{d} = d \wedge \mathbf{r} = r] \\
&= \sum_{d \in \mathcal{D}[r]} \frac{1}{2^{|d|}} \frac{2^{|d|}}{Z} \\
&= \frac{|\mathcal{D}[r]|}{Z} \\
&= \frac{\text{supp}(\mathcal{D}, r)}{Z}
\end{aligned}$$

with a normalizing $Z = \sum_{d \in \mathcal{D}} 2^{|d|}$ which is equal to the desired $\sum_{p \subseteq A} |\mathcal{D}[p]|$ and constant for all samples.

The two step procedure for randomly sampling a pattern with a probability that is proportional to its support can now be formalized in the following algorithm:

Algorithm 3 Frequency-based Sampling

Require: dataset $\mathcal{D}$ over attribute ground set A,
Returns: random set $r \sim q_{\text{freq}}(\mathcal{P}(A)) = q_{\text{supp}}(\mathcal{P}(A))$

1. **let** weights w be defined by $w(d) = 2^{|d|}$ for all $d \in \mathcal{D}$
2. **draw** $d \sim w(\mathcal{D})$
3. **return** $r \sim u(\mathcal{P}(d))$

Regarding the computational complexity of the sampling algorithm we can observe that it is indeed efficient, as a single random set can be produced in time $\mathcal{O}(\log|\mathcal{D}| + |A|)$. The two terms correspond to producing a random number for drawing a data record in step 1 and to drawing one of its subsets in step 2, respectively. Both requirements can be achieved via a single initial pass over the dataset. Thus, given an input dataset $\mathcal{D}$ over the attributes A, a family of k realizations of a random set $\mathbf{R} \sim q_{\text{freq}}(\mathcal{P}(A))$ can be generated in time $\mathcal{O}(\|\mathcal{D}\| + k(|A| + \log|\mathcal{D}|))$.

2.4.3. Sampling According to Other Measures

As already stated, it is possible to adapt the idea of Algorithm 3 to sample patterns with a probability proportional to other measures of interest. By adjusting the probability of a data record being drawn in the steps 1 and 2 and refining the rejection procedure in step 3 of Algorithm 3, the returned patterns can be drawn according to other probabilities / interestingness-measures. Boley et al. (2011) and Boley et al. (2012) introduce and discuss a variaty of different adaptions to this two step sampling procedure. Namely pattern sampling according to a patterns *frequency*, its *area*, *discreminativity* and *rarity* are introduced, Here, *discriminativity* referrs to a form of label discriminativity, given by $q_{\text{disc}}(\mathcal{D}, p) = q_{\text{freq}}(\mathcal{D}^+, p) \cdot (1 - q_{\text{freq}}(\mathcal{D}^-, p)$, while *rarity* describes the probability of observing the pattern, weighted by the probabilities of the patterns' items not to occurr; $q_{\text{rare}}(\mathcal{D}, p) = \text{freq}(\mathcal{D}, p) \prod_{p_i \in p}\big(1 - \text{freq}(\mathcal{D}, p_i)\big)$. Boley et al.'s publications introduce also versions of the above mentioned measures that are multiplied with a power of the patterns frequency of occurrence. Examples for this are $\textit{frequency}^2$, $\textit{frequency}^3$, $\textit{area}\times\textit{frequency}$, $\textit{rarity}\times\textit{frequency}^2$ and $\textit{discriminativity}\times\textit{frequency}^+$, where $\textit{frequency}^+$ here denotes the relative support of a pattern only on the positively labeled instances.

A drawback of Algorithm 3 is that the weight w, used in step 2, have to be computed beforehand for each data record. Especially for higher order interestingness measures, as *discriminativity* or measures that incorporate $\textit{frequency}^n$, calculating the initial weights can be costly. In order to realize these more sophisticated interestingness measures, it is necessary to draw an n-tuple of data records. Here the size of the tuple n depends on the complexity of the interestingness measures. This means that the increased expressivity comes at a price: due to the necessity of weight computation for all possible n-tuples of data records, we end up with a space and time complexity of the preprocessing phase

that is polynomial in the order of the cardinality n of the tuples. To counteract this behaviour, Boley et al. (2012) suggest to sample the tuples, based on the simulation of a stochastic process, instead of precomputing all weights in advance. Their proposed approach utilizes *coupling from the past*, a sampling technique that allows to draw the tuples with a controlled probability and limits the memory footprint to be linear. Note that sampling the data record tuples is a random process, such that the execution time can, in theory, exceed the time that would be needed to calculate all sampling weights needed for step 2 of Algorithm 3 in advance. In practice, however, utilizing the proposed *coupling from the past* approach does not only save memory, but also time.

2.4.4. Empirical Demonstration

This section demonstrates the above stated benefits of pattern sampling in an empirical way. Figure 2.12 shows the 10000 patterns from the *primary-tumor* dataset, retrieved by two different algorithms, plotted by their frequency against their Fisher score. The Fisher score is a measure of discriminativity for class labels for an arbitrary number of classes C. It measures the relation of the inter-class variance of a feature to its intra-class variances and is given via:

$$q_{\text{fish}}(p) = \frac{\sum_{c \in C} |\mathcal{D}_c| \cdot [q_{\text{freq}}(\mathcal{D}_c, p) - q_{\text{freq}}(\mathcal{D}, p)]^2}{\sum_{c \in C} \sum_{d \in \mathcal{D}_c} [\delta(d \supseteq p) - q_{\text{freq}}(\mathcal{D}_c, p)]^2}$$

In the above equation, $\mathcal{D}_c$ denotes all data records of a class c and $\delta(d \supseteq p) = \begin{cases} 1, & \text{if } d \supseteq p \\ 0, & \text{otherwise} \end{cases}$ indicates whether a data record d supports a pattern p or not.

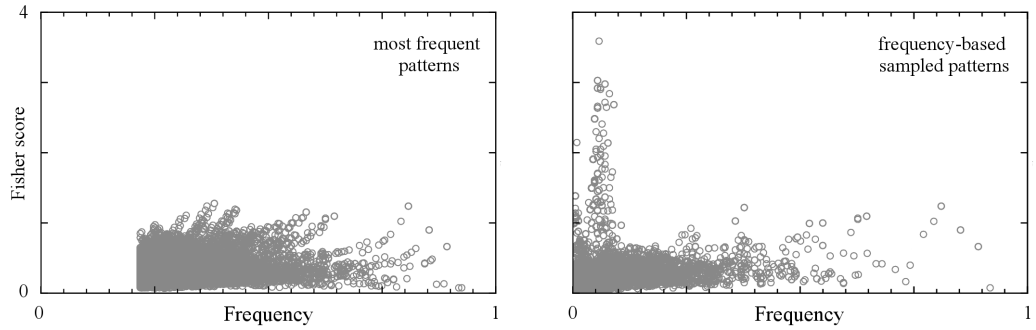


Figure 2.12.: Pattern collections generated from the *primary-tumor* dataset by top-k closed frequent pattern listing (k = 10000) and frequency-based sampling. The drawn patterns are plotted by frequency (x-axis) against Fisher score (y-axis).

The left graphic of Figure 2.12 shows the most frequent patterns, drawn by the LCM algorithm (Uno et al., 2004). By design, the patterns with a lower frequency are drawn last by this type of algorithm. For the above example, this leads to the issue that the

patterns with an extraordinary high Fisher score are not found at all, due to their low frequency. However, using a frequency-based sampling approach (right image of Figure 2.12) results in a pattern collection with an emphasis on frequency, but with a larger diversity among the drawn patterns. This naturally lists also some of the high Fisher score patterns. The example constitutes exactly the motivational case from Section 2.4.1, where the measure of interest diverges from the listing strategy, which in terms can lead to discovered patterns of poor quality. Note that this is not bound to the frequency and Fisher score, as listing a pattern space systematically always bears the danger of ruling out a sub-space of patterns that might be of high interest to the analyst. The exemplary case here is especially severe, as there are far more infrequent than frequent patterns and the time of execution for a frequent pattern mining algorithm grows rapidly with a shrinking support threshold. This is depicted in Figure 2.13 which shows the runtime behaviour of LCM, depending on the support threshold for the datasets from Table 2.6.

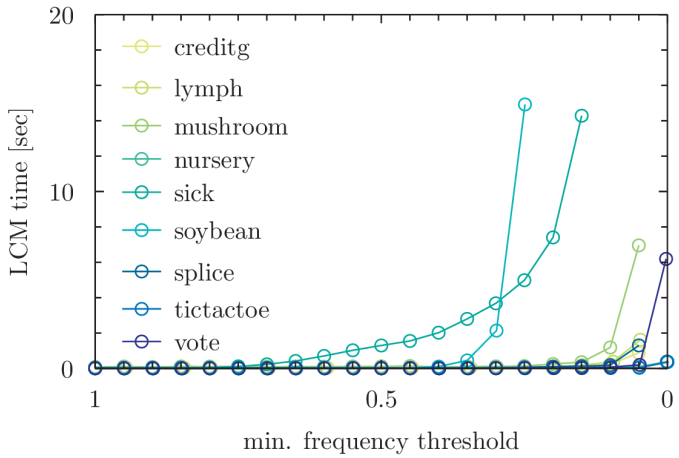
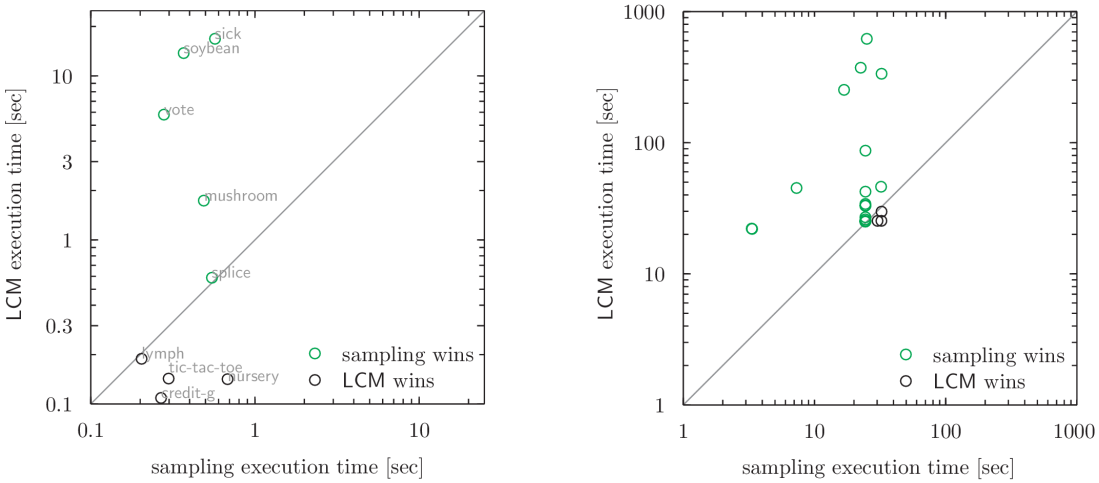


Figure 2.13.: As there are usually far more infrequent than frequent patterns, the execution time of a frequent pattern listing algorithm (in this case LCM) increases rapidly with a lowering support threshold.

When exploring and understanding a dataset via pattern mining, the analyst is usually not limited to finding a pattern collection that strictly follows a given measure of interest. Here, a collection that exhibits a more general coverage of the pattern space with a bias towards a certain measure of interest can be very attractive. In order to be practically feasible, the sampling procedures also have to be competitive to state-of-the-art algorithms in terms of execution speed. Figure 2.14 compares the execution time of frequency-based sampling to that of LCM for the task of drawing an identical number of patterns. For better visibility, the result is split in two pictures, one for medium sized and one for large scale datasets. Note that both figures use logarithmic axes.



(a) Medium sized datasets from Table 2.6

(b) Larger datasets from the FIMI workshop

Figure 2.14.: Time that the LCM algorithm takes versus time of frequency-based sampling to draw an identical number of patterns.

For the experiment, first the LCM algorithm was employed and the support threshold parameter was decreased from 100% in 5% steps, until either all patterns were mined or the execution time exceeded 20 seconds. Then, the same amount of patterns were sampled, using the frequency-based sampling method, and the execution times are compared. The left part of Figure 2.14 shows the results for the nine datasets from Table 2.6. Note that the sampling methods perform better for the larger datasets. The right figure displays the results for two much larger datasets from the FIMI workshop, the 1 GB sized *webdocs* dataset and a 500 MB sized random dataset for all 5%-support thresholds up to an execution time of 500 seconds. This direct execution time comparison against the highly tweaked LCM implementation constitutes a hard setting. First, because within the group of exhaustive pattern mining methods frequent set mining algorithms usually produce the largest output per time unit and second, because the employed LCM implementation is known to be among the fastest of them (winner of the FIMI contest (Bayardo et al., 2004)).

2.5. Summary and Discussion

The chapter started with a formal introduction of notations and definitions to the area of pattern mining. The terms pattern and subgroup were introduced, as well, as different ways to measure the interestingness of patterns. In addition, basic strategies for listing such interesting patterns were presented in combination with commonly applied techniques to speed up the mining process and avoid redundancy.

In general pattern mining techniques have to deal with the problem of pattern redundancy. When using pattern mining as a tool for data exploration, this becomes a key aspect, as it is infeasible for a human analyst to study large sets of potentially insightful patterns over and over again, without loosing focus. For this reason, state-of-the-art algorithms aim to find a compact result set of highly interesting and diverse patterns. A first step towards this is the standard method of considering only the top-k closed patterns. In case the data records are assigned with binary labels, an additional criterion can be applied that condenses the result set even more; the theory of relevance. It reduces the amount of redundancy among the mined patterns by reporting only patterns to the analyst that are considered relevant by the theory of relevance. Using the property that relevant patterns have to be closed on the positively labeled instances, made it possible to develop an algorithm for listing them efficiently. A theoretical and empirical comparison of the runtime and memory complexity with other pattern mining algorithms shows that mining the relevant patterns is competitive with state-of-the-art algorithms. This makes the relevant patterns a good candidate to explore a dataset by pattern mining methods, as for the same time and memory consumption, a more diverse and more condensed result set is compiled.

To focus further on avoiding redundancy, a modified notion of relevance was considered. The Δ-relevant patterns are a generalization of the relevant patterns that allow a dominating pattern to support a controlled amount of additional negatively labeled instances. Note that this new formulation of Δ-relevance is not transitive any more. The idea of reporting only a single representative for a group of similar relevant patterns and discarding the others comes with the benefit that other, possibly more diverse, patterns can now be shown to the analyst. In this chapter, an efficient algorithm for finding these Δ-relevant patterns was introduced. The according experiments indicated a clear reduction of the redundancy among the reported patterns. In addition, for varying values of Δ, the top-20 patterns were utilized to predict the label of a data record. It could be observed, that a slight introduction of slack also increased the predictive performance of the result set. This initial improvement of predictive performance was interpreted as an indicator for an overall better quality of the patters that are finally shown the analyst.

A third part of this chapter covered a technique that enables the analyst to draw pattern samples proportional to different probability distributions. A two step sampling algorithm was introduced that draws random patterns with the probability proportional to their frequency of occurrence. The key idea for frequency-based sampling is that random experiments are good at reproducing frequent events. By adjusting each of the two steps within the algorithm, a whole family of pattern sampling algorithms is characterized. This way, the sampling procedure can, for instance, be biased to prefer patterns that are frequent on the positively labeled data records, but infrequent on the negatively labeled ones. Note that for this section the focus is not on avoiding redundancy among the reported patterns. As the samples are independent of each other, the set of sampled patterns naturally exhibits more diversity in comparison to enumerative pattern mining approaches. In an exploratory data analysis scenario this can be beneficial. As the

analyst might not be willing, or able to wait for the algorithm to evaluate all possible pattern combinations, the most interesting among the first found patterns are the ones that the analyst has to work with. If, however, the algorithm follows a bad heuristic in its search for the interesting patterns, the preliminary results can be poor. A very sever example of this was presented in Figure 2.12. This issue is naturally remedied by employing sampling based methods. Even with an unfavorable sampling bias, the diversity among the sampled patterns still produces some patterns that are of actual interest to the analyst. An additional experimental evaluation of the frequency-based pattern sampling method showed that the patterns can be produced very quickly in terms of "patterns per second", which makes this technique a viable tool for real life data exploration tasks. In the later Section 4, we will see a concrete example of how a larger sampled pattern set can be used to explore a data collection.

While exploring a dataset by using the above proposed pattern mining methods can already give a great amount of insight to the analyst, it is not very interactive. Recall that data mining is a process, where an analyst iteratively explores and deduces aspects of a dataset. However, when applying pattern mining techniques in the classic way, the analyst has to change parameters and re-calculate everything in order to dig deeper into the data. For the above introduced methods, this usually means changing the interestingness measure, the Δ parameter, the k threshold, or selecting another attribute as label. In the next section, we will introduce an alternative approach towards exploratory data mining that does not deliver a pre-selected collection of phenomena to the analyst. This approach focuses on presenting the data collection to the analyst as a whole and lets him intuitively interact with it to discover and study interesting aspects.

3. Interactive Embeddings

In the last chapter we have seen how pattern mining can be used to explore and understand a dataset by guiding the analyst's attention towards interesting phenomena within the data. It is an approach that finds partitionings of the dataset which are outstanding according to some measure of interest. Using different pattern mining techniques, interestingness measures and target-attributes, makes this approach a valuable tool in an exploratory data analysis setting.

Apart from these methods, there also exist techniques that do not directly guide the analyst, but rather try to give a global overview on the distribution of whole data and the relations among its records. This helps the analyst to understand and study the underlying structure of the data and discover aspects that might not have been found by strictly following a quality function. A common way to provide this overview is to project the data into a lower dimensional space (usually two, or three dimensions) and visualize it in a scatter plot. The projected data is also referred to as **embedding**, or embedded into the lower dimensional space. Classically, different embedding techniques with different objectives and parameter settings are used to give the analyst a birds-eye view on interesting projections and aspects of the data. However, switching between these different projections may distract the analyst and cause him to loose focus, as the transitions between different embeddings are usually not obvious. To counteract this, the following chapter introduces methods that enable the analyst to interact directly with the visualization of the embedded data.

Unlike in pattern mining, the data exploration process by studying interactive embeddings is not lead by the algorithm, but rather becomes an interaction between the analyst and the visualization of the data. To do so, the analyst can actively steer the perspective of a two dimensional projection of the data. Altering the perspective

and seeing how related data records move together, zooming and filtering the data collection and inspecting structures within the embedding closer, can help the analyst to understand the underlying structure of the data and formulate hypotheses. One way to provide an interface for the interaction is to let the analyst select data points as **control points** and relocate them within the embedding (see e.g. Paurat and Gärtner (2013), Endert et al. (2011), Brown et al. (2012) and Iwata et al. (2013)). Altering the positions of these control points triggers the utilized embedding technique to recalculate the whole projection, subject to the updated control-point locations. However, the just cited methods are primarily meant to implement a good user interface for tweaking the parameters of an algorithm and none of them is especially designed for the interactive visual exploration of the data's underlying structure.

The contributions of this chapter are tightly coupled to investigating the idea of actively altering a two dimensional projection of high dimensional data in an intuitive way. Following the idea of moving individual data points within the projection to new locations in order to steer the overall embedding of the data, three different algorithms are presented that utilize this interaction technique: i) the least squared error projection algorithm, which solely considers the control points' data and embedding locations to calculate a linear projection with the least squared residual error. ii) a probabilistic approach on embedding data which considers a prior belief about the embedding and the placement of the control points[1] and iii) an interactive version of a kernel PCA that can take several types of constraints into account. In addition, a tool for interactive visualization (InVis) was developed over the course of this thesis, which implements all of the above mentioned techniques in a single program with a unified graphical user interface. Note that all of these algorithms put also emphasis on performance, as immediate visual feedback (i.e. seeing the embedding change in a life-updating manner) can help the analyst to understand common relations among the data records.

3.1. Preliminaries

The following section gives a quick introduction to the used notations, reviews some of the classical static embedding methods and introduces three embedding techniques that enable the analyst to directly interact with the projected data.

3.1.1. The Data

In this chapter the algorithms use a different representation of the data than a transactional database, which is classically used in pattern discovery. Here we consider a dataset $\mathcal{D}$ to consist of n data records, $d_1, \ldots, d_n$ over D attributes, with each data record being an instance of $\mathbb{R}^D$. Each dimension of the vector represents an attribute, so that the data ultimately can be arranged into an $n \times D$ matrix. Table 3.1 shows

[1] It is also shown that for a very basic parameter choice this most likely embedding is equivalent to the least squared error projection algorithm.

the five example cocktails, earlier presented as itemsets in Table 2.1, in their vector representation form, aggregated into a matrix. The values indicate the share of each ingredient within the cocktail, such that e.g. the cocktail *Mojito* is now represented by the vector $(0,\ 0,\ 0.46,\ 0.3,\ 0.05,\ 0,\ 0,\ 0.15,\ 0.04,\ 0,\ 0,\ 0,\ \ldots,\ 0)$.

Cocktail names	**Ingredients**	Coconut milk	Grenadine	Light rum	Lime	Mint	Orange juice	Pineapple	Soda water	Sugar	Tequila	Vodka	...
Mojito	(	0,	0,	0.46,	0.3,	0.05,	0,	0,	0.15,	0.04,	0,	0,	0,)
Piña Colada	(	0.21,	0,	0.58,	0,	0,	0,	0.21,	0,	0,	0,	0,	0,)
Screwdriver	(	0,	0,	0,	0,	0,	0.33,	0,	0,	0,	0,	0.67,	0,)
Tequila Sunrise	(	0,	0.01,	0,	0,	0,	0.25,	0,	0,	0,	0.74,	0,	0,)
⋮													

Table 3.1.: A collection of four well known cocktails, represented as vectors. The values indicate the share an ingredient has to the cocktail.

Note that it is always possible to transform a transactional dataset into a numerical one, e.g. by mapping the categorical attributes to real numbered values. However, it should be done with care. While values like *low*, *medium* and *high* imply a logical order and can be mapped to e.g. 1, 2 and 3, a problem arises if the categorical values are not in an ordered relation. In this case it makes more sense to split the original attribute into new binary valued attributes, one for each occurring category. This way the values do not imply an order.

3.1.2. Embedding Data Into Lower Dimensional Spaces

When talking about embeddings in this work, the idea is that data records are mapped from their original D dimensional instance space $\mathcal{A}$ into another space $\mathcal{B}$. While in theory $\mathcal{B}$ can be of arbitrary dimensionality, the here considered embeddings are usually, for the sake of a scatter plot visualization, two dimensional. Because dimensionality reduction techniques embed the data in a meaningful way into a lower-dimensional space, they are often utilized for the visualization of data. The importance of these techniques for visualization purposes in data science is not new and has already been stated by Tukey (1975). In the following, before diving into interactive embedding techniques, let us quickly review some of the most known classical (and mostly static) embedding techniques.

