Department of Computer Science
Series of Publications A
Report A-2005-1

Summarization Techniques for Pattern Collections in Data Mining

Taneli Mielikäinen

Academic Dissertation

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium B123, Exactum, on May 27th, 2005, at noon.

University of Helsinki
Finland

ISSN 1238-8645
ISBN 952-10-2436-4 (paperback)
ISBN 952-10-2437-2 (PDF)
http://ethesis.helsinki.fi/

Computing Reviews (1998) Classification: E.4, H.2.8, I.2, I.2.4

Helsinki University Printing House
Helsinki, April 2005 (201 pages)

Summarization Techniques for Pattern Collections in Data Mining

Taneli Mielikäinen
Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
Taneli.Mielikainen@iki.fi
http://www.iki.fi/Taneli.Mielikainen/

Abstract

Discovering patterns from data is an important task in data mining. There exist techniques to find large collections of many kinds of patterns from data very efficiently. A collection of patterns can be regarded as a summary of the data. A major difficulty with patterns is that pattern collections summarizing the data well are often very large.

In this dissertation we describe methods for summarizing pattern collections in order to make them also more understandable. More specifically, we focus on the following themes:

Quality value simplifications. We study simplifications of pattern collections based on simplifying the quality values of the patterns. Especially, we study simplification by discretization.

Pattern orderings. It is difficult to find a suitable trade-off between the accuracy of the representation and its size. As a solution to this problem, we suggest that patterns could be ordered in such a way that each prefix of the *pattern ordering* gives a good summary of the whole collection.

Pattern chains and antichains. Virtually all pattern collections have natural underlying partial orders. We exploit the partial orders over pattern collections by clustering the patterns into chains and antichains.

Change profiles. We describe how patterns can be related to each other by comparing how their quality values change with re-

spect to their common neighborhoods, i.e., by comparing their *change profiles*.

Inverse pattern discovery. As the patterns are often used to summarize data, it is natural to ask whether the original data set can be deduced from the pattern collection. We study the computational complexity of such problems.

Computing Reviews (1998) Categories and Subject Descriptors:

E.4	Coding and Information Theory: Data Compaction and Compression
H.2.8	Database Applications: Data Mining
I.2	Artificial Intelligence
I.2.4	Knowledge Representation Formalisms and Methods

General Terms: Algorithms, Theory, Experimentation

Additional Key Words and Phrases: Pattern Discovery, Condensed Representations of Pattern Collections, Post-Processing of Data Mining Results

Acknowledgements

I am most grateful to my supervisors Heikki Mannila and Esko Ukkonen for their tireless guidance, patience and encouragement throughout my studies. Their insightful comments were most useful to help me improve the thesis manuscript considerably. I wish to thank also Jean-François Boulicaut and Dimitrios Gunopulos for serving as the reviewers of the manuscript of the thesis and for their helpful comments.

The studies have been carried out at the Department of Computer Science of the University of Helsinki that has provided me an excellent working environment. I wish to thank especially the computing facilities staff of the department for ensuring the fluent operation of the computing facilities and for their endurance to help me with my numerous problems. Financial support by Helsinki Graduate School for Computer Science and Engineering, From Data to Knowledge research unit, and HIIT Basic Research Unit are gratefully acknowledged.

Many friends and colleagues have provided me invaluable support. Especially Floris Geerts, Bart Goethals, Matti Kääriäinen, Ari Rantanen and Janne Ravantti have hopefully affected the dissertation considerably by their insightful comments and refreshing discussions about the topic. Also the intellectual support by Tapio Elomaa, Patrik Floréen and Hannu Toivonen were most valuable.

I am most indebted to my parents Marketta and Kari for their support and encouragement. Their frequent queries about my studies and the state of the dissertation were most helpful. The cultural support provided by Antti Nissinen is greatly appreciated. My deepest love and gratitude belong to Satu, Iida and Lauri. Their unconditional support and love were vital also for this work.

Contents

CHAPTER 1

Introduction

> "But what kind of authentic and valuable information do you require?" asked Klapaucius.
>
> "All kinds, as long as it's true", replied the pirate. "You never can tell what facts may come in handy. I already have a few hundred wells and cellars full of them, but there's room for twice again as much. So out with it; tell me everything you know, and I'll jot it down. But make it snappy!"
>
> *Stanislaw Lem: The Cyberiad (1974)*

Mankind has achieved an impressive ability to store data [Rie03]. The capacity of digital data storage has doubled every nine months for at least a decade [FU02]. Furthermore, our skills and interest to collect data are also remarkable [LV03].