Linear Embedding Techniques

Probably the best known dimensionality reduction technique is **Principal Component Analysis** (PCA) (for the original publication see Pearson (1901), for introductions to PCA in standard literature see e.g. Jolliffe (1986) and Hastie et al. (2001)). PCA aims

at finding a set of orthogonal axes that explains the most variance within the data. A common way to find these axes is to perform a spectral decomposition of the covariance matrix of the centered data. The following calculation derives this technique.

Consider X to be a set of n D-dimensional random variables $x_1, \ldots, x_n \in \mathbb{R}^D$. Further, let u be a D-dimensional vector of unit length. It is important that the length of u is constant, as the optimization problem otherwise becomes unbounded. To find the first principal direction of the data, we are looking for a vector u^* that maximizes the variance of $x_1, \ldots, x_n$, projected onto it:

$$u^* = \underset{u:\ \|u\|=1}{\operatorname{argmax}}\ \operatorname{var}(uX^\top)$$

With the variance of a random variable W defined as

$$\operatorname{var}(W) = \mathbb{E}[(W - \mathbb{E}(W))^2]$$

and considering the mean as the empirical estimate of the expected value, we retrieve the following equation:

$$u^* = \underset{u:\ \|u\|=1}{\operatorname{argmax}}\ \mathbb{E}[(\ uX^\top - \mathbb{E}(uX^\top))^2]$$

To simplify the calculation we can assume without loss of generality that X is centered around its mean, which leads us to:

$$\begin{aligned} u^* &= \underset{u:\ \|u\|=1}{\operatorname{argmax}}\ \mathbb{E}[(\ uX^\top - \underbrace{\mathbb{E}(uX^\top)}_{=0}\ )^2] \\ &= \underset{u:\ \|u\|=1}{\operatorname{argmax}}\ \mathbb{E}[(uX^\top)^2] \\ &= \underset{u:\ \|u\|=1}{\operatorname{argmax}}\ \frac{1}{n} \cdot uX^\top X u^\top \end{aligned}$$

Note that for a collection C of centered D-dimensional data records $\frac{1}{n} \cdot X^\top X$ can be considered the covariance matrix Σ of C, as

$$\Sigma = \mathbb{E}[(C - \underbrace{\mathbb{E}(C)}_{=0})^\top \cdot (C - \underbrace{\mathbb{E}(C)}_{=0})] = \mathbb{E}[C^\top C] = \frac{1}{n} \cdot C^\top C.$$

Because the eigenvector of a matrix that corresponds to the largest eigenvalue, signifies the largest stretch of the linear transformation performed by the matrix, we can follow that the particular u^* which maximizes $u^\top \Sigma u$ has to be the eigenvector of Σ with the largest eigenvalue. Projecting X into a space that is orthogonal to this first principal direction and repeating the process yields the second principal direction, which exhibits the largest variance of X orthogonal to the first principal direction. This process can be applied iteratively in order to find the first k principal directions of the data. Later on in Section 3.4 a kernelized interactive extention of PCA will be introduced.

Projection Pursuit (PP) techniques are linear embedding methods that were originally proposed by Kruskal (1969, 1972) with the goal of finding and visualizing interesting projections of high dimensional data. The first publicly accessible implementation of PP was introduced in a publication of Friedman and Tukey (1974), which also coined the name *Projection Pursuit*. The key idea behind PP is to project the data into a two or three dimensional space such that the resulting embedding deviates heavily from a normal distribution and reveals a large amount of structure. To do so, projection pursuit calculates a measure (in the literature referred to as *index*) to each projection axis of the data which tries to capture the amount of structure. For each projection axis, this index is greedily optimized to steer the projection towards more interesting angles.

In the version, proposed by Friedman and Tukey, the function to calculate the index is the product of two factors. One factor captures the global spread (they used the trimmed standard deviation), while the other one measures local density of the data (here the sum of the pairwise distances within an ϵ-neighborhood for each data point was used). Utilizing this index-function usually results in an embedding which shows cluster structures that are well separated. Huber (1985) gives a good survey on PP methods, their applications, benefits and downsides and different objective functions which drive the heuristic search for an interesting projection.

Figure 3.1 below shows three different linear projections of the cocktail dataset, all derived by PP in different stages of optimizing the index value. The tool GGobi (see `http://www.ggobi.org`) generated the figures and provided the implementation of the PP algorithm. The left figure shows an initial random projection of the data that looks roughly normal distributed and exhibits a low corresponding index value. As PP has not started to optimize the index value, the embedding is not biased towards showing structure. The middle and the right figure depict the stages of PP during optimizing the index value and after convergence.

Figure 3.1.: Three stages of projection pursuit, visualizing the cocktail dataset. Depicted are the projection of the data before, during and after optimizing the index value.

Non-Linear Embedding Techniques

For the non-linear embedding methods, there are two major directions. One tries to preserve the global structure of the data in the embedding, while the other tries to preserve the local structure. The following section introduces a few of these methods.

It is by no means a comprehensive survey, but rather tries to indicate the range of the different approaches and how well the field is studied.

Of the global structure preserving techniques, **Multi Dimensional Scaling** (MDS) techniques are probably the most known ones. With their origin dating back to the mid sixties (Kruskal, 1964; Torgerson, 1965) and still being actively used (Cox and Cox, 2000), the general idea of MDS is to find a lower dimensional embedding of the data such that the pairwise (Euclidean) distances resemble the pairwise distances, measured in the original high dimensional space. More formal, let Δ and δ be the pairwise distance matrices of the data, in the original high dimensional space and the lower dimensional embedding space. Further let $||A||_F$ denote the Froebenius norm of the matrix A, given by $||A||_F = \sqrt{\sum_{a_{ij} \in A} (a_{ij})^2}$. The task of MDS is to find an embedding of the data that minimizes $||\Delta - \delta||_F^2$. There are two major methods to find such an embedding. One method iteratively refines an initial embedding, minimizing a loss function, in the context of MDS referred to as **stress**. The other method performs a spectral decomposition of the centred matrix of all pairwise squared-distances. The coordinates of the k-dimensional embedding can now be found in the k most significant eigen-vectors. Note that only the first of these methods results in a non-linear embedding of the data.

A very natural extension, introduced by Tenenbaum et al. (2000), to the classic MDS algorithm is **Isometric mapping** (Isomap). It assumes that the data lies on a lower dimensional manifold, which is embedded into the higher dimensional space. To account for that, the distances are measured along this manifold and not in Euclidean space. As an example for a manifold think of a dataset where the data records all lie on the hull of a sphere, as depicted in the following Figure 3.2.

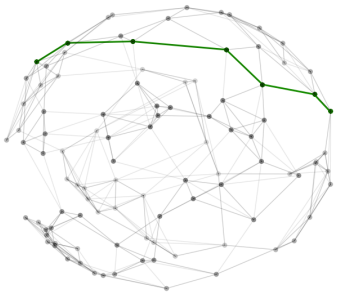


Figure 3.2.: Approximating the shortest path between two points on the hull of a sphere. The estimated geodesic distance between two points is the sum of the Euclidean distances along the shortest path, connecting them on the k-nearest-neighbor-graph.

Although the data "lives" in a three dimensional space, the position of each record is constrained to lie on a curved plane (the manifold), which gives the data an intrinsic dimensionality of two. The geodesic distance can be approximated, by accumulating the Euclidean distances of the shortest paths on the k-nearest-neighbor-graph of the data. Figure 3.2 illustrates this for a dataset where all data records lie on the hull of a sphere.

Once all pairwise distances on the manifold are estimated and composed into a matrix, a lower dimensional embedding of the data that reflects these distances can then be retrieved via classical MDS.
Another structure preserving technique is the originally by Schölkopf et al. (1997) introduced **Kernel Principal Component Analysis** (KPCA). It is an extension of classic PCA which uses kernel methods to perform the linear operations of PCA with a non-linear mapping. The idea behind this will be introduced more thoroughly in Section 3.4.1. Note that, depending on the choice of the kernel, this technique can be used to either preserve the global, or the local structure of the data.

Locality preserving methods

Some, usually non-linear, embedding techniques try to preserve the local structure of the high dimensional data in the embedding. Common to all these techniques is the assumption that the data lies on a lower dimensional manifold that exists within the high dimensional space in which the data is represented.

An example for such a locality preserving method is the **Locally Linear Embedding** method (LLE). The algorithm tries to flatten the manifold by aligning the local neighborhoods. To do so, every data record is expressed as a linear combination of its k-nearest neighbors. The algorithm then constructs a lower dimensional embedding, with each data point being described by the same linear combination of its neighbors.

Local Tangent Space Alignment, which was introduced by Zhang and Zha (2004), also follows the thought of flattening the manifold. The key idea behind this method is that if unfolded correctly, all hyperplanes that are tangent to the manifold should align. To approximate the tangent hyperplanes, the algorithm finds the d first principal components of the k-nearest neighbors for each point (here d is the dimensionality of the embedding space).

Another non-linear technique that preserves the local neighborhood is the **Laplacean Eigenmap** embedding (Belkin and Niyogi, 2003). Similar to a PCA it performs a spectral decomposition and embeds the data via the first d eigenvectors. The key difference is, that that the decomposition is not done on the covariance matrix, but instead on the Laplacean matrix of the k-nearest neighbor graph of the data.

Other Embedding Techniques

There are numerous other embedding techniques and the here presented list of algorithms that can be used to embed data into a new space is by no means comprehensive. Some techniques try to find a small set of outstanding representatives of the data and express the rest as a linear, or convex combination of them. Examples for this are **Simplex Volume Maximization** (Thurau et al., 2010) and **Archetypal Analysis** (Cutler and Breiman, 1994). However, most techniques, including already mentioned ones, like PCA,

MDS, Isomap and Laplacean Eigenmaps, use matrix factorization techniques to embed the data. Here a data matrix X is factorized into a set of matrices that, when multiplied together, generate or approximate X again. The resulting factor matrices can then be used generate a lower dimensional embedding of the data. For instance, the **CUR** matrix factorization technique (Drineas et al., 2006) tries to find a set of columns C and rows R from the data matrix and an additional diagonal matrix U, such that $CUR \approx X$. For data that has only positive entries, the **Non-negative Matrix Factorization** (Lee and Seung, 1999) technique can be of use. Here, the data matrix is factorized into two matrices W and H, which contain only positive entries. Depending on the case, this can be helpful when trying to interpret the results. Lastly, an excellent survey on different embedding techniques and how they relate to each other can be found in *"Dimensionality reduction: A comparative review"*, published by van der Maaten et al. (2009).

3.1.3. Interacting with Embeddings

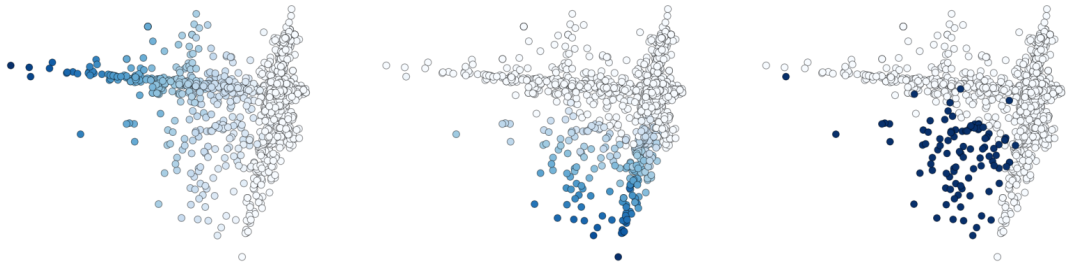
There are many possibilities that enable an analyst to interact with the visualization of embedded data. One already mentioned way of interaction is to actively steer the projection of the data into the lower dimensional embedding space. The Sections 3.2, 3.4 and 3.3 introduce different methods that empower an analyst to directly steer the projection. On the other hand, the analyst does not necessarily have to alter the projection in order to interact with the embedding. There are numerous ways to explore a static embedding, e.g. by color coding properties of the data records, or by annotating them. In the following, both ways of interaction are discussed in more detail and according interaction methods are introduced.

Examining Static Structures

A great guide line to exploring data was formulated by Ben Shneiderman in his famous visualization mantra "*Overview first, then zoom and filter, details on demand*" (Shneiderman, 1996). Consider the overview on the data given by the rendered embedding. The analyst can observe all data records scattered in the plane and hopefully spot interesting structures and patterns that emerge. The question arises how to examine such structures on a larger scale, which constitutes the "zoom and filter" step. To this end, several techniques come to mind.

One way to filter and guide the focus of the analyst, is to simply highlight a property of the data records within the embedding. This can be e.g. be done by using colors. In the following Figure 3.3, an embedding of the cocktail data into its first two principal directions is depicted. The images(a) and (b) highlight the two most frequently appearing ingredients *Vodka* and *Orange juice* by color. Here, the intensity of the color is proportional to the value of the attribute and reveals that not only the cocktails containing *Vodka* are to the left side of the embedding, but also that there is an increase in the amount of *Vodka* the further left a point is embedded. Image (c) on the right of the figure highlights all cocktails that simultaneously contain *Vodka* and *Orange juice*. This

image illustrates two things. First, the color does not necessarily have to indicate the value. It can also simply encode whether a data record possesses a certain property, or not. Second, not only attributes that are directly part of the dataset can be highlighted. Here, the simultaneous presence of two attributes is highlighted, but it could well be a more complicated property, like e.g. the numbers of neighbors within an ϵ-neighborhood in order to emphasize dense regions.



(a) (b) (c)

Figure 3.3.: The figure shows the cocktail dataset projected onto the first two principal directions. The first two images tint the cocktails proportional to their share of *Vodka* (a) and *Orange juice* (b) with a blue intensity. The right image (c) emphasizes the set of cocktails that contain *Vodka* and *Orange juice* simultaneously.

Of course there are other ways to highlight and emphasize data records within the embedding; widely used are also the point's opacities or their sizes. In general any graphical property of the points can be used. The Figure 3.4 below conveys the same information, as Figure 3.3 (a), but utilizes the point's transparencies and sizes to highlight the share of *Vodka*.

(a) (b)

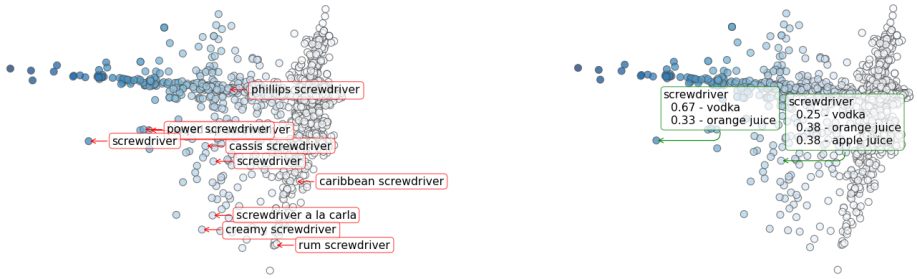
Figure 3.4.: Highlighting can also be done via transparency (a), or the size of the points (b). Again the share of *Vodka* is indicated by the highlighting method. Note that in these illustrations the points do have a minimum transparency/size, such that cocktails containing no *Vodka* at all are still displayed.

A conceptually different approach to interact with an embedding is to convey information about a region of interest within the embedded data that has been selected by the analyst. These information can e.g. be the output of standard knowledge discovery methods, like listing the most frequent or relevant patterns (in case a label is present). The first image of Figure 3.5 illustrates this concept. Depicted are the five most frequent patterns for the data records of the region that hosts the cocktails containing *Vodka* and *Orange juice* at the same time. Using this method, it becomes clear that the two ingredients are dominant here, but also other fruity and juicy components show their presence. To further investigate a structure, or region of interest, the analyst can now filter out all data records which do not belong to the marked region. Re-embedding the remaining sub-selection of data records often shows a structure of its own, which again can be investigated deeper. The second image of Figure 3.5 illustrates this and shows a PCA embedding of only the cocktails selected in the first image. The new embedding often directly raises new questions about structure, or symmetries that can be found and invite the analyst to explore it deeper.

(a) (b)

Figure 3.5.: (a) Selecting a subset of the data records and querying a summary. This particular selection holds the cocktails containing *Vodka* and *Orange juice*. (b) Re-embedding only the selected cocktails in a new PCA embedding shows that the sub-selection has a structure of its own.

Remembering Shneiderman's visualization mantra, the last part of it suggested to provide *details on demand*. Although not complex to realize, being able to retrieve detailed information on individual data records is certainly a "must have" for any interactive visualization. Figure 3.6 presents two different ways to fulfill such requests; first, by enabling the analyst to search for data records by their Id or name and second by allowing him to inspect the attribute values of a single data record.



(a) (b)

Figure 3.6.: (a) Searching for all cocktails with the word "screwdriver" as part of their name. (b) Querying the data behind two of those *Screwdriver* cocktails reveals why they were projected to different locations.

Altering the Projection

In addition to interacting with an embedding via highlighting, filtering and querying detailed information, there is also the possibility to alter the projection itself. The idea behind all here presented methods to interact with the visualization and shape the embedding, is to incorporate domain knowledge by providing feedback to the underlying embedding algorithm on individual data records. In this work, the utilized approach is to interact with the embedding via the *placement* of **control points**. The idea behind control points is that the analyst may select a data record within the visualized projection and relocate it to a new position. Moving a control point to a new location is interpreted as feedback by the underlying embedding algorithm and triggers a re-calculation of the projection, which tries to respect the desired location of the control point. Note that there may exist constellations of control points, where the underlying algorithm is not able satisfy all of the desiderata. How this issue is resolved depends on the underlying technique.

An analogy to steering a projection via control points can be found in observing the shadow of a point cloud, for instance, as depicted in Figure 3.7 a cup.[2] The cup can be oriented in such a way that certain parts of the shadow fall onto desired locations. Demanding new positions for these desired locations calls for a re-orientation of the cup. This way the projection can actively be steered. Figure 3.7 illustrates this analogy by using three control points as desired projection locations. In the following Sections 3.2, 3.3 and 3.4, three different algorithms are introduced that make use of control point placement.
In general, there are also other methods to actively provide feedback on individual data records to an embedding algorithm and this way alter the projection. Two commonly used methods, which are not discussed in more detail in this work, are the usage of **must-link/cannot-link** constraints between pairs of data records, and the idea of taking label

[2] The data records are spacial coordinates, retrieved from laser-scanning an IKEA cup. For the dataset see Neumann et al. (2012).

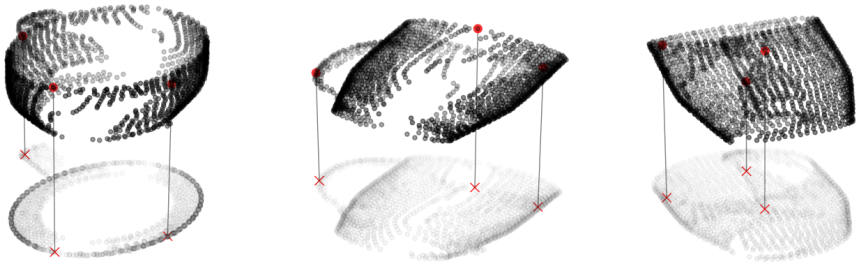


Figure 3.7.: The constellation of the control points, marked by red x's, and the angle of the projection are dependent. Demanding specified locations for the control points induces a certain projection.

information on individual data records into account, when calculating the embedding. For must- and cannot-link constraints, the analyst specifies pairwise similarity examples; for the label incorporation, the analyst simply labels individual data records within the visualization. The underlying embedding algorithm incorporates this kind of feedback, by projecting data records that must-link, or are of the same label, in proximity to each other and the data records that cannot-link, or are of different label, apart from each other.

3.2. Least Squared Error Projection

Least Squared Error Projection (LSP) is a very simple, yet effective, method to steer an embedding via control points. It enables the user to layout a two dimensional embedding of a possibly higher dimensional dataset by selecting some of the embedded data points as control points and placing them to a desired location. LSP then calculates the linear transformation with the least accumulated squared error in the residuals between the desired and the projected location of the control points. Once this transformation is found, it is applied to all data records in order to retrieve the full embedding.

3.2.1. Concept and Computation of LSP

Consider a dataset X with n data records $x_1, ..., x_n$ from an instance space $\mathcal{X} \subseteq \mathbb{R}^D$ and the general task to map $\{x_1, ..., x_n\}$ into an embedding space $\mathcal{Y} \subseteq \mathbb{R}^2$, yielding $\{y_1, ...y_n\}$. To determine this mapping, the user chooses a set of m data records from X, denoted by X_m, and fixes their coordinates in the embedding space, providing Y_m. It can be assumed without loss of generality that the first m records of X are those control points. Given X_m and Y_m, LSP calculates a linear transformation with the smallest accumulated squared error in the residuals between the m desired and the m actual control point locations. Regarding X_m and Y_m as data matrices of shape $m \times D$ and $m \times 2$, we are looking for a projection matrix $P \in \mathbb{R}^{D \times 2} : \mathcal{X} \to \mathcal{Y}$ that solves the equation $PX_m^\intercal \approx Y_m$.

This is a system of linear equations that can be solved efficiently. Starting from $PX_m^\top = Y_m$, we augment the equation by right-multiplying on both sides first X_m and then second the inverse of $X_m^\top X_m$, yielding:

$$PX_m^\top = Y_m$$

$$PX_m^\top X_m = Y_m X_m$$

and

$$P(X_m^\top X_m)(X_m^\top X_m)^{-1} = Y_m X_m (X_m^\top X_m)^{-1}$$

Together with $(X_m^\top X_m)(X_m^\top X_m)^{-1}$ being the $m \times m$ identity matrix and $X_m(X_m^\top X_m)^{-1}$ being the pseudo inverse of X_m, denoted by X_m^+, we finally retrieve:

$$P = Y_m \underbrace{X_m (X_m^\top X_m)^{-1}}_{X_m^+} = Y_m X_m^+ \tag{3.1}$$

Note that if there is no inverse to a matrix, the pseudo inverse is known to solve the underlying system of linear equations with a minimum squared residual error. This computation of P can be solved efficiently, especially since it only depends on the data and embedding positions of the m control points and not all n data points. Once the projection matrix is found, the final embedding of all n data records can be retrieved by applying P to all data records. This can easily be done via the matrix multiplication $PX^\top = Y$.

When interacting with the embedding and altering the locations of the control points, P has to be recalculated. As long as the control points stay the same and only their locations within the embedding change, P can be derived by multiplying the matrices $Y_m X_m^+$ with a time complexity of $\mathcal{O}(mD)$, where only Y_m changes. However, if the user alters the selection of the control points, e.g. by adding a new one, the pseudo inverse X_m^+ has to be recalculated, which leads to an additional calculation with a time complexity of $\mathcal{O}(mD^2 + m^2D + m^3)$ (see Section 3.2.3).

3.2.2. Direct Interaction with the Embedding

In practice, the computation of a single LSP embedding can be performed several hundred times per second. This opens the possibility of a live-updating interactive embedding. In this scenario, the analyst selects a control point within the visualization and drags it to a new location. While dragging the control point, the projection matrix is continuously updated with the intermediate locations of the control point by calculating $Y_m X_m^\dagger$ and the resulting embedding is instantly rendered. Since altering the location of a control point by only a tiny amount, leads to only small changes in the resulting embedding (as is shown in the upcoming Section 3.2.3), the transitions between the individual embeddings are smooth. Being able to steer the angle of projection in such a way and without leaving the visualization lets the user experience the interface as a natural environment. This supports the analysts cognitive workflow and helps to keep the focus,

while studying the data. The direct visual feedback of seeing how the distribution of all data records changes upon interaction enables the analyst to understand the underlying structure of the data and formulate, or test hypotheses.

The following Figure 3.8 shows three intermediate steps of an interaction with an LSP embedding of the cocktail dataset. In this example, four random control points are placed roughly at opposing locations to generate the embedding. One can see how dragging the *Artillery* control point results in a smooth transition of the sequence of embeddings. As most cocktails in this dataset do not share any ingredient with the *Artillery* cocktail, their locations are not influenced by the interaction. On the other hand, cocktails which do share ingredients with this control point, are influenced. To illustrate this in the below figure, as the *Artillery* cocktail contains a significant amount of *Gin* (86%), the share of this ingredient within each cocktail is color coded in blue. One can observe that the cocktails containing more *Gin* are influenced stronger by the displacement of the *Artillery* control point, than the ones containing less.

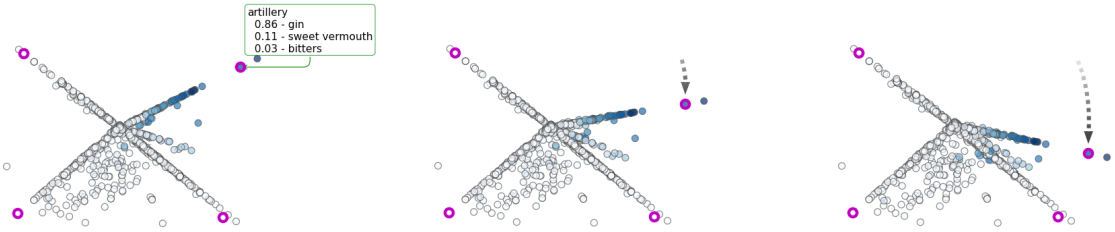


Figure 3.8.: Three intermediate steps of an interaction with the LSP embedded cocktail dataset, using four control points (highlighted in pink). A stronger blue color indicates more presence of *Gin* in a cocktail. One can see how the placement of the *Artillery* control point (which contains 86% *Gin*) influences cocktails that contain more *Gin* stronger.

3.2.3. Evaluation

This section demonstrates three properties of the LSP technique. In *Scalability*, the time complexity is analyzed and it is shown that the underlying algorithm scales well with larger datasets, however, the graphic library that is used to render the visualization doesn't. In *Stability*, it is shown that using LSP as a live-updating embedding technique is intuitive to the analyst, as it behaves in a robust and expected way, when altering the placement of the control points. Finally, in *Flexibility*, it is demonstrated that LSP can illuminate different aspects of the data, by placing a only few control points in an exemplary way.

Scalability

As the least squared error projection method only depends on positions and the original high dimensional data of the control points, the calculation of the projection matrix scales well with the size of the dataset. In practice, the pseudo inverse is usually

calculated by performing a **singular value decomposition** (SVD).[3] The computational complexity of an SVD for an $m \times D$ matrix lies in $\mathcal{O}(mD^2 + m^2D + m^3)$. As the number of control points is usually rather small, the cubic term does not delay the calculation too much, however, for very high dimensional datasets the quadratic term may very well have a significant impact. Also note that while the analyst does not change the selection of the control points, it is not necessary to calculate the pseudo inverse and the projection matrix can be calculated via matrix multiplication in $\mathcal{O}(mD)$.

In practice, though, the bottle neck is not the embedding algorithm, but the graphical library which renders the visualization. To illustrate that, the performance of LSP is benchmarked on 28 datasets from the UCI machine learning repository, the cocktail and the ICDM 2001 abstracts dataset (Kontonasios and Bie, 2010). In the experiments, the LSP embedding is calculated, based on ten uniform at random selected and placed control points. Over the course of ten seconds, it was measured how many embeddings were calculated and how many of these were actually rendered. The Figures 3.9, 3.10 and 3.11 visualize the results of this experiment. The properties of the datasets and the numeric results can be seen in Table 3.2.

Image 3.9 shows the amounts of updates that could be rendered, i.e. drawn, per second, which mainly depends on the number of data records that have to be rendered. The second image, Figure 3.10, illustrates that the embedding algorithm LSP is able to deliver many more updates per second, than the graphical library can handle. Crucial for the amount of embeddings that can be calculated (not rendered) is the size of the dataset. However, even for the dataset with the largest size (in this case *webtender*, aka the cocktail dataset), the algorithm is still able to deliver ~850 updates per second, where as only ~16 of these can be rendered. The speedup between what is calculated and what is rendered, depicted in Figure 3.11, mainly depends on the number of attributes the dataset has. This is not surprising, since for high dimensional datasets, as earlier stated, the computational complexity of LSP is dominated by the number of attributes.

[3] Here, the matrix X is decomposed into $U\Sigma V^\top$, with Σ being a matrix that contains only entries on the diagonal. The pseudo inverse $X^\dagger$ is then given by $V\Sigma^\dagger U^\top$, where $\Sigma^\dagger$ is simply the element-wise reciprocal of Σ, as Σ contains no off-diagonal entries.

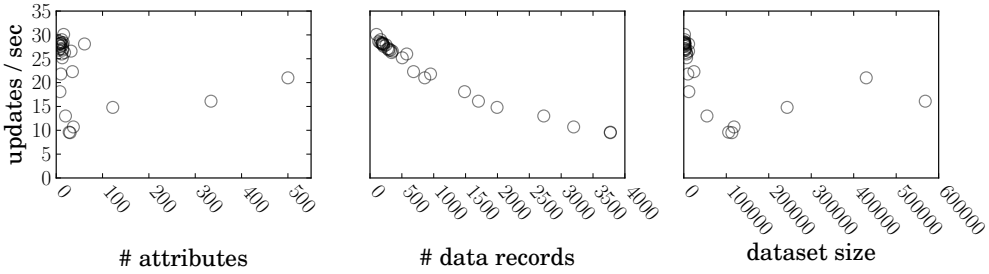


Figure 3.9.: LSP experiments on 30 datasets on the scalability of *rendered* updates. The number of displayed updates per second depends mainly on the number of data records.

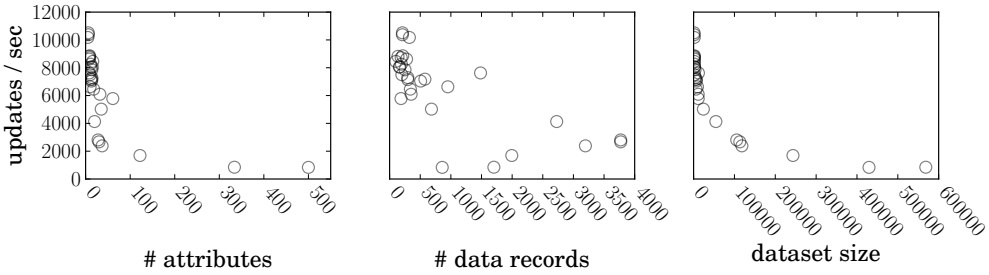


Figure 3.10.: Experiments on the scalability of *calculated* updates. The number of calculated updates depends on the size of the dataset.

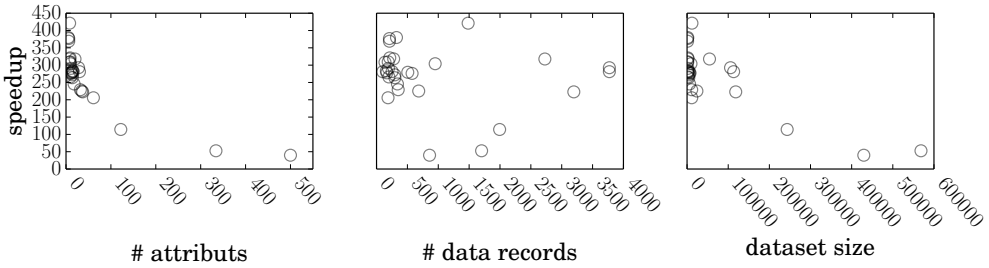


Figure 3.11.: The speedup seems to depend mainly on the number of attributes.

Dataset	Columns	Rows	Size	Rendered updates/sec	Calculated updates/sec	Speedup
autoPrice	15	159	2385	28.4	8052.9	283.5
bodyfat	14	252	3528	27.9	7855.9	281.5
breastTumor	8	277	2216	27.1	8620.4	318.1
cholesterol	14	297	4158	26.8	7290.0	272.0
cleveland	14	297	4158	27.1	7146.0	263.7
communities	122	1994	243268	14.8	1690.0	114.2
cpu	6	209	1254	28.1	10372.0	369.1
galaxy	5	323	1615	26.8	10178.5	379.8
glass	9	214	1926	27.6	8859.0	321.0
housing	13	506	6578	25.2	7041.1	279.4
hypothyroid	28	3772	105616	9.6	2809.0	292.6
ICDM abstracts	500	859	429500	21.0	836.7	39.8
ionosphere	32	351	11232	26.6	6093.3	229.1
kr vs. kp	37	3196	118252	10.7	2384.2	222.9
lowbwt	9	189	1701	28.2	8727.0	309.5
machine cpu	6	209	1254	27.9	10512.4	376.8
movies	20	2727	54540	13.0	4130.7	317.7
primary-tumor	18	339	6102	26.3	6461.4	245.7
pharynx	11	193	2123	28.4	8255.9	290.7
pwLinear	10	200	2000	28.2	7498.5	265.9
sensory	12	576	6912	26.0	7179.1	276.1
servo	12	167	2004	29.0	8056.6	277.8
sick	30	3772	113160	9.5	2671.3	281.2
soybean	35	683	23905	22.3	5023.7	225.3
stock	10	950	9500	21.8	6629.8	304.1
triazines	61	186	11346	28.1	5783.6	205.8
veteran	8	137	1096	28.6	8819.4	308.4
webtender	334	1702	568468	16.1	848.2	52.7
yeast	8	1484	11872	18.1	7624.5	421.2
zoo	16	101	1616	30.1	8456.9	281.0

Table 3.2.: Scalability experiments for the LSP method on 30 datasets. The column *Size* denotes the number of entries in the dataset ($n \times m$). *Speedup* measures the ration between calculated and rendered embeddings.

Stability

In order to achieve a smooth live-updating interactive embedding, there should not be any sudden and unexpected jumps in the visualization upon interaction. The following experiment demonstrates that LSP possesses this property, by showing that altering the location of the control points by only a tiny amount, also only leads to small changes in the resulting embedding.

In the experiment, a dataset is embedded via LSP, based on 5, 10, 15, 20 and 25 uniform at random chosen control points that are placed to the positions of their PCA embedding. This base-embedding is then perturbed, by dis-locating each control point by some offset into a random direction. The offset is a multiple of the median pairwise distance of the embedded data records and the multiplying factor is denoted here as **perturbation**. To measure the **distortion** between the base- and the perturbed embedding, the average displacement of a point between the two embeddings is calculated and scaled by the median pairwise distance of the base-embedding. Figure 3.12 shows this experiment, averaged over 100 runs on all 30 datasets from Table 3.2. Notable are two effects; first, overall there seems to be the tendency that using more control points stabilizes the embedding. However, the distortion, when using five control points, is clearly lower than the distortion of using ten. The explanation for this lies in the fact that the control points are sampled uniform at random and not chosen carefully. As the PCA embedding does not scatter the data records uniformly, but tends to have a dense center, randomly selecting only a few control points increases the chance of drawing them from close to the center. This implies a higher chance, that the resulting embedding does not exhibit much spread, which again disturbs the measurement of distortion. The second effect to notice is that the distortion scales with the intensity of the perturbation. This shows that the embedding is robust towards little changes in the placement of the control points and that the results of the interaction are feasible to provide a live-update that is intuitive to the analyst, as it behaves in an expected way.

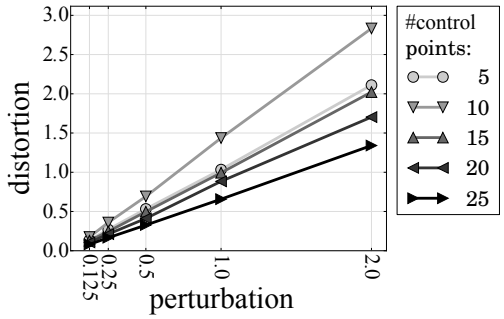


Figure 3.12.: Little perturbation in the control points' placement induces only small distortion of the resulting embedding. One can observe that the usage of more control points stabilizes the projection.

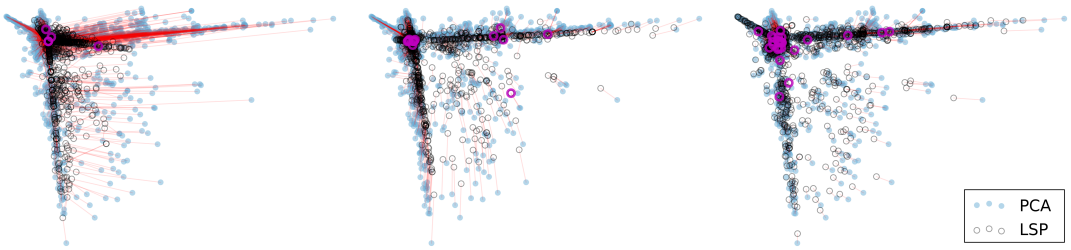
Flexibility

To demonstrate flexibility, a target embedding is mimicked by LSP. With no control points set, LSP projects all data records to the origin of the embedding coordinate system. To approximate the desired target embedding, more and more control points are chosen and placed to the locations, which they possess in the target embedding.

To quantify the similarity of both embeddings, the difference between the target and the approximated embedding is measured by the average **root mean squared error** (rmse) of the displacement of each point, which for two points $x, y \in \mathbb{R}^d$ is calculated by

$$\text{rmse}(x, y) = \sqrt{\frac{1}{d} \sum_{i=1}^{d} (x_i - y_i)^2}.$$

Note that this expression has no upper bound and can be any non-negative number. As target embedding, the projection of the cocktail dataset onto its first two principal components was chosen. For our experiments this means that a smaller rmse indicates a better fit of the approximation, however, rmse values between different experiments are not directly comparable. In the experiment, control points are select uniform at random and placed according to their PCA embedding locations. Figure 3.13 shows three examples of mimicking a target embedding for different amounts of used control points. The black circles show the resulting LSP embedding for placing 5, 15 and 30 (emphasized in pink) control points to their according PCA embedding locations. Blue depicts the targeted PCA embedding and the red lines indicate the residuals between the embeddings. Although for 5 control points the difference to the desired embedding is strong, and thus the rmse is high, one can already see the structure of the intended embedding arising.



(a) (c) (b)

Figure 3.13.: Mimicking a PCA embedding via LSP on the cocktail dataset, using 5 (a), 15 (b) and 30 (c) control points. The blue points in the background show the target PCA embedding. By using more and more control points (depicted in pink), the resulting LSP embedding (black circles) resembles more and more the desired PCA embedding. The displacement of each point is highlighted by a red line.

The following Figure 3.14 shows the development of the rmse for different amounts of control points over 100 runs. Depicted are the median in red and the 75% quantile as a box. The whiskers indicate smallest and largest rmse values over the course of the experiment. As expected, the more control points are used to mimic the PCA embedding, the stronger the residual error drops.

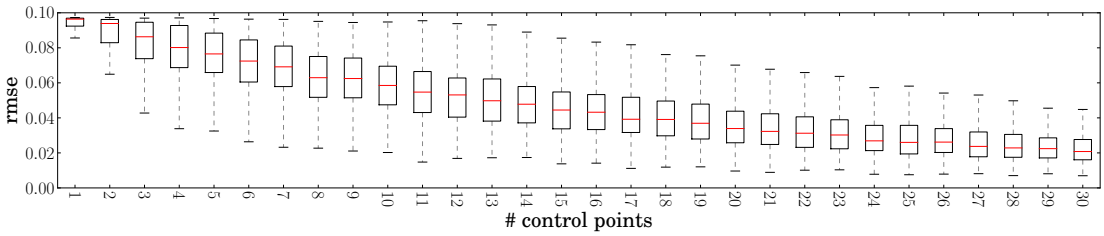


Figure 3.14.: The development of the rmse when approximating the PCA embedding of the cocktail dataset (as shown in Figure 3.13), depending on the number of control points.

However, the error does not solely depend on the amount of control points. It is also coupled to the dimensionality of the dataset. The more dimensions a dataset possesses, the more control points are needed to consolidate the embedding. Figure 3.15 displays the average rmse over 100 runs, depending on the number of utilized control points and the dimensionality of the dataset. Again the cocktail data was chosen and for each run both, control points and the dimensions (attributes) were uniform sampled at random. One can see that higher dimensional datasets need more control points to mimic the PCA embedding well.

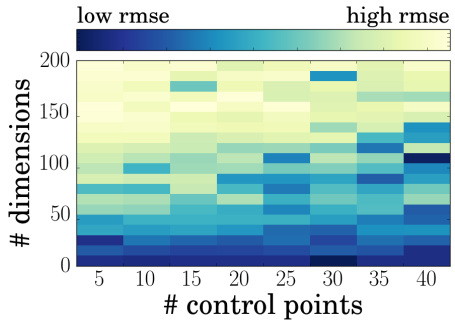


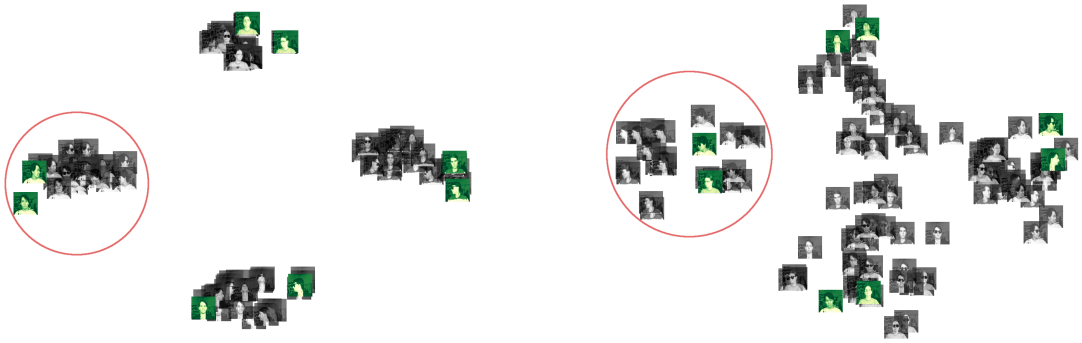
Figure 3.15.: Root mean squared error (averaged over 100 runs) of the approximation of the PCA embedding on the cocktail dataset for different numbers of control points and dimensions of the dataset.

To show that LSP can be useful on high dimensional data, even with only few control points, consider the *CMU Face Images* dataset from the UCI dataset repository (Asuncion and Newman, 2007). The dataset consists of several gray-scale face images of different poses for twenty persons. Each image has a resolution of 32×30 pixels and treating each pixel as an individual dimension, yields a 960 dimensional dataset. In this experiment, all images from four different persons were considered and embedded via PCA and LSP. The following Figure 3.16 shows the images embedded in their first two most significant principal directions.

(a) (b)

Figure 3.16.: The left image (a) shows the complete PCA embedding of the facial images. For better visibility, the right image (b) focusses on the cluster, emphasized by a red circle in image (a). Both figures do not immediately reveal any interesting structure among the facial images.

On the other hand, by utilizing LSP and arranging eight of the image control points, different aspects of the dataset can be emphasized. In Figure 3.17 (a) two images of each person were selected as control points. Already grouping two control points per person in different regions of the embedding space clusters the images by person. On the same data, selecting two examples of each pose as control points and grouping the pictures by these poses reveals a different aspect of the data. In Figure 3.17 (b) one can see that the embedding now highlights the different poses of *looking-up*, *-straight*, *-left* and *-right*. The following Figure 3.18 shows again a zoom into the left most cluster of both embeddings. The resulting projections do not always yield clusters, as in the above figures, but arranging the control points with a concept in mind usually yields a local neighborhood that seemed to reflect it. This exemplifies well how an analyst can incorporate domain knowledge to find and study the emerging structures.



(a) (b)

Figure 3.17.: Different aspects of the same dataset can be shown by selecting and placing the control points (highlighted in green) in a different ways. In image (a) the control points are grouped by persons, while image (b) brings out the poses of *looking-straight*, *-up*, *-left* and *-right*. The red circled areas are displayed separately in the Figure 3.18.

(a) (b)

Figure 3.18.: For better visibility, a zoom into the left clusters of the Figures 3.17 (a) and (b).

To further demonstrate the flexibility of LSP, the experiment of mimicking the embedding into the first two principal components by setting control points accordingly was performed on the 30 datasets from Table 3.2. The results are depicted in the following Figure 3.19, where each rmse measure is given as the average over ten runs. Notably, all scatter plots exhibit the tendency of less rmse over the usage of more control points, thus the approximation of the target embedding becomes more accurate. For datasets of lower dimensionality (like e.g. *galaxy*) the projection matrix onto the first two principal components can even be mimicked exactly after placing only a few control points. Also note that the rmse, although it technically has no upper bound, in these experiments never exceeds a value of 1.8. This comes due to the employed implementation of the PCA target embedding. Here, a library was used that not only centers the data, but also normalizes it by scaling each dimension to possess a standard deviation of 1.0. Because of this normalization, most data records are embedded into a range of -1 and 1 for each axis, which impedes the rmse from growing large.

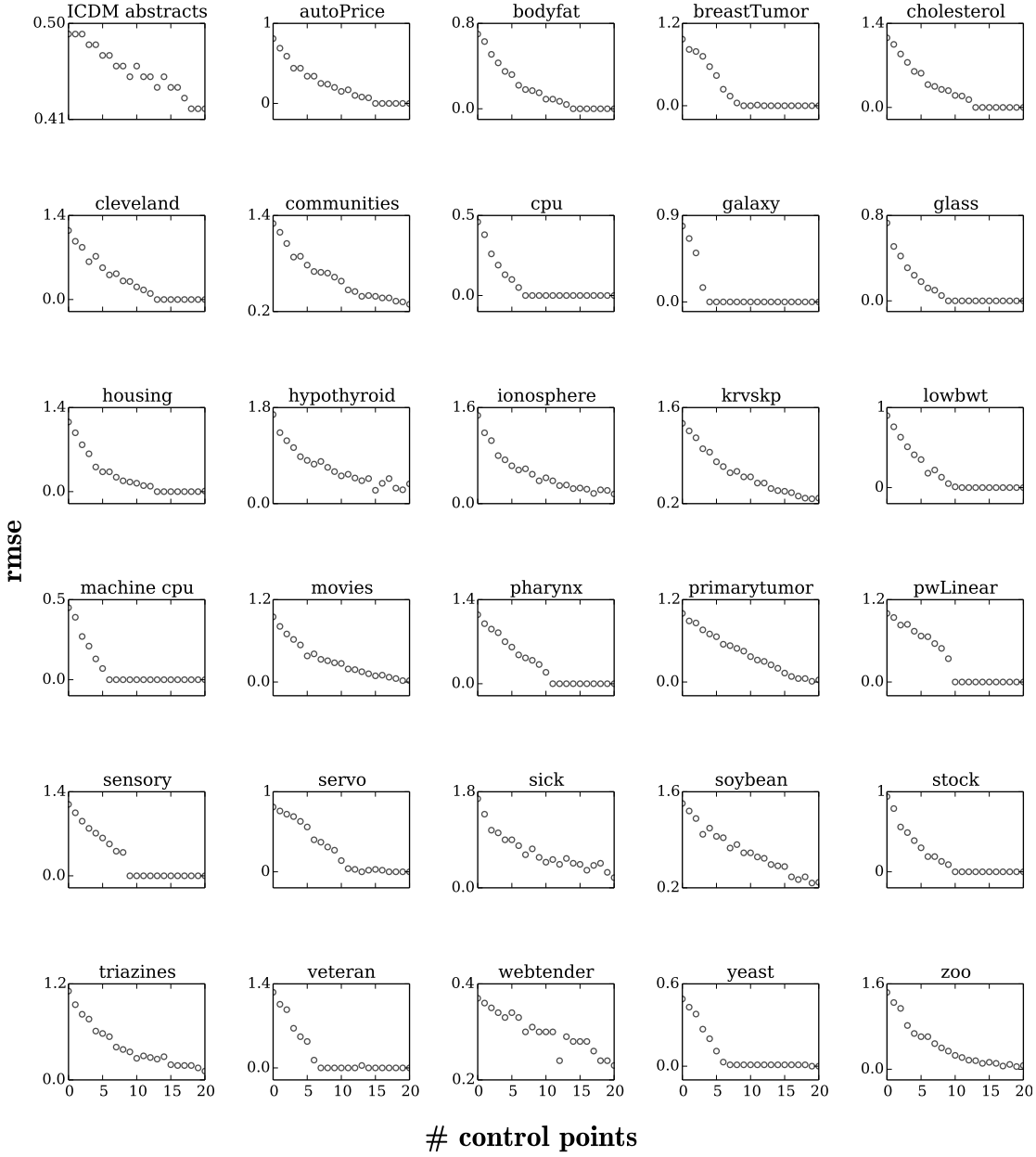


Figure 3.19.: Mimicking the embedding of the data into the third and fourth principal components via LSP on the 30 datasets from Table 3.2. Depicted is how the rmse between the target and the actual embedding develops over the amount of control points used.

3.3. Most Likely Embedding

While LSP is a fast and flexible method to create an interactive embedding, it also has some drawbacks. Computing the projection solely based on the control points' placement has the disadvantage that the resulting embedding may be poor with badly chosen, or too few control points utilized. Note that in the very initial embedding of LSP, with no control points set, every data record is projected to the origin of the embedding space. Further considering a sparse dataset with only few, or poorly chosen control points, many data records may not share any non-zero attributes with the control points. This is problematic, as it forces LSP again to embed these data records to the origin.

Another, related, drawback is that there is no inherent spread among the embedded points that stems from data records simply being different, regardless of the control points. To exemplify this, think of the cocktail dataset and a scenario, where the only control point that is placed is a cocktail that contains 100% *Vodka*. In this case, two cocktails that are completely different, except for a common share of *Vodka*, would be embedded exactly to the same location. The straight lines that form the arms in Figure 3.8 of the last chapter exhibit this behaviour and demonstrate that the effect also occurs in practice. This can be counter intuitive when utilizing the interactive embedding to explore the data. To remedy this effect, this section introduces a probabilistic interactive embedding method that considers a prior belief about the embedding. Key to the approach is the assumption that the projection matrix is matrix-normal distributed, a matrix-valued extension to the normal distribution. Given a prior belief on the projection matrix and conditioned on the control points' placements as evidence, the embedding with the least uncertainty about the placement of the data records is calculated. Hence the name **Most Likely Embedding** (MLE). A very condensed survey on this method can also be found in the workshop publication (Paurat et al., 2014). Note that this idea is close to Iwata, Houlsby and Ghahramani's work (Iwata et al., 2013), who utilizes MLE as a semi-supervised static embedding technique, rather than directly interacting with the embedding. Additionally and in contrast to their method, the here introduced variant does not use the Laplacian of the nearest-neighbor graph, but instead the projection onto the first two principal components as prior belief about the embedding. In addition, the here discussed method considers the placement of the control points not to be exact. It rather assumes that analyst places the control points "about right" and that small deviations from the location are tolerable.

3.3.1. Matrix Normal Distribution

The matrix normal distribution is a generalization of the multivariate normal distribution to matrix-valued entries. Similar to the multivariate normal distribution, mean and variance terms are required. For the matrix normal distribution, however, there exist a row and a column variance and the parameters are given in matrix form. For a random matrix $R \in \mathbb{R}^{p \times q}$ that follows the matrix normal distribution $\mathcal{MN}_{p,q}(M, \Sigma, \Psi)$

the probability density function is given by

$$p(R|M,\Sigma,\Psi) = \frac{1}{(2\pi)^{\frac{pq}{2}}|\Sigma|^{\frac{q}{2}}|\Psi|^{\frac{p}{2}}} \cdot \exp\Big(-\tfrac{1}{2}\operatorname{tr}\big[\Sigma^{-1}(R-M)\Psi^{-1}(R-M)^\top\big]\Big).$$

Here $M \in \mathbb{R}^{p\times q}$ is the location parameter that encodes the mean and $\Sigma \in \mathbb{R}^{p\times p}$ and $\Psi \in \mathbb{R}^{q\times q}$ relate to the row and column covariances of M. Note that there is also a direct relationship between the matrix normal and the multivariate normal distribution. If the matrix R follows the matrix normal distribution $\mathcal{MN}_{p,q}(M,\Sigma,\Psi)$, then $\operatorname{vect}(R)$ follows the multivariate normal distribution $\mathcal{N}\big(\operatorname{vect}(M), \Psi\otimes\Sigma\big)$.[4]

The parameters M, Σ and Ψ of a matrix normal distribution have intuitive interpretations. The parameter M is simply the (matrix-valued) expected value of R, that is $M = \mathbb{E}[R]$. From $\operatorname{vect}(R) \sim \mathcal{N}\big(\operatorname{vect}(M), \Psi\otimes\Sigma\big)$, we can see that the term $\Psi\otimes\Sigma$ can also be rewritten as $c\Psi \otimes c^{-1}\Sigma$ for an arbitrary constant $c \in \mathbb{R}\setminus\{0\}$. For the choice of $c = 1$, however, Σ and Ψ are directly the row and column covariance matrices of M and we retrieve

$$\Sigma = \mathbb{E}\big[(R-M)^\top(R-M)\big] \text{ and}$$
$$\Psi = \mathbb{E}\big[(R-M)(R-M)^\top\big].$$

A very useful property for our purpose of the matrix normal distribution is that it is closed under bilinear transformations. This means that, given $R \in \mathbb{R}^{p\times q}$ has a matrix normal distribution $R \sim \mathcal{MN}_{p,q}(M,\Sigma,\Psi)$, for any two arbitrary matrices $A \in \mathbb{R}^{n\times p}$ and $B \in \mathbb{R}^{m\times q}$ it holds that $ARB^\top \sim \mathcal{MN}_{n,m}(AMB^\top,\ A\Sigma A^\top,\ B\Psi B^\top)$.

Consider the scenario of embedding D dimensional data into a d dimensional space, and suppose we have a matrix normal belief about a linear embedding matrix $R \in \mathbb{R}^{d\times D}$, i.e. $p(R) = \mathcal{MN}(R|M,\Sigma,\Psi)$. Given a vector from the high-dimensional input space, $x \in \mathbb{R}^D$, our belief about the embedded location $u = Rx^\top$ is multivariate normal, meaning:

$$p(u) = p(Rx^\top) = \mathcal{N}\big(Mx^\top,\ (x\Psi x^\top)\Sigma\big).$$

This can be seen by setting $A = I,\ B = x$. Also notice, that if the norm of x is fixed to a constant $xx^\top = C$, the vector with the most uncertainty about its embedded location subject to this constraint is in the direction of the eigenvector of Ψ with largest eigenvalue. The same holds about the vector with the least uncertainty, as it is in the direction of the eigenvector of Ψ with smallest eigenvalue. Now, in a similar way, given a dataset of n data records $\in \mathbb{R}^D$, $X \in \mathbb{R}^{n\times D}$, our belief about all embedded data records $U = RX^\top$ is matrix normal and we have

$$p(U) = \mathcal{MN}(U \mid MX,\ \Sigma,\ X^\top\Psi X).$$

[4] Here $\operatorname{vect}(R)$ denotes the vectorization operation, which stacks the columns of R to form a vector and the $\otimes$-symbol signifies the Kronecker product.

In order to be able to refine our belief about the projection matrix R by considering evidence (i.e. the placement of control points), we will need to be able to calculate **conditional distributions** of R.

Conveniently, the family of matrix normal distributions is closed under the operation of conditioning it on observing a subset of the rows, or columns. Analogous to conditioning a multivariate normal distribution, this means that conditioning a matrix normal distributed matrix R on observing a subset of the columns, or rows, leaves us with a matrix normal distribution on the remaining columns, or rows. Consider the matrices R, M, Σ and Ψ partitioned in the following way:

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \quad M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \quad \Psi = \begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix},$$

with $R_{11} \in \mathbb{R}^{r \times s}$, $M_{11} \in \mathbb{R}^{r \times s}$, $\Sigma_{11} \in \mathbb{R}^{s \times s}$ and $\Psi_{11} \in \mathbb{R}^{r \times r}$.

Further, we define

$$R_{1*} = \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} \text{ and } R_{*1} = \begin{bmatrix} R_{11} \\ R_{21} \end{bmatrix}$$

and R_{2*}, R_{*2}, Σ_{1*}, etc. in an analogous way. Now we can express the conditional row and column probabilities as follows:
Given that Σ_{22}^{-1} exists,

$$p(R_{1*} \,|\, R_{2*}) = \mathcal{MN}(M_{1*} + \Sigma_{12}\Sigma_{22}^{-1}(X_{2*} - M_{2*}),\ \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21},\ \Psi) \tag{3.2}$$

and given that Ψ_{22}^{-1} exists,

$$p(R_{*1} \,|\, R_{*2}) = \mathcal{MN}(M_{*1} + (X_{*2} - M_{*2})\Psi_{22}^{-1}\Psi_{21},\ \Sigma,\ \Psi_{11} - \Psi_{12}\Psi_{22}^{-1}\Psi_{21}). \tag{3.3}$$

In the next section, we will use this property to learn the most likely projection matrix, given a prior belief about the embedding and a set of control points placed by the analyst within the embedding.

3.3.2. Deriving the Most Likely Embedding

Suppose we believe that the projection matrix R follows a matrix normal distribution

$$p(R|\theta) = \mathcal{MN}(R|M, \Sigma, \Psi),$$

where θ denotes the hyperparameters M, Σ, and Ψ of the distribution. In the following, the dataset is denoted by X and consists of n data records from $\mathbb{R}^D$, organized in the data matrix of the shape $n \times D$. The m data records that correspond to control points are denoted as X_m and their locations within the two dimensional embedding are given by Y_m. In addition, for a simpler notation, the tuples of control points and corresponding data records (X_m, Y_m) that will be used as evidence are denoted by $\mathcal{E}$. Given this, the current understanding of the desired embedding of X can be presented to the analyst in

a straightforward way, by showing the maximum a posteriori (MAP) embedding of X. This means that the presented embedding is given by $MX^\top$, the mean of the projection matrix R, which constitutes the most likely embedding.

Let us further assume that the control points may not be exactly located, as intended by the analyst, but are corrupted by some iid[5] isotropic Gaussian distributed noise around the locations. Again, this can be expressed as the matrix normal distribution

$$p(Y_m|RX_m^\top, \theta, \sigma^2) = \mathcal{MN}(Y_m|RX_m^\top, I, \sigma^2 I), \tag{3.4}$$

which indicates that each of the values in Y_m differs from $RX_m^\top$ by entrywise iid Gaussian noise with a variance of σ^2. From now on, σ^2 will also be included in the set of hyperparameters θ.

Now, in order to incorporate user placed control points, consider concatenating R and $RX_m^\top$ to form the block matrix $\begin{bmatrix} R & RX_m^\top \end{bmatrix}$. Again, we can utilize to the bilinearity of matrix normal distributions and together with $A = I$ and $B^\top = \begin{bmatrix} I_D & X_m^\top \end{bmatrix}$, we can derive the following distribution:

$$p\left(\begin{bmatrix} R & RX_m^\top \end{bmatrix}\right) = \mathcal{MN}\left(\begin{bmatrix} M & MX_m^\top \end{bmatrix}, \Sigma, \begin{bmatrix} \Psi & \Psi X_m^\top \\ X_m\Psi & X_m\Psi X_m^\top \end{bmatrix}\right).$$

This algebraic expression can be combined with equation (3.4) to find the joint distribution of R and the control point locations Y_m. By doing so, we retrieve

$$p\left(\begin{bmatrix} R & Y_m \end{bmatrix}\right) = \mathcal{MN}\left(\begin{bmatrix} M & MX_m^\top \end{bmatrix}, \Sigma, \begin{bmatrix} \Psi & \Psi X_m^\top \\ X_m\Psi & X_m\Psi X_m^\top + \sigma^2 I \end{bmatrix}\right).$$

Finally, by utilizing equation 3.2, we can reason about the linear projection matrix R that is most likely, given a prior belief about the embedding and conditioned on the observed values Y_m:

$$p(R|X_m, Y_m, \theta) = \mathcal{MN}(R|M_{R|\mathcal{E}}, \Sigma, \Psi_{R|\mathcal{E}}),$$

where

$$M_{R|\mathcal{E}} = M + (Y_m - MX_m^\top)(X_m\Psi_{R|\mathcal{E}}X_m^\top + \sigma^2 I)^{-1}X_m\Psi_{R|\mathcal{E}} \quad \text{and} \tag{3.5}$$

$$\Psi_{R|\mathcal{E}} = \Psi - \Psi X_m^\top(X_m\Psi X_m^\top + \sigma^2 I)^{-1}X_m\Psi. \tag{3.6}$$

In order to retrieve the final most likely embedding of all the data points X, we simply have to calculate $M_{R|\mathcal{E}}X^\top$. In case of a live-updating embedding, there are two different interaction scenarios that may occurr. The first one happens on relocating a control point. Here, only the matrix $M_{R|\mathcal{E}}$ has to be recalculated. In the second case, the analyst alters the selection of the control points, i.e. selecting new or de-selecting old ones. Now, $\Psi_{R|\mathcal{E}}$ has to be updated before $M_{R|\mathcal{E}}$ can be calculated.

[5] iid stands for "independent and identically distributed".

3.3.3. Connection to LSP

It is interesting to mention that MLE constitutes a generalization of LSP. Recall the projection matrix P of the LSP method from Section 3.2. In equation 3.1 it was calculated via $P = Y_m X_m (X_m^\top X_m)^{-1} = Y_m X_m^\dagger$. In case of the here presented most likely embedding method, this projection matrix corresponds to $M_{R|\mathcal{E}}$. With no prior in M (meaning that M is an all zero matrix $\mathbf{0}$), assuming that the control points are placed exact (meaning $\sigma = 0$) and the covariance among the non-control point data records is ignored (meaning that Ψ is the identity matrix $\mathbf{I}$), MLE turns out to be LSP. Using these assumptions, equation 3.5

$$M_{R|\mathcal{E}} = \underbrace{M}_{=\mathbf{0}} + (Y_m - \underbrace{M X_m^\top}_{=\mathbf{0}})(X_m \underbrace{\Psi}_{=\mathbf{I}} X_m^\top + \underbrace{\sigma^2 I}_{=\mathbf{0}})^{-1} X_m \underbrace{\Psi}_{=\mathbf{I}}$$

transforms into

$$M_{R|\mathcal{E}} = Y_m (X_m X_m^\top)^{-1} X_m.$$

Using the rules $(AB)^{-1} = B^{-1}A^{-1}$ and $AB = B^\top A^\top$ for matrix calculus, the expression $Y_m (X_m X_m^\top)^{-1} X_m$ can be rewritten as $Y_m X_m (X_m^\top X_m)^{-1}$, which is exactly the formulation of the projection matrix P from Section 3.2.

3.3.4. Evaluation

In this section the scalability and the flexibility of MLE are evaluated. In all presented experiments, the prior belief about the embedding is the projection of the data onto the first two principal components of a PCA. Keep in mind, however, that this prior belief can easily be substituted by any other embedding, it also does not necessarily have to be the result of a linear projection. An example of this method starting with a non-linear embedding is e.g. given by Iwata et al. (2013), who employ a Laplacean eigenmap embedding (see Section 3.1.2) as prior belief.

Scalability

MLE is a fast method that can be utilized to interact with an embedding in a live-updating manner. As earlier stated in Section 3.3.2, the calculations differ, depending on the type of interaction. If it solely consists of relocating control points, then only $M_{R|\mathcal{E}}$ has to be calculated according to equation 3.5. This can be done with a time complexity of $\mathcal{O}(mD^2)$, assuming $m \ll D$, as the computational complexity of calculating the pseudo inverse lies in $\mathcal{O}(mD^2 + m^2 D + m^3)$ (see Section 3.2.3) and all matrix multiplications in equation 3.5 can be done in $\mathcal{O}(mD^2)$, or less. If, however, the set of control points changes, the matrix $\Psi_{R|\mathcal{E}}$ has to be calculated in addition. For the additional cost of calculating $\Psi_{R|\mathcal{E}}$ via equation 3.6, the same arguments hold and it can also be done with a time complexity of $\mathcal{O}(mD^2)$. Table 3.3 shows that MLE is for the 30 datasets from Table 3.2 well suited to handle live-updating the embedding upon user interaction.

Dataset	Updates/Second only $M_{R\|\mathcal{E}}$	Updates/Second $\Psi_{R\|\mathcal{E}}$ and $M_{R\|\mathcal{E}}$	Dataset	Updates/Second only $M_{R\|\mathcal{E}}$	Updates/Second $\Psi_{R\|\mathcal{E}}$ and $M_{R\|\mathcal{E}}$
autoPrice	5899.3	3366.1	machine cpu	6067.5	3470.2
bodyfat	5814.6	3336.8	movies	3282.9	2279.9
breastTumor	5814.3	3407.2	pharynx	5630.1	3202.2
cholesterol	5660.9	3288.0	primary-tumor	5247.9	3011.3
cleveland	5617.0	2880.6	pwLinear	4877.4	3173.6
communities	2727.8	756.1	sensory	5004.0	3071.0
cpu	6101.7	3504.8	servo	5682.5	3170.5
galaxy	5708.6	3347.6	sick	2670.7	1962.1
glass	5923.9	3294.3	soybean	4602.3	1870.5
housing	5317.3	3135.0	stock	4513.0	2818.0
hypothyroid	2797.3	2001.8	triazines	5200.5	2372.2
ICDM abstracts	1850.5	89.0	veteran	5753.1	3340.3
ionosphere	5239.4	2909.0	webtender	1497.0	206.0
kr vs. kp	2974.4	1924.1	yeast	4122.4	2739.8
lowbwt	5803.0	3322.9	zoo	5720.6	3226.0

Table 3.3.: Updates per second that MLE achieves for changing five control points on the 30 datasets, averaged over 10 runs. Different calculations are needed for (i) only relocating the control points and for (ii) altering the set of selected control point.

Note that calculating the projection matrix only depends on d, D and m, but not on the number of attributes n. This is e.g. reflected in the amount of updates per second for the two datasets *pwLinear* and *stock*. Both have 10 attributes, but differ in the amount of data records, namely 200 and 950. Nevertheless, both achieve similar high amount of updates per second.

Flexibility

To demonstrate the flexibility of MLE, the experiment of mimicking a PCA embedding from Section 3.2.3 is recreated. However, as the implementation of the here utilized version of MLE considers the first two principal components already as prior for the embedding, the flexibility of the method is now demonstrated by placing the control points in such a way that now the third and fourth principal component are approximated. The results are depicted in the Figure 3.23. Again, the rmse measures are averaged over ten runs. For most of the scatter plots, the graphs exhibit a tendency of less rmse for a larger number of placed control points, which indicates a better approximation of the target embedding.

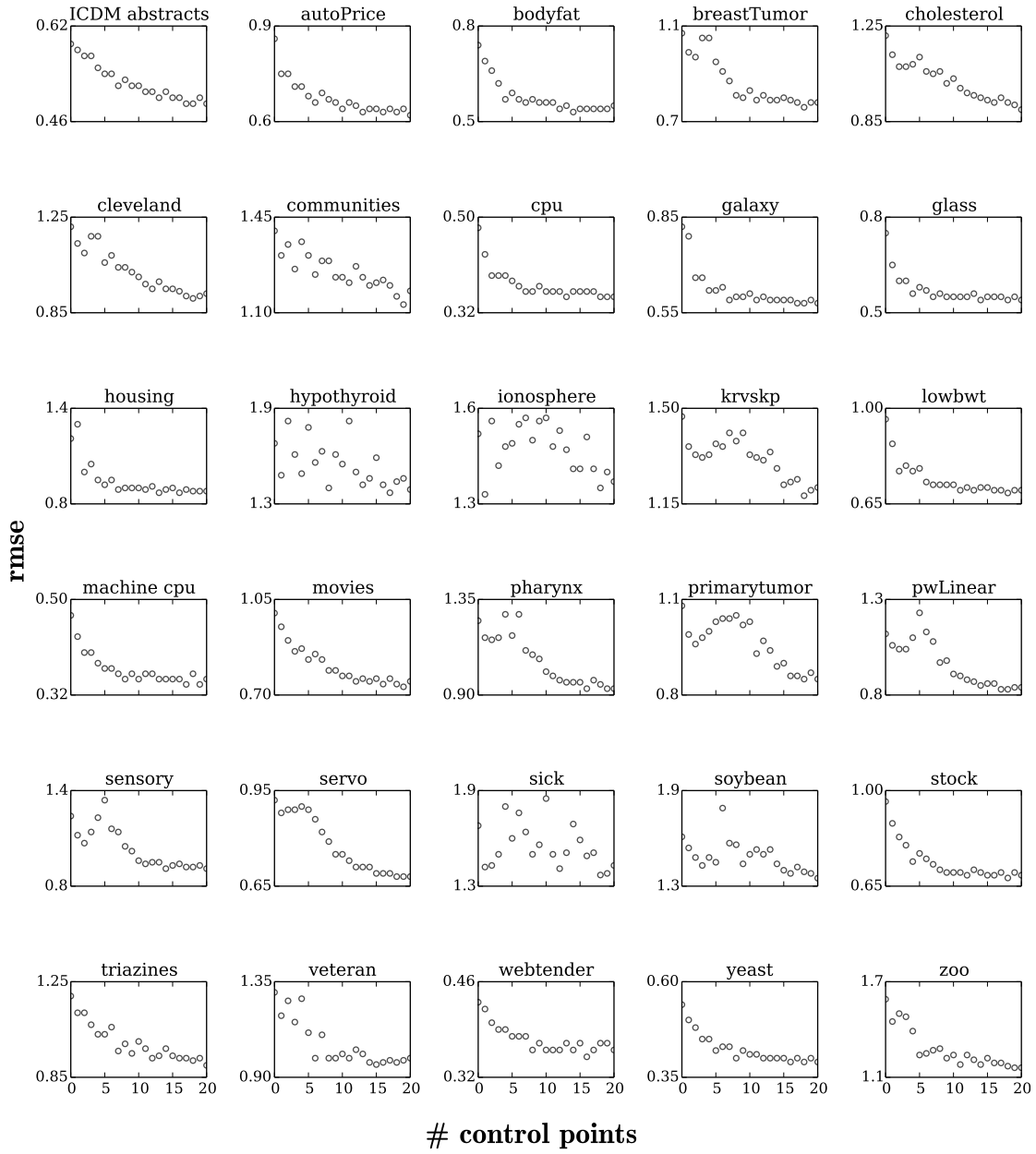


Figure 3.20.: Mimicking the $\mathsf{PCA}_{3,4}$ embedding via MLE on the 30 datasets. Depicted is how the rmse between the target and the actual embedding develops over the amount of used control points.

3.4. Constrained Kernel Principal Component Analysis

In this section, a **constrained kernel principal component analysis** (cKPCA) is introduced which not only embeds the control points to the user specified locations, but also puts emphasis to the variance among the embedded data points. This approach naturally remedies the spread issue that LSP exhibits for sparse data and augments the well established PCA method with the concept of interacting with it via control points. Apart from the point that PCA is well understood and has a clear interpretation, the kernelized version additionally offers some benefits. First, a kernel-PCA (KPCA) is able to capture non-linear dependencies within the data and second, show Ham et al. (2004) that choosing specially designed kernels can transform the PCA embedding into other embeddings, like e.g. Isomap, MDS or LLE. Hence, the here introduced cKPCA directly yields a way to provide interaction via control points to other (until now static) embedding techniques.

3.4.1. A Kernelized Version of PCA

Since in real world data often non-linear dependencies among the data records exist, considering solely linear projections may lead to poor results. A remedy to this can be found by utilizing **kernel methods**. Here the idea is to map the data in a very specific way to a (usually higher dimensional) feature space, with the hope that the mapping exposes a structure within the data that eases the task at hand. A very good general book on kernel methods, how they work, where they can be applied and how to design kernels has been published by Schölkopf and Smola (2002). It also contains a chapter on KPCA, which goes much along the lines of Schölkopf et al. (1997). In addition, a nice application of KPCA with the goal of de-noising data was published by Mika et al. (1998).

Recalling the regular PCA from Section 3.1.2, the idea was to find the axis u that exposed the most variance, when linearily projecting the data records $X = \{x_1, \ldots, x_n\}$ onto it. This brought us to the optimization problem

$$\begin{aligned} u^* &= \underset{u:\ ||u||=1}{\operatorname{argmax}}\ uX^\top Xu^\top, \quad \text{which can be rewritten as} \\ &= \underset{u:\ ||u||=1}{\operatorname{argmax}}\ \sum_{i=1}^{n} (\langle x_i, u \rangle)^2. \end{aligned} \tag{3.7}$$

Note that at the core of the above stated equation 3.7 lies a linear projection of the data X onto u. Generalizing this optimization problem can be done by substituting the linear function of this projection with another (possibly non-linear) function f. The idea here is to map all instances x and x' from the instance space $\mathcal{X}$ via a function Φ into some dot product space $\mathcal{V}$. Let us denote the inner product of the mapped data $\langle \Phi(x), \Phi(x') \rangle_{\mathcal{V}}$ with $k(x, x')$ and introduce a special dot product space that will be used for $\mathcal{V}$.

Definition (**Reproducing Kernel Hilbert Space**) from the book "Learning with Kernels" (Schölkopf and Smola, 2002). *Let $\mathcal{X}$ be a nonempty set (often called the index set) and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with the dot product $\langle \cdot, \cdot \rangle$ (and the norm $\|f\| := \sqrt{\langle f, f \rangle}$) if there exists a function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties.*

1. *k has the reproducing property $\langle k(x, \cdot), f \rangle = f(x)$ for all $f \in \mathcal{H}$; in particular $\langle k(x, \cdot), k(x', \cdot) \rangle = k(x, x')$*
2. *k spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{span\{k(x, \cdot) | x \in \mathcal{X}\}}$ where $\overline{X}$ denotes the completion of the set X.*

Choosing $\mathcal{V}$ to be a Hilbert space $\mathcal{H}_k$ of the kernel k that possesses the reproducing property and denoting $f(x)$ to be the inner product $\langle k(\cdot, x), f \rangle_{\mathcal{H}_k}$, the PCA problem of equation 3.7 can be restated as

$$f^* = \underset{f \in \mathcal{H}_k:\ \|f\|=1}{\operatorname{argmax}} \sum_{i=1}^{n} (\langle k(\cdot, x_i), f \rangle_{\mathcal{H}_k})^2. \tag{3.8}$$

Note that the optimizer f^* of this problem can be an arbitrary element of $\mathcal{H}_k$. However, this is not the case, as the following shows. We can write the optimizer f^* as the sum of two othogonal parts v and w, with v being an element of $\mathcal{H}_X = \operatorname{span}\{k(x_i, \cdot) | x_i \in X\}$, i.e. the span the data spans within $\mathcal{H}_k$, and $w \in \mathcal{H}_k \setminus \mathcal{H}_X$. Inserting $f = v + w$ into equation 3.8 and using the bilinearity of the dot product yields

$$\langle k(\cdot, x_i), v + w \rangle_{\mathcal{H}_k} = \langle k(\cdot, x_i), v \rangle_{\mathcal{H}_k} + \underbrace{\langle k(\cdot, x_i), w \rangle_{\mathcal{H}_k}}_{=0, \text{ as } w \perp \mathcal{H}_X}$$

This lets us conclude, that the optimizer f^* lies in $\mathcal{H}_X$ and thus can be represented as a linear combination of the training data x_i, leading us to $f^* = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$ with $\alpha \in \mathbb{R}^n$. Stating that f^* can be written in this algebraic form is also referred to as the **weak representer theorem**. Schölkopf et al. (2001) have shown that for a whole family of optimization problems, defined over a reproducing kernel Hilbert space, which all minimize a regularized empirical risk functional, the minimizer f^* can be expressed as a finite linear combination of kernel products from the training data x_i. The significance of this **representer theorem** is that for many different learning algorithms the solution can be expressed in terms of the training examples.

3.4.2. The cKPCA Optimization Problem

The key idea behind the here introduced cKPCA is to augment the optimization problem of a classic (kernel) PCA with constraints that respect the placement of the control points. The resulting embedding is a projection of the data that embeds the control points to the user specified locations and simultaneously maximizes the variance of the unlabeled data 'along' the set of unit norm functions in Hilbert space. To ensure feasibility of the

resulting optimization problem while retaining satisfactory visualization, the usual hard orthogonality constraint of the principal directions is replaced by a conveniently chosen soft-orthogonality term in the objective function.

Let $X = \{x_1, \ldots, x_n\}$ be a sample from an instance space $\mathcal{X}$ with positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Without loss of generality it can be assumed that the first m points are labeled with $y = \{y_1, \ldots, y_m\}$. Furthermore, let $\mathcal{H}$ be the reproducing kernel Hilbert space of kernel k and $\mathcal{H}_X = \operatorname{span}\{k(x_i, \cdot) | x_i \in X\}$. To calculate each of the d dimensions of the embedding the unit $\mathcal{H}_X$-norm functions $f_1, \ldots, f_d$ of the following optimization problem are solved in succession

$$\begin{aligned} f_s^* = \underset{f \in \mathcal{H}}{\operatorname{argmax}} \quad & \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i) - \langle f, \mu \rangle \right)^2 - \nu \sum_{s'=1}^{s-1} \langle f_{s'}, f \rangle^2 \\ \text{subject to} \quad & \|f\|_{\mathcal{H}_X} = 1, \\ & f(x_i) = y_{is} \;\; \forall \;\; 1 \leq i \leq m, \end{aligned} \tag{3.9}$$

where $\mu = \frac{1}{n} \sum_{i=1}^{n} k(x_i, \cdot)$. Each term of this problem has a clear meaning. In the equation, the first term maximizes the variance and second encourages the found direction to be orthogonal to the already found ones; so far a classic KPCA with soft orthogonality. For the hard constraints, the fist part restrains the solution to lie on a hypersphere and the second condition demands the control points placement to be respected by the projection. Efficiently solving this problem remains possible in this case and follows along the lines of the upcoming Section 3.4.3.

3.4.3. Solving the cKPCA Optimization Problem

In this section the optimization problem 3.9 is rewritten in terms of matrix operations, such that a closed form solution can be given. The problem is defined over the reproducing kernel Hilbert space $\mathcal{H}$ with kernel $k(\cdot, \cdot)$ and the weak representer theorem (Dinuzzo and Schölkopf, 2012; Schölkopf et al., 2001) implies that its optimizer f_s^* can be represented as $f_s^* = \sum_{j=1}^{n} \alpha_{sj} k(x_j, \cdot)$, with $\alpha_{s1}, \; \alpha_{s2}, \; \ldots, \; \alpha_{sn} \in \mathbb{R}$.

Let K be the kernel matrix with its first m rows (the labeled ones) be denoted as K_m. Further let H be the centering matrix $H = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, where $\mathbf{1}$ denotes a vector of all ones. As an optimizer $f_s \in \mathcal{H}$, we can write $f_s = u_s + v_s$ with $u_s \perp v_s$ and $u_s \in \mathcal{H}_X$. Plugging the substitution into Equation (3.9), we conclude that the optimization objective is independent of v_s and the weak representer theorem holds in this case. For the computation of the s-th variance direction f_s $(s > 1)$, we additionally have the orthogonality terms $\langle f_s, f_{s'} \rangle = \langle u_s + v_s, f_{s'} \rangle = \langle u_s, f_{s'} \rangle$ $(s' < s)$, which are also independent of v_s. The hard constraint term $f(x_i) = y_{is}$ is also independent of v_s as it holds that $f_s(x) = u_s(x)$ for all $x \in X$. Therefore, the weak representer theorem holds for problem (3.9). Using $f = \sum_{j=1}^{n} \alpha_j k(x_j, \cdot)$ we can rewrite the individual terms of problem 3.9 in the following way:

$$
\begin{aligned}
f(x_i) &= \sum_{j=1}^{n} \alpha_j k(x_j, x_i) &&= K_i \alpha \\
\langle f, \mu \rangle &= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i k(\vec{x}_i, \vec{x}_j) &&= \frac{1}{n} \mathbf{1}^\top K \alpha \\
\sum_{s'=1}^{s-1} \langle f_{s'}, f \rangle^2 &= \alpha^\top K \left(\sum_{s'=1}^{s-1} \alpha_{s'} \alpha_{s'}^\top \right) K \alpha &&= \sum_{s'=1}^{s-1} \left(\alpha^\top K \alpha_{s'} \right)^2
\end{aligned}
$$

Now, constructing the matrix $HK = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)K = K - \frac{1}{n}\mathbf{1}\mathbf{1}^\top K$ and observing its i^{th} row yields us $K_i - \frac{1}{n}\mathbf{1}^\top K$. This result can be multiplied with α and we retrieve the term $K_i\alpha - \frac{1}{n}\mathbf{1}^\top K\alpha$, which is exactly $f(x_i) - \langle f, \mu \rangle$, the first term over which is summed in problem 3.9. Using the fact that $H^2 = H$ and K is symmetric, $\sum_{i=1}^{n} \left(f(x_i) - \langle f, \mu \rangle \right)^2$ can be expressed as $\sum_{i=1}^{n} (HK\alpha)_i^2 = (HK\alpha)^\top HK\alpha = \alpha^\top KHK\alpha$ and problem (3.9) can be rewritten, as follows

$$
\begin{aligned}
\alpha_s = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmax}} \quad & \frac{1}{n} \alpha^\top KHK\alpha - \nu \sum_{s'=1}^{s-1} \left(\alpha^\top K \alpha_{s'} \right)^2 \\
\text{subject to} \quad & \alpha^\top K \alpha = 1 \\
& K_m \alpha = y_s.
\end{aligned}
\tag{3.10}
$$

This can be reformulated in a much nicer form. Introducing a substitution $K^{\frac{1}{2}}\alpha = u$ and denoting

$$
\begin{aligned}
W &= K^{\frac{1}{2}} \left(\frac{1}{n} H - \nu \sum_{s'=1}^{s-1} \alpha_{s'} \alpha_{s'}^\top \right) K^{\frac{1}{2}}, \\
L &= K_m K^{-\frac{1}{2}},
\end{aligned}
$$

problem (3.10) can be stated as

$$
\begin{aligned}
\underset{u \in \mathbb{R}^n}{\operatorname{argmax}} \quad & u^\top W u \\
\text{subject to} \quad & u^\top u = 1, \\
& Lu = y_s.
\end{aligned}
\tag{3.11}
$$

To solve this problem, the linear term $Lu = y_s$ needs to be eliminated from the problem statement. If this term is of rank $m < n$, it can be eliminated and problem 3.11 can be rewritten to optimize a quadratic over an $(n-m)$-dimensional hypersphere. To do so, let us start with a QR factorization of the matrix $L^\top$. It implies that $L = R^\top Q^\top$, where $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $R \in \mathbb{R}^{n \times m}$ is an upper triangular matrix. Introducing a substitution

$$
Q^\top u = \begin{bmatrix} x \\ z \end{bmatrix}
$$

with $x \in \mathbb{R}^m$ and $z \in \mathbb{R}^{n-m}$, the objective function $u^\top W u$ of problem (3.11) becomes $u^\top Q Q^\top W Q Q^\top u = (Q^\top u)^\top Q^\top W Q (Q^\top u)$. Now, considering the matrix $Q^\top W Q$ partitioned into four sub-matrices, with the split at position m

$$Q^\top W Q = \begin{bmatrix} A & B^\top \\ B & C \end{bmatrix}, \text{ with } A \in \mathbb{R}^{m \times m}, B \in \mathbb{R}^{(n-m)\times m} \text{ and } C \in \mathbb{R}^{(n-m)\times(n-m)},$$

we can transform the objective function $u^\top W u$ into $x^\top A x + 2 z^\top B x + z^\top C z$. A similar transformation can be applied to the constraint terms. First, y_s is transformed into

$$y_s = L u = R^\top (Q^\top u) = \begin{bmatrix} \bar{R} \\ \mathbf{0}_{(n-m)} \end{bmatrix}^\top \begin{bmatrix} x \\ z \end{bmatrix} = \bar{R}^\top x$$

where $\bar{R}$ are the first n positions of R and $\mathbf{0}_{(n-m)}$ is a vector of length $(n-m)$ of all zeros. From this expression we can follow that $x = (\bar{R}^\top)^{-1} y_s$. Now for the other constraint, the following transformation can be done

$$1 = u^\top u = (Q^\top u)^\top (Q^\top u) = x^\top x + z^\top z$$

Knowing that x is a constant vector that we can calculate via $x = (\bar{R}^\top)^{-1} y_s$, this equation can be rearranged to obtain

$$z^\top z = \underbrace{1 - x^\top x}_{:= \, t^2}.$$

Again, using x to be a constant vector, we can eliminate $x^\top A x$ from the optimization objective and rewrite problem (3.11) with $b = -Bx$ as

$$\begin{aligned} \underset{z}{\text{argmax}} \quad & z^\top C z - 2 b^\top z \\ \text{subject to} \quad & z^\top z = t^2, \end{aligned} \tag{3.12}$$

This is a canonical form of optimization problems with a known solution, provided and derived by Gander et al. (1989). The solution steps follow along these lines: To compute the solution to an optimization problem of the form of problem 3.12, one can first form the principal Lagrangian function and show that the maximum is achieved for the largest value of the Lagrangian parameter associated with the hypersphere constraint. Then, it is shown that finding the largest value of this parameter is equivalent to solving a quadratic eigenvalue problem. Furthermore, the quadratic eigenvalue problem can be written as a linear eigenvalue problem using block matrices, which yields the following solution:

$$z^* = (C - \lambda_{max} \mathbf{I}_{(n-m)})^{-1} b,$$

where λ_{max} is the largest real eigenvalue of

$$\begin{bmatrix} C & -\mathbf{I}_{(n-m)} \\ -\frac{1}{t^2} b b^\top & C \end{bmatrix} \begin{bmatrix} \gamma \\ \eta \end{bmatrix} = \lambda \begin{bmatrix} \gamma \\ \eta \end{bmatrix}.$$

Hence, the solution to problem (3.10) is given by

$$\alpha_s^* = K^{-\frac{1}{2}} Q \begin{bmatrix} \left(\bar{R}^\top\right)^{-1} y_s \\ \left(C - \lambda_{max} \mathbf{I}_{(n-m)}\right)^{-1} b \end{bmatrix}.$$

As Q, $\bar{R}$, C, λ_{max} and b can be calculated from the original data and the placement of the control points, the above equation provides a solution to problem 3.9, given by $f = \sum_{j=1}^{n} \alpha_j k(x_j, \cdot)$.

3.4.4. Other Knowledge-based Constraints

An enhanced version of the initial constrained kernel PCA problem 3.9 was investigated by Oglic et al. (2014). The idea there is to formulate additional knowledge based constraints and integrate them into the optimization problem, such that it can still be transformed to the canonical form of problem 3.12 and thus be solved in the same way. The new objective function can be formulated as

$$\begin{aligned} f_s^* = \underset{f \in \mathcal{H}}{\operatorname{argmax}} \quad & \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i) - \langle f, \mu \rangle\right)^2 + \lambda \Omega(f, \theta) \\ \text{subject to} \quad & \|f\|_{\mathcal{H}_X} = r, \\ & \Psi(f, \theta) = 0, \end{aligned} \tag{3.13}$$

with Ω denoting a soft- and Ψ a hard-constraint term. The variable $r \in \mathbb{R}^+$ describes the radius of the hypersphere over which is optimized and θ represents a set of hyper-parameters, which depend on the utilized constraint type.

Oglic et al. (2014) show for three different types of constraints how they can be formulated and merged with the optimization problem without altering the canonical form of problem 3.12. In the previous Section 3.4.2, the placement of the control points was treated as hard constraint. To still ensure feasibility of the optimization problem, the hard orthogonality of the maximum variance directions, as it is usually demanded for a PCA, was replaced by a soft-orthogonality. This results in an embedding that maximizes the variance along the directions and their orthogonality as much as possible, while ensuring that the projection of the control points falls exactly to the user desired locations. Using the new formulation, it is possible to flip this around and demand strictly orthogonal directions of maximum variance, while the projections of the control points should fall "as good as possible" to the user desired locations. A different type of knowledge based constraint that can be incorporated, is to provide must- and cannot-link information about individual pairs of data records. Here, the analyst declares some of the data records to be linked together, and some of them to be disjoint. The resulting embedding places must-linked data records close to each other and spatially separates the cannot-linked ones. As a last constraint type, label information can be incorporated. Here, the underlying assumption is that data records of the same class-label should be embedded

close to each other. In their publication, Oglic et al. (2014) suggest to place a term in the objective function which rewards a placement of the labeled data records close to the class mean within the embedding. As a benefit to this enhanced formulation of the optimization problem, all of these constraint types can also be utilized simultaneously, as substitution allows it to transform the problem to the canonical form 3.12 again. Note that technically, it is possible to demand the same thing simultaneously as hard- and as soft-constraint. (For instance, the placement of control points, or the orthogonality of the embedding directions.) However, as the hard constraint constitutes the optimal case for the soft constraint, it will drive the solution of the optimization problem.

3.4.5. Evaluation

Apart from investigating the scalability and flexibility of cKPCA, this evaluation also studies the spread of the data points in the embedding.

Spread

Figure 3.21 (left) illustrates the problem with using LSP as an embedding technique on the ICDM 2001 abstracts dataset (Kontonasios and Bie, 2010) and shows how cKPCA is able to overcome it. The LSP embedding collapses towards the origin mainly because the dataset has sparse entries. The five control points (highlighted in red) have few to no attributes in common with the other embedded data records, which leaves LSP unable to embed them anywhere but the origin. As cKPCA also maximizes the variance, the resulting embedding has more spread. This gives the user more insights about the underlying structure of the data, as well as the possibility to better select new control points and interact with the embedding. Note that the small number of control points reflects the actual use case of an interactive embedding. In general, a user would not want to interact with too many control points, but rather with a few known or highly expressive ones.

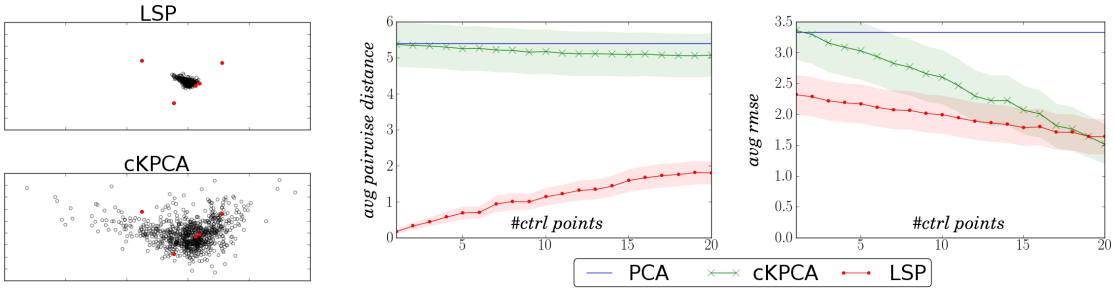


Figure 3.21.: Left, LSP and cKPCA embedding of the ICDM 2001 abstracts dataset. Middle and right, control points are placed according to their third and fourth principal components coordinates. The middle one shows the development of the averaged pairwise distance of the embedded data over the number of control points selected. The right one shows the development of the root mean squared error between the third and forth principal component and the actual embedding.

The middle picture of Figure 3.21 shows how the average pairwise distance of the embedded data develops depending on the amount of control points. In this experimental setting a number of control points were chosen uniform at random and placed according to their third and fourth principal components coordinates. One can see how cKPCA starts as a regular PCA and develops with more and more control points selected towards the new embedding, while keeping the high spread among the embedded data records from the beginning on. LSP on the other hand initially places all points at the origin and only slowly develops the desired spread. The right part of Figure 3.21 shows how the rmse between the projection onto the third and forth principal direction and the cKPCA and LSP embedding develops over the amount of placed control points. With an increasing number of control points, cKPCA develops away from the regular PCA embedding towards the new embedding.

In addition, the stray of cKPCA and LSP was evaluated on the thirty datasets that were already used in the evaluation of LSP in Section 3.2.3. The above explained experimental setup was performed for a fixed number of five control points. For the cKPCA algorithm, a linear kernel was used and the ν-value, which weights the axes-orthogonality term was set to 1.0. Table 3.4 shows the averaged results over 10 runs. One can see that the resulting embeddings, especially for the sparse datasets *ICDM abstracts* and *webtender*, have a higher average pairwise distance among the embedded data for the cKPCA algorithm. This can be interpreted as a sign of more stray among the embedded data.

	Pairwise distance			**Pairwise distance**	
Dataset	cKPCA	LSP	**Dataset**	cKPCA	LSP
autoPrice	2.25	1.32	machine cpu	1.22	1.12
bodyfat	1.88	1.17	movies	2.48	1.52
breastTumor	1.85	1.38	pharynx	2.01	1.20
cholesterol	2.08	1.09	primary-tumor	2.23	1.00
cleveland	2.17	1.09	pwLinear	1.93	1.50
communities	6.13	2.54	sensory	2.09	1.40
cpu	1.25	1.00	servo	2.09	1.43
galaxy	1.20	1.20	sick	2.43	1.04
glass	1.85	1.43	soybean	3.24	1.62
housing	2.01	1.28	stock	1.92	1.58
hypothyroid	2.38	1.05	triazines	4.23	2.41
ICDM abstracts	3.83	0.36	veteran	1.81	1.56
ionosphere	2.90	1.49	webtender	0.52	0.08
kr vs. kp	2.71	1.21	yeast	0.91	0.58
lowbwt	1.84	1.30	zoo	2.53	1.58

Table 3.4.: Average pairwise distances for the cKPCA and the LSP algorithm for a fixed amount of five control points on thirty datasets, averaged over 10 runs. As cKPCA is designed to expose more spread among the projected data records, it shows consistently larger values than LSP.

Scalability

Solving the optimization problem behind cKPCA is a time consuming operation, which can be performed with a time complexity of $\mathcal{O}(dn^3)$, where d is the dimensionality of the embedding space. For datasets with only few records this approach is fast enough for a live-updating embedding. However, as soon as the number of data records exceeds a couple of hundred, the calculation becomes infeasible for live interaction. Oglic et al. (2014) show in their publication how to utilize rank-one updates, when updating the selection of control points. This way, only one initial costly calculation with cubic time complexity has to be performed. Subsequent selection and de-selection of control points, as well as and relocating them can be calculated in $\mathcal{O}(d^2n^2)$. This is a huge improvement, which makes cKPCA also viable for larger datasets. Table 3.5 shows the execution times on the 30 datasets for the regular and the rank-one update approach. In the experiment, the average execution time was measured of ten times updating a cKPCA embedding with five control points, chosen uniform at random. In addition to Table 3.5, the Figure 3.22 shows how the execution times depend on the number of data records, as listed in Table 3.2. Note that the plot has a logarithmic scale on the y-axis.

	Seconds/Update			Seconds/Update	
Dataset	**Regular**	**Rank-one**	**Dataset**	**Regular**	**Rank-one**
autoPrice	0.04	0.06	machine cpu	0.09	0.06
bodyfat	0.16	0.11	movies	1050.68	1.88
breastTumor	0.22	0.08	pharynx	0.12	0.08
cholesterol	0.27	0.13	primary-tumor	0.36	0.13
cleveland	0.26	0.14	pwLinear	0.08	0.09
communities	355.11	1.87	sensory	4.76	0.24
cpu	0.09	0.06	servo	0.06	0.05
galaxy	0.33	0.04	sick	3172.33	2.88
glass	0.09	0.07	soybean	11.17	0.28
housing	1.28	0.18	stock	33.82	0.23
hypothyroid	3180.72	2.89	triazines	5.65	0.02
ICDM abstracts	26.29	0.53	veteran	0.07	0.05
ionosphere	0.40	0.14	webtender	208.09	1.32
kr vs. kp	1823.93	2.72	yeast	137.98	0.47
lowbwt	0.07	0.06	zoo	0.01	0.02

Table 3.5.: Execution time of the 30 datasets from Table 3.2 for performing an update step with the regular cKPCA approach and for using rank-one updates.

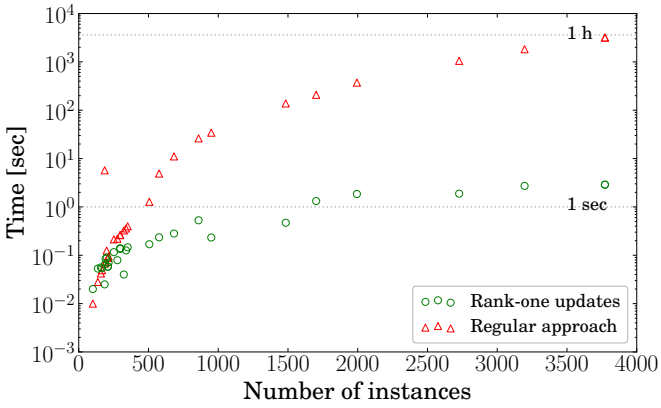


Figure 3.22.: Execution time of the 30 datasets for updating the cKPCA embedding versus the number of data records, using the straight forward approach (triangles) and utilizing rank-one updates (circles).

Flexibility

To demonstrate the flexibility of cKPCA, the experiment of mimicking a PCA embedding from Section 3.2.3 is recreated. However, as the initial embedding of cKPCA with a linear kernel coincides with the regular PCA, in the following experiments, the projection onto the third and the fourth principal component is mimicked via the placement of control points. The results are depicted in the following Figure 3.23, again with each rmse measure given as the average over ten runs. As expected, all scatter plots exhibit the tendency to show less rmse for a larger number of placed control points, which indicates a better approximation of the target embedding.

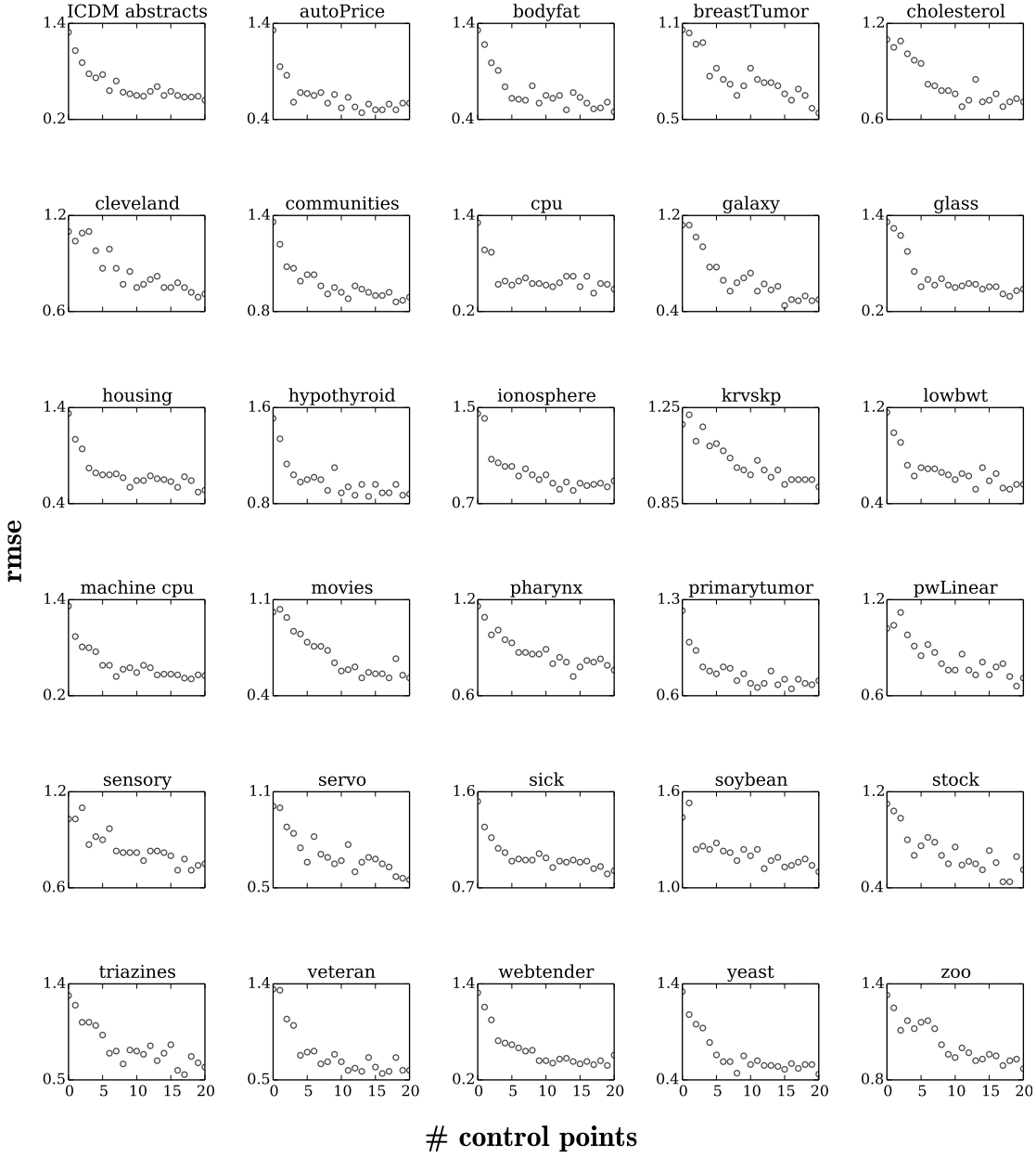


Figure 3.23.: Mimicking the $\mathsf{PCA}_{3,4}$ embedding on the 30 datasets from Table 3.2. Depicted is the development of the rmse between the target and the actual embedding, depending on the amount of placed control points.

3.5. Summary and Discussion

Visualizing high dimensional multivariate data via lower dimensional embeddings of it has a long tradition. In addition to just visualizing the data in a static way, this chapter investigated semi-supervised embedding techniques that let the analyst directly interact with the visualization and steer the whole embedding by moving individual data records within it. To this end, the interactive embedding methods LSP, MLE and cKPCA were introduced. Using these techniques, the analyst has not only the possibility to observe the whole data and its inter-dependencies from a birds-eye perspective, but also to intuitively browse it and elaborate a metric that reflects domain knowledge for the task at hand.

Computationally, LSP is by far the fastest of the three introduced methods, since the calculation of the projection matrix solely depends on the control points and their embedding locations. As the experiments have shown, it is also flexible in terms of mimicking a target embedding and only few control points are needed to shape out a concept. However, depending on the sparsity of the dataset at hand, embeddings that are generated by using the LSP technique may lack a certain spread among the embedded data records. This phenomenon becomes more severe with sparser data. An additional drawback is that the initial embedding, with no control points placed, projects all data records to the origin. For practical purposes, however, LSP does not have to start with no control points placed. A reasonable initialization can, for instance, be made by using the most extreme data records from the regular PCA embedding and to pre-place them as control points.

The MLE technique solves the problem of the initialization, as it is based on the idea of refining the prior belief about an embedding. In terms of efficiency, MLE can be calculated quite fast, as it runs for $m \ll D$ with a time complexity of $\mathcal{O}(mD^2)$. This makes it a viable choice to interact with embeddings of several thousand data records in a life-updating manner.[6] It is also notable that the algorithm merely demands any embedding coordinates as initial belief about the embedding. Obvious candidates for these can be found in the classical static embeddings (PCA, LLE, Isomap, etc.), which focus on structural aspects of the data. Using such a technique, offers the chance that the initial embedding already exhibits a good layout to the analyst. In addition to this flexible initialization, MLE also possesses a natural extension towards active learning. Knowing the conditioned covariance matrix $\Psi_{R|\mathcal{E}}$, which is calculated along the way of deriving the most likely embedding, enables the analyst to reason about the uncertainty in the placement of each data record. In a scenario, where the analyst's goal is not data exploration, but rather interactively composing a metric, this can be helpful. Here, the

[6] Like for LSP, the performance is slowed down the most by the graphical library that was used to draw the embedding.

embedding algorithm could suggest control point candidates to the analyst that help to converge faster on the desired layout.

Like MLE, also the cKPCA method from Section 3.4 remedies the geometric spread issue that LSP may encounter and comes with the extension of kernels. In a natural way, cKPCA also solves the initialization problem of LSP, as the plain embedding, without any constraints and using a linear kernel turns out to be a regular PCA embedding. It is notable that Ham et al. (2004) show in their work that MDS, Isomap and LLE can be considered instances of a kernel PCA with a suitably defined kernel matrix. Thus, the here introduced cKPCA offers a convenient way to extend these embedding techniques into interactive versions. The clear drawback of using cKPCA is its computational complexity. Even though the application of rank-one updates accelerates the interactive part massively, an initial computation of with a time complexity $\mathcal{O}(dn^3)$ of the matrix that is rank-one updated cannot be skipped. In addition, the calculations that are performed on interaction still depend on the number of data records. Hence, a fluent live-updating interaction with cKPCA via control points is only possible for smaller datasets and remains an open issue. A way to remedy the dependence on the number of data records might be to down-sample the kernel matrix to a convenient size and work only with an approximation of the KPCA.

In addition to the investigations and the experiments of this chapter, over the course of this thesis, the InVis tool has been developed, which integrates all three interactive (and some classic static) embedding methods in one application. The tool can be found at `http://www-kd.iai.uni-bonn.de/index.php?page=software_details&id=31`. Reading about an analyst interacting with data in a playful way and doing it are two very different things. The following Figure 3.24 shows a screenshot of an interactive session with InVis on the cocktail dataset, the manual can be found in Appendix A.

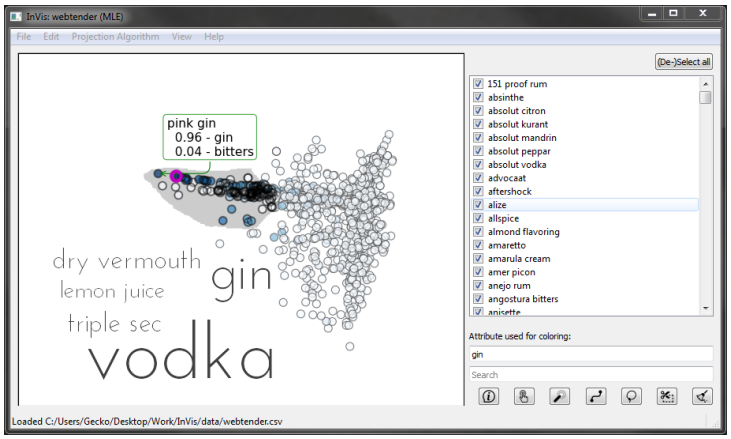


Figure 3.24.: A screenshot of the InVis tool for interactive data visualization that was developed over the course of this thesis.

The next Chapter will discuss a way to combine pattern mining and interactive embedding techniques in a fruitful manner. To do so, a pattern collection that is too large for manual inspection will be mined, then interactive embedding methods will be used to discover the common concepts that the patterns revolve around.

4. Synthesis

This chapter introduces a general procedure, which facilitates interactive embedding methods to enable the user to interactively explore and understand large amounts of discovered patterns. In Section 3.1.3 we have already seen a way to integrate pattern mining techniques into the explorative workflow of interacting with an embedding. Figure 3.5 there shows in addition to the visualized embedding the ten most interesting patterns of a selected region, i.e. the patterns augment the visualization. This section, however, follows a different direction and proposes to create an embedding of the patterns themselves. Now every point in the embedding represents a pattern. Further utilizing the earlier introduced interactive embedding methods, enables the analyst to quickly gain an overview on the distribution of all interesting patterns and their underlying structure.

Note that classical pattern mining algorithms reduce the output for the analyst to a small set of highly interesting and diverse patterns. However, by discarding most of the patterns, these methods have to make a trade-off between ruling out potentially insightful patterns and possibly drowning the analyst in results. Combining interactive embedding methods with pattern discovery, on the other hand, excels by working with larger pattern collections, as the underlying pattern-distribution emerges more clearly. Actively exploring this distribution enables the analyst to understand the major concepts that make a pattern of the considered dataset interesting and helps to interpret the patterns that are reported by classical pattern-mining methods.

4.1. Embedding Patterns

The question arises how to embed a set of discovered patterns. While the natural representation of patterns are item sets, most embedding techniques either rely on the data records being represented as vectors, or on a matrix of pairwise distances between the records. A very natural way to measure the distance between two sets is to turn Jaccard's similarity index into a distance measure. The Jaccard index is defined in the

following way:
Let A and B be two finite sample sets. The Jaccard similarity index $J(A,B)$ between these two sets is the size of the sets intersection over the size of their union.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

This similarity is bounded between zero and one and can be turned into a proper distance measure, denoted as Jaccard's distance $D_J(A,B)$, by subtracting it from one.

$$D_J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

In order to embed a set of patterns, Berardi et al. (2006) proposed to measure the pairwise distances between the patterns via Jaccard's distance and employ classical MDS to render the pattern-embedding. Their work, however, does not trace the benefits of considering larger pattern collections, nor does it utilize interactive methods to explore the embedding. Their main goal is to introduce a more insightful presentation for a set of discovered patterns. However, Berardi et al. (2006) do point out another interesting aspect of measuring the distance between two patterns. Considering that patterns describe subsets of a given dataset, there are two different sets which can be used to calculate the pattern distance; which one to employ depends ultimately on the task.

1. The **intention** is the attribute set which constitutes the pattern itself. Utilizing the intention-set considers the difference between the patterns in terms of how the patterns are described.

2. The **extention** is the support-set of a pattern. This is the set of all data records (or their transaction IDs) that support a given pattern. This can e.g. be used to measure the difference in the patterns semantic meaning, as one interesting aspect of a dataset (given by a sub set of its data records) can be described by several completely different patterns.

Another way to measure the distance between two sets over a finite common element language is to simply vectorize the sets by encoding each pattern as a binary vector over all ordered attributes and measure their Euclidean distance.[1] Consider, for instance, the pattern $\{$*Vodka, Orange juice*$\}$, which describes the earlier already mentioned *Screwdriver* cocktail. Assuming that the presence of the attributes *Vodka* and *Orange juice* is encoded in the first two dimensions of the binary representation, this pattern translates into the vector $(1,1,0,\ldots,0)$ Using this canonical form, every pattern over a dataset has a vector representation and all distances between the patterns can be calculated. In the remainder of the chapter this binary vectorization will be used in a case study on the cocktail dataset that demonstrates the above sketched synthesis between interactive embedding methods with pattern discovery.

[1] Note that the order can be arbitrary, but has to be fix. To this end, often the order of appearance in the representation of the data, or the lexicographical order of the attributes are utilized.

4.2. Interacting with Pattern Embeddings – A Case Study

This section introduces a simple and effective framework to combine pattern mining methods and the use of interactive exploration via an embedding in a natural way. The approach mainly consists of two steps: (i) mining a larger collection of patterns and (ii) exploring a visualized embedding of the patterns in an interactive way. To apply this idea practically, the procedure has to be broken down in a finer manner, as e.g.:

1. Mine a large collection of patterns.

2. Represent the patterns in a canonical way as vectors.

3. Embed these vectors with an interactive embedding method and explore the pattern distribution.

4. Inspect the emerging structures of interest deeper.

Note that each of the four steps still has a large amount of freedom, i.e. how many patterns to find, which algorithm to use, which vector representation to employ, the choice of the interactive embedding technique, etc. For the upcoming case study, three exemplary pattern mining methods were chosen and the resulting pattern collections were explored with the following settings of the above introduced framework:

1. 1000 patterns were mined from the cocktail dataset, using (i) frequent item-set mining, (ii) pattern sampling according to the rarity measure, as introducend in Section 2.4.3 and (iii) the top-1000 subgroup descriptions, using the binomial test quality measure, as introducen in Section 2.1.2.

2. Each pattern is represented by a binary vector over all occurring attributes of the pattern collection in lexicographical order, as described earlier in Section 4.1.

3. The pattern vectors are visualized, using the MLE technique from Section 3.3 with an initial PCA embedding as prior. The performed interaction to find interesting structures was mainly done by selecting and relocating control points.

4. Inspecting these structures deeper, as introduced in Section 3.1.3, was done by highlighting patterns that contain certain ingredients by color, by listing the five most-present single items of the structure in a tag cloud and by re-embedding and inspecting a selected region.

The following Table 4.1 shows a list of the top-10 patterns, retrieved by four classical pattern mining approaches. Over the course of the case study, these results will be used to illustrate how visually exploring larger pattern collections can help to interpret the top-k patterns and to understand the major aspects of the data and their relations.

Frequent (closed) item sets
Vodka
Orange juice
Amaretto
Pineapple juice
Grenadine
Gin
Baileys
Tequila
Kahlúa
Triple sec

Sampled patterns with high lift
Vodka & Cranberry juice
Vodka & Triple sec
Baileys & Kahlúa
Vodka & Gin
Vodka & Blue curaçao
Pineapple juice & Malibu rum
Vodka & Amaretto
Vodka & Rum
Orange juice & Amaretto
Vodka & Tequila

closed subgroups
Baileys
Crème de cacao
Milk
Kahlúa
Baileys & Kahlúa
Cream
Irish cream
Vodka & Baileys
Crème de banana
Baileys & Butterscotch schnapps

Δ_1-relevant subgroups
Baileys
Crème de cacao
Milk
Kahlúa
Cream
Irish cream
Crème de banana
Butterscotch schnapps
Whipped cream
Vodka & Kahlúa

Table 4.1.: The ten highest quality patterns, delivered by different pattern-mining approaches on the cocktail dataset. Note that here the top-10 frequent item sets are also all closed. The high-lift patterns were sampled according to their rarity measure and the label for subgroup discovery indicates whether a cocktail is creamy or not.

4.2.1. Exploring the 1000 Most Frequent Patterns

Let us begin the case study by investigating the results of mining the 1000 most frequent patterns on the cocktail dataset. Figure 4.1 shows these patterns, represented as binary vectors over all items, embedded into their first two principal directions. Immediately, two well separated clusters can be seen that resemble roughly in their shape.

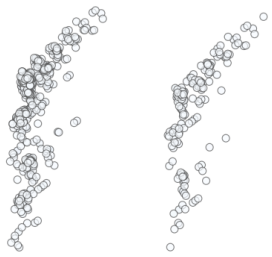


Figure 4.1.: The 1000 most-frequent item sets of the cocktail dataset, embedded into their first two principal directions.

Studying these clusters closer reveals that the right one contains only patterns that include the ingredient *Vodka*, the single most-frequent attribute in the original dataset, whereas the left one doesn't (see Figure 4.2, left). The second most-frequent ingredient, *Orange juice*, determines whether a pattern is mapped to the top or to the bottom of the embedding (see Figure 4.2, right).

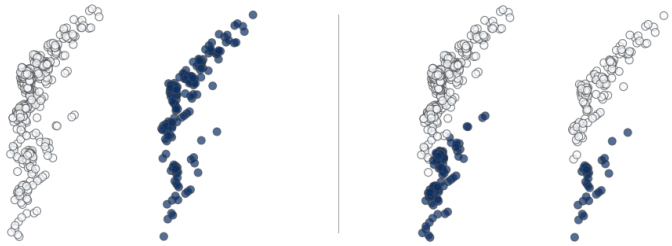


Figure 4.2.: Highlighting the ingredients *Vodka* (left picture) and *Orange juice* (right picture) in the plain PCA embedding of the 1000 most-frequent patterns of the cocktail dataset.

Interacting with the embedding by relocating two control points, as shown in Figure 4.3, unravels the blending of the patterns that contain *Orange juice* and the ones that don't. The resulting four clusters clearly separate the patterns by their presence or absence of the ingredients *Vodka* and *Orange juice*.

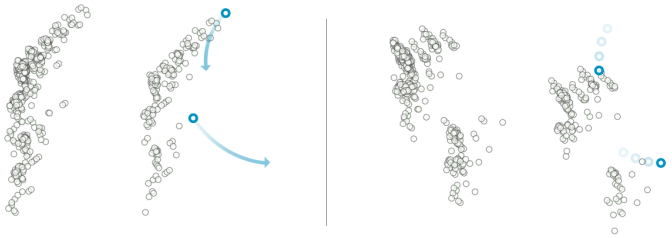


Figure 4.3.: Dragging two control points (emphasized in blue) to new locations, reveals a structure that was previously hidden in the PCA embedding. The four clusters indicate the presence or absence of the two ingredients *Vodka* and *Orange juice*.

Figure 4.4 inspects one of these emerging structures, the top-right "*Vodka* and no *Orange juice* cluster" from Figure 4.3, in a closer manner.
With a glance at the top-left picture of Figure 4.4 it is visible that the corresponding patterns containing *Vodka* but no *Orange juice* also frequently contain other strong alcohols, especially *Rum*, *Gin*, and *Triple sec*. There can also a sub-cluster structure within this particular embedding be observed, which is determined by the presence or absence of the ingredients *Rum* (top-right, highlighted in green) and *Gin* (bottom-left, highlighted in blue). The ingredient *Triple sec* (bottom-right, highlighted in red), although frequent within this cluster, seems not to contribute to the sub-structure, but can be found in all of the sub-clusters. This is an interesting finding, as *Triple sec* is much

Figure 4.4.: A closer look at the top-right cluster of Figure 4.3 reveals the ingredients that the patterns from the "*Vodka* and no *Orange juice* cluster" are frequently mixed with (top-left). The other three pictures indicate the presence of *Rum* (highlighted in green), *Gin* (blue), and *Triple sec* (red).

more frequent than *Rum*. In fact, *Rum* does not even occur among the ten most-frequent ingredients, yet it has a striking influence on the sub-structure of this cluster. Note that this is an insight that could not have been drawn purely from the results of Table 4.1. In the following sections similar studies will be performed, with pattern collections that were drawn according to more-sophisticated interestingness measures than frequency of occurrence.

4.2.2. Exploring 1000 Patterns Sampled According to their Rarity

A fruitful way to quickly draw patterns from a dataset according to different interestingness measures is to sample. Although sampling itself provides diversity among the drawn patterns, sorting them by the measure and listing only the top-k ones, as is done to retrieve the ten entries for the rare patterns in Table 4.1, can reintroduce a certain amount of redundancy. On the other hand, diversity is not impaired when exploring the set of all sampled patterns using this interactive exploration technique and the analyst is further enabled to discover the different concepts among the patterns.

In this study, a 1000 patterns were sampled from the cocktail dataset, according to their rarity measure, a variant of the lift measure which promotes patterns containing items that are statistically dependent, as already mentioned in Section 2.4.3. The rarity of a pattern approximates the probability of occurrence of the whole pattern weighted by the probabilities of the single items that build the pattern not occurring. So for a pattern p, consisting of k of these items $P = \{p_1, \ldots, p_k\}$, and the transactional database over a fix set of items $\mathcal{D}$, the rarity of p is calculated by

$$q_{\text{rare}}(\mathcal{D}, p) = \text{freq}(\mathcal{D}, p) \prod_{p_i \in p} \big(1 - \text{freq}(\mathcal{D}, p_i)\big).$$

Also note that there is a relation to the classic lift measure of a pattern. While rarity considers the absence-frequency of the singleton items, lift considers the inverse of the singleton's frequency. This can be seen, when writing the equation to calculate the lift measure in a similar format, as the equation to calculate the rarity above:

$$\text{lift}(\mathcal{D}, p) = \text{freq}(\mathcal{D}, p) \prod_{p_i \in p} \frac{1}{\text{freq}(p_i, \mathcal{D})}$$

In the following, the samples of the rare patterns were drawn by using the *direct local pattern sampling tool* which was provided by Boley et al. (2012) and can be downloaded from http://kdml-bonn.de/?page=software_details&id=23. The retained collection of the sampled patterns demonstrates well how the proposed approach benefits from the use of interactive embedding techniques. The plain PCA embedding of the frequent patterns in the previous Section 4.2.1 already exhibited a clear structure, which directly invited the analyst to further explore it. For this particular set of sampled patterns, however, this is not the case. Figure 4.5 shows the sampled rare patterns embedded into two dimensions, using different techniques, namely PCA, Isomap, and LLE.

Figure 4.5.: 1000 patterns sampled from the cocktail dataset, according to the *rarity* measure (Boley et al., 2012) and embedded, using different techniques: principal component analysis (left), locally linear embedding (middle), and isometric mapping (right).

Although these static embeddings exhibit no structures that immediately raise the analysts attention, relocating just one control point in the interactive embedding reveals clusters that were previously obscured. The following Figure 4.6 (top) shows this . The two middle pictures of the figure highlight the patterns containing *Vodka* (left) and *Orange juice* (right). The *Vodka* cluster can be clearly identified, but the other clusters come as a surprise. They do not relate to the *Vodka* / *Orange juice* segmentation that was already discovered in Section 4.2.1, but capture concepts of their own. The two highlighted ones at the bottom of the figure revolve around juicy and *Rum*-heavy cocktails. Because of the initially mentioned redundancy among the highest rated rare patterns, the results from Table 4.1 mainly exhibit patterns associated with *Vodka*. This interactive discovery approach, however, was able to overcome this drawback and reveal other, novel concepts among the high-rarity patterns.

Figure 4.6.: Relocating a control point, using the interactive embedding reveals a clear cluster structure (top). The middle pictures highlight the patterns containing *Vodka* (left) and *Orange juice* (right). The bottom pictures inspect the composition of two of these clusters.

4.2.3. Exploring the 1000 Most Striking Subgroup Descriptions

As we know from Section 2.1.2, patterns can be discovered according to different measures of interest. In the previous parts of this case study, the pattern sets were drawn proportional to their frequency or rarity. In some cases, however, the analyst might also want to consider label information. A classic pattern-mining approach that does so is subgroup discovery. It ranks the patterns by how much the label distribution of the data records described by the pattern diverges from the label distribution of the whole dataset. This section investigates the top-1000 closed subgroup descriptions from the cocktail dataset, ranked according to the binomial test quality measure, see Section 2.1.2. Figure 4.10 shows the embedding of these 1000 patterns into their first two principal directions.

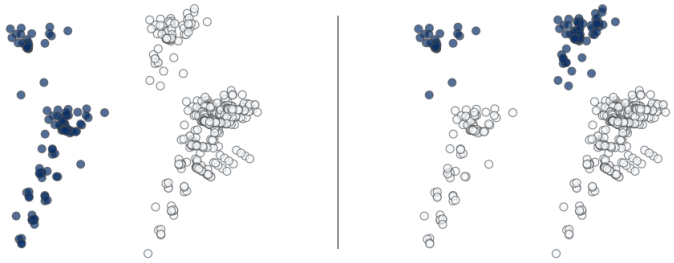


Figure 4.7.: The top-1000 subgroup descriptions associated to the label *creamy*, embedded onto their first two principal components. The four clusters coincide with the presence/absence of the two most striking ingredients among creamy cocktails: *Baileys* (left) and *Kahlúa* (right).

Similar to the embedding of the frequent patterns, but without the help of any interaction, the mined patterns fall directly into four clusters. This time, the clustering goes along with the presence or absence of two other frequently occurring ingredients: *Baileys* (left) and *Kahlúa* (right). From the list of frequent patterns in Table 4.1 it is known that these ingredients are highly frequent, while the list of subgroups indicates that they have a stark impact on the label of a cocktail. In this sense, the observed segmentation doesn't come as a total surprise. However, following the results of Table 4.1 might instead rather have lead to an expectation of *Crème de cacao*, instead of *Kahlúa*. The visualization helps to understand the relations among the listed patterns and invites for further exploration of the exhibited structure. To do so, this time the interaction with the embedding does not come via the earlier utilized control points, but rather by focusing on a subset of the distribution. Now the pattern collection is filtered to keep only the ones that contain neither *Kahlúa* nor *Baileys* and re-embed them into their first two principal directions. The selection corresponds to the patterns belonging to the bottom right cluster of Figure 4.7. The re-embedding of these selected patterns can be seen in Figure 4.8 below.

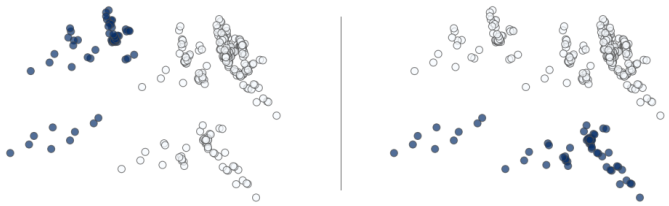


Figure 4.8.: A PCA embedding of the patterns belonging to the bottom right cluster, displayed in Figure 4.7. Again, the embedded patterns can be neatly segmented by the presence of two highly frequent ingredients, this time *Vodka* (left) and *Crème de cacao* (right).

As the re-embedding is not a zoom, but a newly calculated PCA embedding, structures can be discovered that were previously hidden due to the covariance among the patterns that are now filtered out. Once again the patterns form four clusters, corresponding to

highly frequent ingredients, this time *Vodka* and *Crème de cacao*. Note that this 'four cluster segmentation' is not immanently part of the used method, but stems form the sparsity which transactional databases often expose. To achieve a clearer separation of the clusters in the visualization, control points can be placed, as shown in the following Figure 4.9.

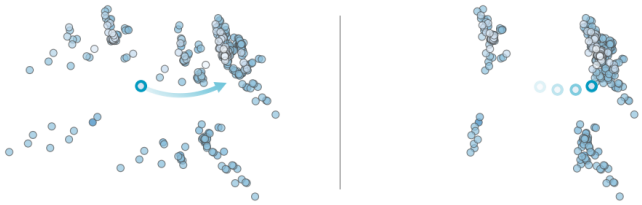


Figure 4.9.: To retrieve a better separation between the clusters, appropriate control points are selected and relocated.

As an example, let us pick two of the clusters from Figure 4.9 and study their compositions. Figure 4.10 below shows the five most-frequent ingredients within the patterns of these clusters in a tag cloud.

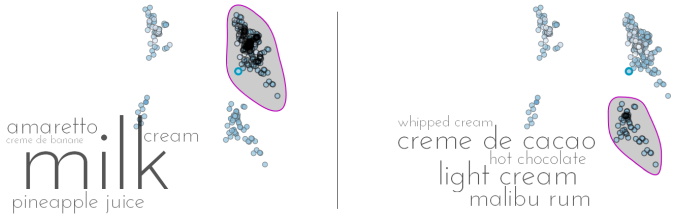


Figure 4.10.: Inspecting the contents of two of the emerging clusters. One interesting finding is the occurring separation between milky and chocolaty patterns. The cluster segmentation stems from the presence of the ingredients *Vodka* and *Crème de cacao*.

It can be observed that the inspected regions contain patterns that stem from two different types of creamy cocktails: milky and chocolaty ones. This is an interesting finding, as the strict separation between the clusters does not stem from the milky ingredients within the patterns, but from the ingredients *Vodka* and *Crème de cacao*. However, using the interactive visualization, made it possible to craft the hypothesis that milky and chocolaty cocktails form different types of creamy cocktails, offering a good next direction to explore.

4.3. Summary and Discussion

The case study demonstrated the possibilities which the research areas of interactive embeddings and pattern discovery can offer for one another. Although only demonstrated on a toy example, the study shows how interactive embedding techniques can provide an edge to counteract the information overload that comes naturally with the consideration of larger pattern collections. The presented framework mainly follows a two step procedure: (i) mine a large collection of patterns and (ii) explore a visualized embedding of the patterns in an interactive way. For the second step it is helpful to follow the information-seeking mantra and explore the obtained pattern collections in a top-down manner. Starting with a visual overview of the whole pattern distribution and then digging deeper on striking structures by interacting with the visualization and investigating the emerging structures in different ways. In the here presented case study this was namely done by reshaping the embedding via relocation of control points, filtering out and re-embedding the remaining patterns, listing the most-frequent items of an inspected structure and highlighting all patterns that contain an ingredient of interest.

Also note that interactively exploring the pattern collections brought up some insights that could not have been drawn by purely considering the results of Table 4.1. For instance, by inspecting the sub-clusters that emerged from our interaction, we found a surprisingly strong influence of the ingredient *Rum* on the cocktails containing *Vodka* but not those containing *Orange juice*. However, the strength of this approach lies not in these discoveries, but in the deeper understanding of the relations among the patterns that it provides in combination with the classical pattern-mining methods. By exploring the pattern embedding, interacting with it, exposing interesting structures, and always collating the crafted theories and insights with the pattern mining results from Table 4.1, we were able to develop an understanding of the different concepts that the original cocktail data revolves around.

5. Conclusion

Exploratory data analysis is a field of growing interest. As more and more data is collected in almost all areas of our life, naturally the desire arises to make sense and use of it. Commercial providers, like IBM Cognos[1], Tableau[2], SiSense[3], Spotfire[4] and Qlik[5], to name just a few of the interactive data dashboard and exploration suits, are starting to react to this demand. These commercial systems aim primarily at displaying selected attributes of the data (and aggregations of it) in linked visualizations, displayed together on a common dashboard. The analyst can then interact with the linked visualizations of the monitored attributes in the classical ways, as described in Section 3.1.3, and draw data-driven conclusions. However, there is a key difference between the here presented work and the commercial systems. A good dashboard is build from a collection of well chosen linked visualizations that monitor key aspects of the data. In order to select the right attributes and aggregations to display on the dashboard, the analyst has to know them in advance. In contrast, the here presented exploratory data analysis techniques aim at supporting the analyst in the task of finding these meaningful relations within the data in the first place.

For the practical use, these techniques have repeatedly demonstrated their usefulness. To give just two examples, in a data analysis piloting project with a globally operating power plant manufacturer, they helped to identify non-obvious reasons for why different divisions of the manufacturer performed drastically different. In another data analysis project, together with a braking-pad manufacturer, interactive projections lead the way in understanding the driving aspects behind an ingredient mixture and certain desired properties of the finished braking-pad.

Summary

This work studied the task of exploring multivariate data in an intuitive way. To this end, two very different approaches were investigated. The first one covered local pattern mining methods, which automatically discover and report interesting partitionings of the dataset. These partitionings usually yield insights about the data to the analyst and trigger further questions which, together with the reported results, guide the analyst in his data exploration. With respect to the pattern mining community, this work provides

[1] http://www.ibm.com/software/analytics/cognos
[2] http://www.tableau.com
[3] http://www.sisense.com
[4] http://spotfire.tibco.com
[5] http://www.qlik.com

the investigation of efficient algorithms for mining relevant and Δ-relevant patterns, as well as sampling patterns with a probability proportional to different interestingness measures.

In chapter 3, an alternative approach to exploratory data analysis was presented. There several interactive embedding methods were studied, which enable an analyst to directly shape and visually explore a projection of the data. These techniques allow a user to grasp the collection of all data records as a whole, understand the relations between the data records, incorporate domain knowledge and ultimately let him discover the interesting aspects of the dataset on his own. The chapter introduces three different techniques, which allow the analyst to directly control the projection by the usage of control points. These three different methods to directly interact with the embedded data constitute (i) a fast and scalable approach which, however, does not cope too well with sparse data, (ii) a probabilistic one that considers a prior believe about the desired embedding and evidence in form of placed control points and last (iii) an interactive kernel PCA that can, given the right kernel, also be utilized as interactive versions of Isomap, MDS and LLE. Note that all approaches towards data exploration, studied in the chapters 2 and 3, yield interpretable results to the analyst. Patterns are represented by a set of "attribute equals value" assignments, which is usually easy to understand by a domain expert. On the other hand, embeddings of the dataset show each data record as a point in a scatter plot graphic, where the distance between points encodes the relation between the data records.

In Chapter 4 of this work, a synthesis of the two data exploration approaches was introduced, which illustrated how both research areas can capitalize from one another in a practical setting. There, a general procedure is introduced that remedies the information overload an analyst may face, when investigating a large list of collected patterns. The idea behind this procedure is to expand the application range of the earlier introduced interactive embedding methods towards pattern collections. This allows the analyst to explore the dataset on different levels. The embedded patterns represent driving aspects of the underlying dataset, while the embedding reveals relations between the patterns. The analyst is now equipped with the power to gain an overview and understanding on the relations of the data records, as well, as an overview on the striking aspects of the dataset, given by its interesting patterns. To the best of the authors knowledge, this approach is the very first towards interactively exploring a large pattern collection in a visual way.

In addition to these contributions, the InVis tool for interactive data visualization was developed over the course of this thesis. It implements the interactive embedding techniques that were introduced in this work and some static embedding methods in a single program with a unified user interface. In order to be usable as a productive tool for interactive visual data exploration, it also implements some common static embedding methods and basic interaction methods, like searching, highlighting and filtering. An overview of the tool's full functionality can be seen in Appendix A.

Future Work

Having reviewed the direct contributions of this work, it is also important to note that it opens potential research topics which might emerge from it. One possible future research topic that might derive from this thesis is, to explain unpopulated regions of an embedding. Here the analyst would mark an empty region of the embedding space and local patterns could be used to describe on an intuitive level why this particular region is blank. Finding, e.g. frequent attribute combinations of virtual data records that would be embedded into the empty space could help the analyst to further understand the underlying distribution of the data and verify old, or craft new hypotheses about it. A step towards this idea has already been investigated and published on by Paurat et al. (2013a). There, the authors construct meaningful and novel data records that project to a user specified location within any linear projection. As there can be infinite many candidates to choose from, this is not a trivial task; the problem is also known as the pre-image problem.

Another possible future research direction of this work could be to equip all interactive embedding methods with additional functionality to express domain knowledge. Such could e.g. be the employment of *must-* and *cannot-link* constraints. For cKPCA, Oglic et al. (2014) have already shown how other constraint types can be incorporated into the optimization objective of the problem by introducing specially designed punishment and reward terms. However, for the LSP and MLE approaches this constitutes still a non-trivial and open issue.

In general, this work investigated different exploratory data analysis techniques and addresses the idea of combining and integrating them. However, apart from the here investigated methods, there exist numerous other approaches that can help an analyst to explore a dataset at hand; for instance text mining, clustering, classification, regression, time series analysis and other pattern mining and visualization techniques, to name just a few general techniques. In practice, however, they are often only used stand-alone to tackle a specific analysis task at hand. Daisy-chaining the techniques and comparing and investigating the intersection of the results is rarely done. It is the strong belief of the author that integrating data preparation, mining and visualization techniques into a fluent and overarching workflow constitutes the future of exploratory data analysis. Chapter 4 introduced an integrated workflow of two different knowledge discovery methods and elaborated the benefits of this synthesis. A great future research direction would be to find a general approach towards integrating different knowledge discovery techniques, but also data cleaning, transformation, aggregation and visualization techniques in an intuitive and practically usable way.

Bibliography

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and Inkeri A. Verkamo. Fast Discovery of Association Rules. In *Proceedings of the Conference on Advances in Knowledge Discovery and Data Mining*, pages 307–328. MIT Press, 1996.

Mohammad Al Hasan and Mohammed J. Zaki. Output Space Sampling for Graph Patterns. *Proceedings of the Very Large Databases Endowment*, 2(1):730–741, 2009.

Arthur Asuncion and David J. Newman. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2007.

Martin Atzmüller and Florian Lemmerich. Fast Subgroup Discovery for Continuous Target Concepts. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, pages 35–44, 2009.

Roberto Bayardo, Bart Goethals, and Mohammed J. Zaki, editors. *Proceedings of the IEEE International Conference on Data Mining Workshop on Frequent Itemset Mining Implementations, 2004*, volume 126 of *Proceedings of the Central Europe Workshop*, 2004. CEUR-WS.org.

Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural computation*, 15(6):1373–1396, 2003.

Margherita Berardi, Annalisa Appice, Corrado Loglisci, and Pietro Leo. Supporting Visual Exploration of Discovered Association Rules Through Multi-dimensional Scaling. In *Proceedings of Foundations of Intelligent Systems*, pages 369–378. Springer, 2006.

Mario Boley and Henrik Grosskreutz. Non-redundant Subgroup Discovery Using a Closure System. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 179–194. Springer, 2009.

Mario Boley, Tamás Horváth, and Stefan Wrobel. Efficient discovery of interesting patterns based on strong closedness. *Statistical Analysis and Data Mining*, 2(5-6): 346–360, 2009.

Mario Boley, Thomas Gärtner, and Henrik Grosskreutz. Formal Concept Sampling for Counting and Threshold-Free Local Pattern Mining. In *Proceedings of the SIAM International Conference on Data Mining*, pages 177–188, 2010.

Mario Boley, Claudio Lucchese, Daniel Paurat, and Thomas Gärtner. Direct Local Pattern Sampling by Efficient Two–Step Random Procedures. In *Proceedings of the 17th ACM SIGKDD Conferences on Knowledge Discovery and Data Mining*. ACM, 2011.

Mario Boley, Sandy Moens, and Thomas Gärtner. Linear Space Direct Pattern Sampling Using Coupling from the Past. In *Proceedings of the 18th ACM SIGKDD Conferences on Knowledge Discovery and Data Mining*, pages 69–77. ACM, ACM, 2012.

Eli T. Brown, Jingjing Liu, Carla E. Brodley, and Remco Chang. Dis-function: Learning Distance Functions Interactively. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 83–92. IEEE, 2012.

Soumen Chakrabarti, Earl Cox, Eibe Frank, Ralf Hartmut Güting, Jiawei Han, Xia Jiang, Micheline Kamber, Sam S Lightstone, Thomas P Nadeau, Richard E Neapolitan, et al. *Data Mining: Know It All: Know It All*. Morgan Kaufmann, 2008.

Vineet Chaoji, Mohammad Al Hasan, Saeed Salem, Jérémy Besson, and Mohammed J. Zaki. ORIGAMI: A Novel and Effective Approach for Mining Representative Orthogonal Graph Patterns. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 1(2):67–84, 2008.

Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling*. Chapman and Hall/CRC, 2000.

Adele Cutler and Leo Breiman. Archetypal Analysis. *Technometrics*, 36(4):338–347, 1994.

Francesco Dinuzzo and Bernhard Schölkopf. The representer theorem for hilbert spaces: A necessary and sufficient condition. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 189–196, 2012.

Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition. *SIAM Journal on Computing*, 36(1):184–206, 2006.

Alex Endert, Chao Han, Dipayan Maiti, Leanna House, Scotland Leman, and Chris North. Observation-level Interaction with Statistical Models for Visual Analytics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 121–130. IEEE, 2011.

William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge Discovery in Databases: An Overview. *AI magazine*, 13(3):57, 1992.

Jerome H. Friedman and John W. Tukey. A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions onComputers*, 100(9):881–890, 1974.

Walter Gander, Gene Golub, and Urs von Matt. A Constrained Eigenvalue Problem. *Linear Algebra and Its Applications*, 114-115:815–839, 1989.

Gemma C. Garriga, Petra Kralj, and Nada Lavrač. Closed Sets for Labeled Data. *Journal of Machine Learning Research*, 9:559–580, 2008. ISSN 1533-7928.

Henrik Grosskreutz and Daniel Paurat. Fast and Memory–efficient Discovery of the Top–k Relevant Subgroups in a Reduced Candidate Space. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer, 2011.

Henrik Grosskreutz, Stefan Rüping, and Stefan Wrobel. Tight Optimistic Estimates for Fast Subgroup Discovery. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 440–456. Springer, 2008.

Henrik Grosskreutz, Daniel Paurat, and Stefan Rüping. An Enhanced Relevance Criterion For More Concise Supervised Pattern Discovery. In *Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2012.

Jihun. Ham, Daniel D. Lee, Sebastian Mika, and Bernhard Schölkopf. A Kernel View of the Dimensionality Reduction of Manifolds. In *Proceedings of the 21st International Conference on Machine Learning*, ICML 2004, 2004.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining Frequent Patterns without Candidate Generation. In *Proceedings of the Special Interest Group on Management of Data*, pages 1–12, 2000.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.

Peter J. Huber. Projection Pursuit. *The annals of statistics*, pages 435–475, 1985.

Tomoharu Iwata, Neil Houlsby, and Zoubin Ghahramani. Active Learning for Interactive Visualization. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pages 342–350, 2013.

Ian T. Jolliffe. *Principal Component Analysis*. Springer, 1986.

Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo Approximation Algorithms for Enumeration Problems. *Journal of algorithms*, 10(3):429–448, 1989.

Willi Klösgen. Explora: A Multipattern and Multistrategy Discovery Assistant. In *Proceedings of the Conference on Advances in Knowledge Discovery and Data Mining*, pages 249–271. MIT Press, 1996.

Kleanthis-Nikolaos Kontonasios and Tijl De Bie. An Information-theoretic Approach to Finding Informative Noisy Tiles in Binary Databases. In *Proceedings of the SIAM International Conference on Data Mining*, 2010.

Joseph B. Kruskal. Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, 29(1):1–27, 1964.

Joseph B. Kruskal. Toward a Practical Method Which Helps Uncover the Structure of a Set of Multivariate Observations by Finding the Linear Transformation Which Optimizes a New 'Index of Condensation'. In *Proceedings of the Conference on Statistical Computation*, pages 427–440. Academic Press, New York, Citeseer, 1969.

Joseph B. Kruskal. Linear Transformation of Multivariate Data to Reveal Clustering. *Multidimensional Scaling: Theory and Applications in the Behavioural Sciences*, 1: 179–191, 1972.

Nada Lavrač and Dragan Gamberger. Relevancy in Constraint-Based Subgroup Discovery. In *Proceedings of Constraint-Based Mining and Inductive Databases*, pages 243–266, 2005.

Nada Lavrač, Dragan Gamberger, and Viktor Jovanoski. A Study of Relevance for Learning in Deductive Databases. *The Journal of Logic Programming*, 40(2-3):215–249, 1999.

Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupčo Todorovski. Subgroup Discovery With CN2-SD. *Journal of Machine Learning Research*, 5(Feb):153–188, 2004. ISSN 1533-7928.

Daniel D. Lee and Sebastian H. Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401(6755):788–791, 1999.

Florian Lemmerich and Martin Atzmüller. Fast Discovery of Relevant Subgroup Patterns. In *Proceedings of the Florida Artificial Intelligence Research Society*. AAAI, 2010.

Sebastian Mika, Bernhard Schölkopf, Alexander J. Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel PCA and De-Noising in Feature Spaces. In *Proceedings of the Conference on Neural Information Processing Systems*, volume 11, pages 536–542. Citeseer, Citeseer, 1998.

Shinichi Morishita and Jun Sese. Traversing Itemset Lattice with Statistical Metric Pruning. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 226–236. ACM, 2000.

Marion Neumann, Roman Garnett, Plinio Moreno, Novi Patricia, and Kristian Kersting. Propagation Kernels for Partially Labeled Graphs. In *Proceedings of the International Conference on Machine Learning Workshop on Mining and Learning with Graphs*, Edinburgh, UK, 2012. dtai.cs.kuleuven.be/events/mlg2012/.

Siegfried Nijssen, Tias Guns, and Luc De Raedt. Correlated Itemset Mining in ROC Space: A Constraint Programming Approach. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 647–656. ACM, 2009.

Dino Oglic, Daniel Paurat, and Thomas Gärtner. Interactive Knowledge–Based Kernel PCA. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer, 2014.

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information systems*, 24(1):25–46, 1999.

Daniel Paurat and Thomas Gärtner. InVis: A Tool for Interactive Visual Data Analysis. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer, 2013.

Daniel Paurat, Roman Garnett, and Thomas Gärtner. Constructing Cocktails from a Cocktail Map. In *Proceedings of the Neural Information Processing Systems 1st Workshop on Constructive Machine Learning*. http://www-kd.iai.uni-bonn.de/cml2013/, 2013a.

Daniel Paurat, Dino Oglic, and Thomas Gärtner. Supervised PCA for Interactive Data Analysis. In *Proceedings of the Neural Information Processing Systems 2nd Workshop on Spectral Learning*. Springer, 2013b.

Daniel Paurat, Roman Garnett, and Thomas Gärtner. Interactive Exploration of Larger Pattern Collections: A Case Study on a Cocktail Dataset. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2nd Workshop on Interactive Data Exploration and Analytics*. http://poloclub.gatech.edu/idea2014/, 2014.

Karl Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel Principal Component Analysis. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

Bernhard Schölkopf, Ralf Herbrich, and Alexander J. Smola. A Generalized Representer Theorem. In *Proceedings of the 14th Conference on Computational Learning Theory*. Springer, 2001.

Colin Shearer. The CRISP-DM Model: the New Blueprint for Data Mining. *Journal of data warehousing*, 5(4):13–22, 2000.

Ben Shneiderman. The Eyes Have it: A Task by Data Tpe Taxonomy for Information Visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE, IEEE, 1996.

Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

Christian Thurau, Kristian Kersting, and Christian Bauckhage. Yes We Can: Simplex Volume Maximization for Descriptive Web-scale Matrix Factorization. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1785–1788. ACM, ACM, 2010.

Warren S. Torgerson. Multidimensional Scaling of Similarity. *Psychometrika*, 30(4): 379–393, 1965.

John W. Tukey. Mathematics and the Picturing of Data. In *Proceedings of the International Congress of Mathematicians*, volume 2, pages 523–531. EMS Publishing House, 1975.

John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company, 1977.

Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases. In *Proceedings of the Conference on Discovery Science*, pages 16–31. Springer, 2004.

Laurens J. P. van der Maaten, Eric O. Postma, and Jaap H. van den Herik. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10 (1-41):66–71, 2009.

Matthijs van Leeuwen and Arno J. Knobbe. Non-redundant Subgroup Discovery in Large and Complex Data. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 459–474. Springer, 2011.

Stefan Wrobel. An Algorithm for Multi-relational Discovery of Subgroups. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 78–87. Springer, 1997.

Zhen-yue Zhang and Hong-yuan Zha. Principal Manifolds and Nonlinear Dimensionality Reduction via Tngent Space Alignment. *Journal of Shanghai University (English Edition)*, 8(4):406–424, 2004.

Appendix

A. InVis User Manual

InVis is a tool for **In**teractive **Vis**ualization of high dimensional datasets, which can be downloaded from the following location: `http://www-kd.iai.uni-bonn.de/index.php?page=software_details&id=31`. It is free for academic use and open source under the MIT-license. At the current state it covers a set of static and interactive algorithms that enable a user to explore two dimensional projections of a dataset. The following Figure A.1 shows a screenshot of the graphical user interface of the tool, without any dataset loaded. In the following sections, the menu entries and the user interface will be explained.

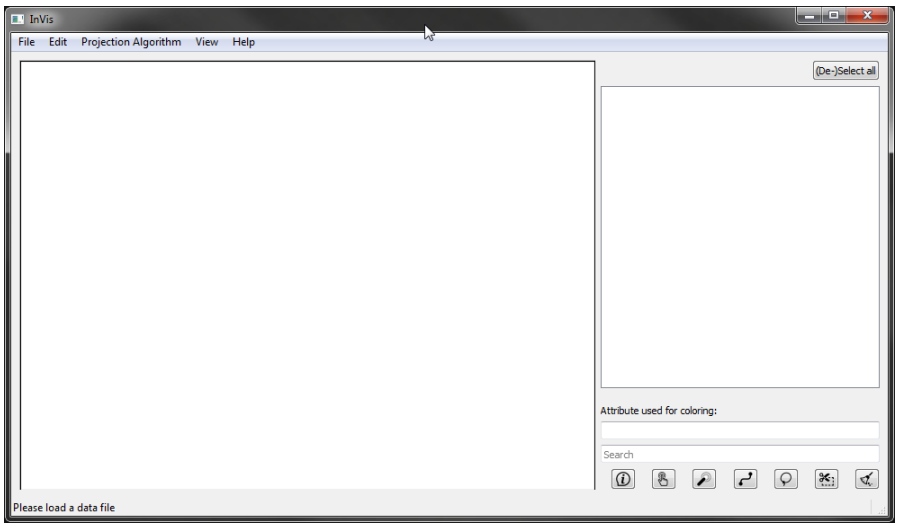


Figure A.1.: Starting up the InVis tool.

The File Menu

The File menu lets the user load a dataset and export the parameters that generate the currently viewed projection. In addition, basic implementations of four different pattern mining algorithms are available that let the user export the top-k patterns in a format that can be re-imported by the tool.

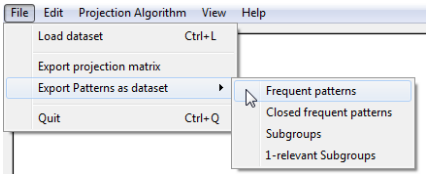


Figure A.2.: The File menu.

Loading a Dataset

In general, csv, arff and libsvm data-files can be loaded into the tool. Note, that for loading csv files, the parser is quite strict. The data is read row-wise, with the first line being a header and every subsequent row being interpreted as a data-record. The first column is considered to be a string-valued ID or name of each data record. All values of a row have to be separated by commas and only the name entry is allowed to be non-numeric. Also, by default, the last column is considered as label and the numeric entries may not be quoted (e.g. ”12.54”). The following table illustrates the csv dialect that is well understood by the tool.

Example of an accepted csv file
name,a1,a2,a3,label
Name1,1.0,2.3,2.1,100
Name2,3.0,1.3,2.3,80

Example of a not accepted csv file
missing values can be interpreted as zero
”Name1”; 1.0; 2.3; ”2.1”; A
”Name2”; 3.0; ; ”2.3”; B

Once a dataset is loaded, the tool automatically performs a principal component analysis and renders a visualization of the data, projected into the first two principal directions. The attributes of the dataset are displayed on the right hand side. In case, the user wants to ignore an attribute, he can do so by un-checking the corresponding entry.

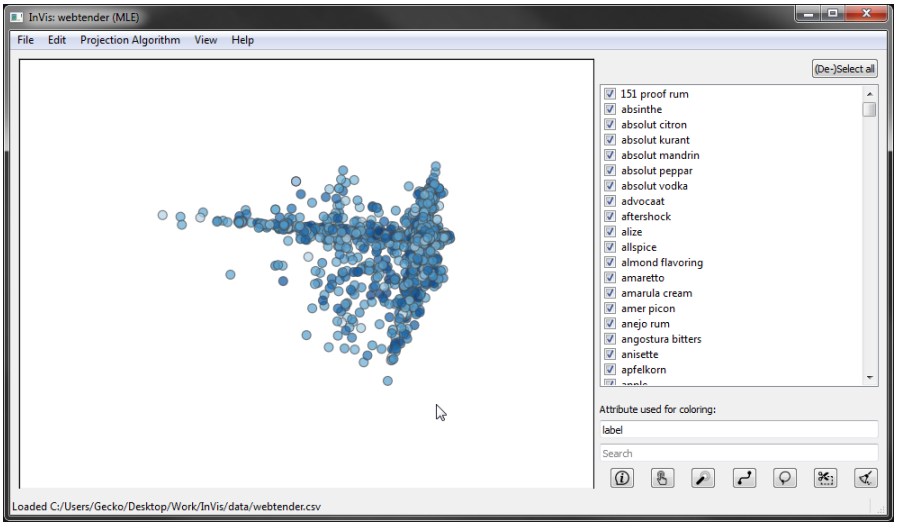


Figure A.3.: The initial view, after the webtender dataset is loaded.

The Edit Menu

This menu lets the user adjust the area (in pixel) that is considered adjacent to an embedded data record. Especially when using a touch screen, this can be a helpful option. Also the way that the numeric entries of the data records are discretized can be adjusted here. This option can be helpful, when exporting patterns from the dataset (via the Edit menu), or for displaying the ten most frequent item sets within a selected area of the embedding (see next Section). In addition, for the ease of use, when utilizing the experimental feature of must-link and cannot-link constraints, this menu offers to clear all links.

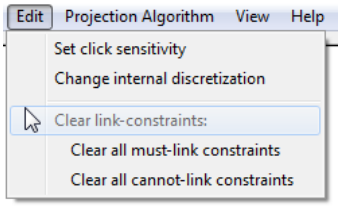
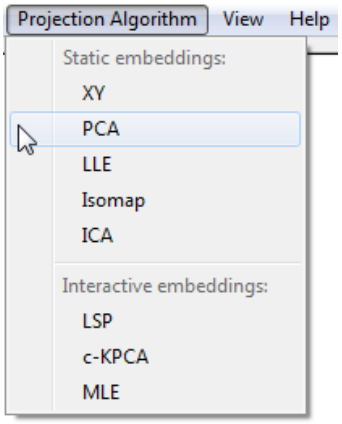


Figure A.4.: The edit menu.

The Projection Algorithm Menu

Here, the user can select the embedding algorithm that renders the visualization. The menu is sub-divided into static and interactive techniques. The first constitute a set of classic embedding methods, which have proven to be fruitful time and time again. The second set of algorithms allows the user to playfully interact with the layout embedding.



[
Available projection algorithms in InVis.]The set of algorithms that can be used by the analyst to project the data into two dimensions.

Static Embeddings

The *static embeddings* let a user have a look at the data via commonly used embedding techniques:

- XY (An xy-scatter plot of the two first selected attributes)
- PCA (Princial Component Analysis)
- LLE (Locally Linear Embedding)
- Isomap (Isometric mapping: basically, multidimensional scaling applied to the knn-graph)
- ICA (Independent Component Analysis)

Note that while in a static embedding, the user can still interact with the visualization via searching, highlighting and filtering, etc.. The user can also set and un-set control points, however, since the embeddings are static, he cannot relocate them.

Interactive Embeddings

The *interactive embeddings* let the user actively layout and shape the projection of the data by selecting and re-location individual data records within the embedding as control points. Relocating these control points triggers the underlying embedding algorithm to re-calculate the projection with respect to the user provided feedback. The result is rendered instantly, which yields a life updating visualization. The three interactive embedding techniques implemented in this tool are:

- LSP (Least Squared Error Projection)

- cKPCA (Constrained Kernel Principal Component Analysis)
- MLE (Most Likely Embedding)

The View Menu

This menu lets the user control the look and feel of the visualized data. The adjustments can be made in the following way: The first four entries let the analyst chose the color scheme in which the data records are highlighted; the default is a blue scale. In addition, the point size for each data record can be set proportional to the considered label value. This can e.g. be of use when studying a dataset of patterns, with the label being their support.

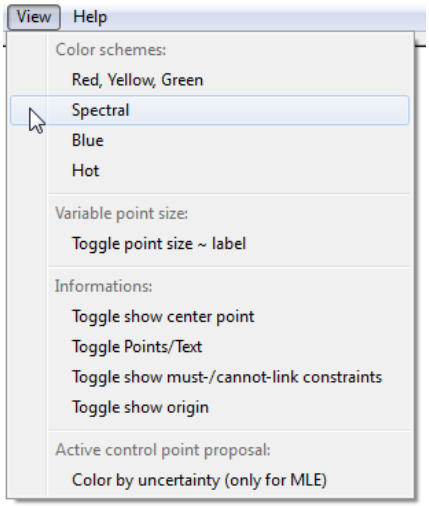


Figure A.5.: Options that can be adjusted in the view menu.

The third section lets the user toggle the visibility of various elements which may be of help during an analysis session. The last option is an experimental feature which is only available for the MLE algorithm. A side product that can easily be calculated by this probabilistic algorithm, is the confidence about the location of each data record within the embedding. This confidence is then used to colorize the visualization. In one of the following sections, a "magic wand" button is introduced that auto-selects good control points for the MLE method. The magic behind the selection procedure uses exactly this confidence value.

The Help Menu

A survey of the most commonly used interaction methods with the visualization and their keyboard shortcuts. In addition, this document is displayed on pressing F1.

- **Left-click & drag** lets the user re-locate the nearest control-point in a "drag 'n drop" like manner.
- **Right-click** displays information of the clicked data-record (e.g: attribute_name:values>0)
- **Middle-click** lets the user select or de-select a data-record as control-point.
- **Mouse-wheel** lets the user zoom in and out on the mouse pointer.
- **Ctrl+left-click-lasso-select** lets the user select all data records in a region of the embedding.

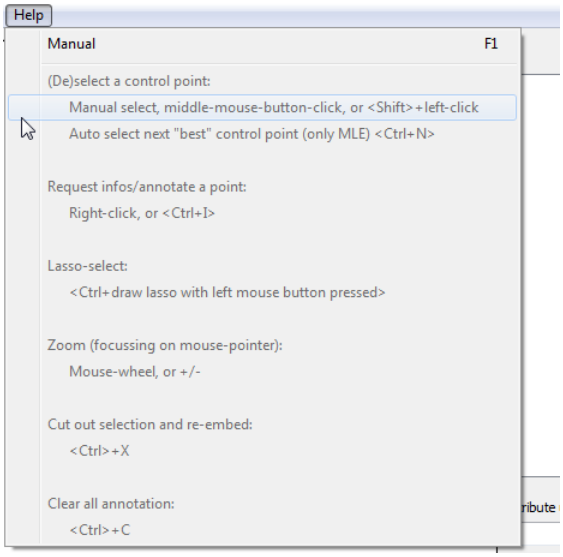


Figure A.6.: A quick reminder of the shortcuts for interaction with the canvas.

Interaction

When using the InVis tool in combination with a touch screen, a keyboard might be disturbing. For this scenario (and for shortcut lazy users) all interaction methods are also available via the graphical user interface. Note that the buttons change their appearance to a colored version once they are active. The buttons meanings are the following:

ⓘ, ⓘ Query information on an individual data record, by clicking on it.

, Selecting a data record as control point.

, "Magic wand" control point selection (only available for MLE).

, , Introduce must- and cannot-link constraints data record pairs.

, Lasso-select all data records within a region in the embedding.

, Consider only the lasso-selected data records.

Clear all search annotations and information queries.

When querying information on a single data record, usually the attribute values of that data record are of interest. For a high dimensional dataset this can quickly get out of hand. For this reason, here only the non-zero entries are displayed.

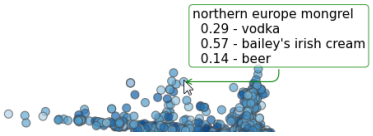


Figure A.7.: Queried information on a single data record.

The selected control points are highlighted with a pink bold border. This way they are easy to distinguish from the regular data records.

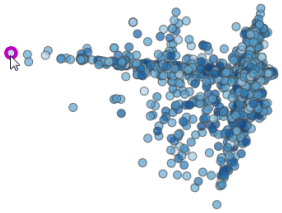


Figure A.8.: A control point.

When lasso-selecting an area, the enclosed data records are emphasized and a word cloud of the ten most frequent attribute sets is displayed in the bottom left corner. This helps the user to quickly grasp the dominant attributes and attribute combinations within the region of interest.

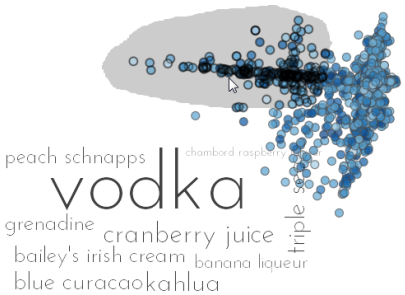


Figure A.9.: A lasso-selected area and its most influential attribute combinations.

Highlighting and Searching

A user can also search for a sub-string that is contained in the data-records name. This can be done by using the free text fields at the bottom right corner of the user interface. The matching results will be highlighted in red. Here, the search term “bloody” reveals the embedding locations of all data records that possess this term as part of their name (e.g. the bloody marry).

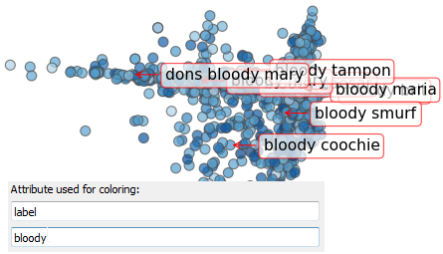


Figure A.10.: Searching parts of the data record ID’s.

The second free text field offers the user to enter an attribute name, by which the data points are colored. The value of the attribute determines the shade of the color. For the webtender dataset, the first principal direction coincides heavily with the attribute *vodka*.

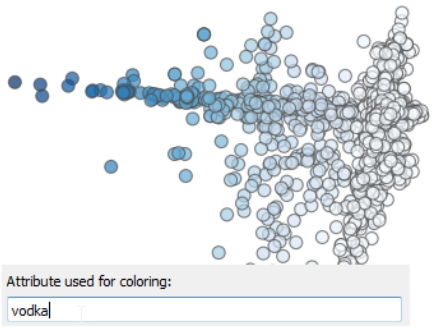


Figure A.11.: Colorizing the data records by an attribute value.