Our ability to process the collected data is not so impressive. In fact, there is a real danger that we construct *write-only data stores* that cannot be exploited using current technologies [FU02]. Besides constructing *data tombs* that contain snapshots of our world for the tomb raiders of the forthcoming generations, this is not very useful. It can be said that we are in a *data rich but information poor* situation [HK01].

In addition to the immense amount of data being collected, the data is becoming increasingly complex and diverse [Fay01, SPF02]: companies collect data about their customers to maximize their expected profit [KRS02], scientists gather large repositories of observations to better understand nature [HAK+02] and governments

of many countries are collecting vast amounts of data to ensure the homeland security which has been recognized to be a very important issue due to the globalization of conflicts and terrorism [Yen04]. When several different data repositories are combined, the data concerning even only a single person can be tremendously large and complex.

Due to the weakness of the current techniques to exploit large data repositories and the complexity of the data being collected, a new discipline known as *data mining* is emerging in the intersection of artificial intelligence, databases and statistics. The current working definition of this new field is the following [HMS01]:

> Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.

On one hand this definition is acceptable for a large variety of data mining scholars. On the other hand its interpretation depends on several imprecise concepts: The meanings of the words 'unsuspected', 'understandable' and 'useful' depend on the context. Also the words 'relationships' and 'summarize' have vast number of different interpretations. This indeterminacy in general seems to be inherent to data mining since the actual goal is in practice determined by the task at hand.

Albeit the inherent vagueness of the definition, the field of data mining can be elucidated by arranging the techniques to groups of similar approaches. The techniques can be divided roughly to two parts, namely to *global* and *local* methods.

Global methods concern constructing and manipulating *global models* that describe the entire data. Global models comprise most of the classical statistical methods. For example, the Gaussian distribution function is a particularly well-known global model for real-valued data. The focus in the data mining research of global methods has been on developing and scaling up global modeling techniques to very large data sets.

Local methods focus on *discovering patterns* from data. Patterns are parsimonious summaries of subsets of data [FU02]. The rule "People who buy diapers tend to buy beer" is a classical example

of such pattern. In contrast to global modeling approach, pattern discovery as a discipline in its own right is relatively new [Han02]. (The term 'discovery' has recently been criticized in the context of data mining to be misleading since data mining is based on scientific principles and it can be argued that science does not discover facts by induction but rather invents theories that are then checked against experience [PB02]. The term is used, however, in this dissertation because of its established use in data mining literature.)

The global and local methods can be summarized in the following way. The global modeling approach views data mining as the task of *approximating the joint probability distribution* whereas the pattern discovery can be summarized in the slogan: *data mining is the technology of fast counting* [Man02].

The distinction to global models and local patterns is not strict. Although a Gaussian distribution is usually considered as a global model, it can be also a pattern: each Gaussian distribution in a mixture of Gaussians is assumed to describe only a part of the data.

This work focuses on pattern discovery. There exist effective techniques to discover many kinds of patterns [GZ03, MT97]. Due to that fact the question of how the discovered patterns could actually be exploited is becoming increasingly important. Often the answer to that question is tightly coupled with the particular application. Many problems, obstacles and characteristics, however, are shared with different applications.

A very important application of patterns is to summarize given data as a collection of patterns, possibly augmented with some auxiliary information such as the quality values of the patterns. Unfortunately, often the size of the pattern collection that faithfully represents the aspects of the data considered to be relevant is very large. Thus, in addition to data tombs, there is a risk of constructing also *pattern tombs*.

1.1 The Contributions and the Organization

The main purpose of this dissertation is to study how to summarize pattern collections by exploiting the structure of the collections and the quality values of the patterns. The rest of the dissertation is

organized as follows.

Chapter 2 provides an introduction to pattern discovery that is sufficient to follow the rest of the dissertation. It contains a systematic derivation of a general framework for pattern discovery, a brief overview of the current state of pattern discovery and descriptions of the most important (condensed) representations of pattern collections. Furthermore, some technical challenges of pattern exploitation are briefly discussed.

Chapter 3 concerns simplifying pattern collections by simplifying the quality values of the patterns. The only assumption needed about the pattern collection is that there is a quality value associated to each pattern.

We illustrate the idea of constraining the quality values of the patterns by discretizing the frequencies of frequent itemsets. We examine the effect of discretizing frequencies to the accuracies of association rules and propose algorithms for computing optimal discretizations with respect to several loss functions. We show empirically that discretizations with quite small errors can reduce the representation of the pattern collection considerably.

Chapter 4 focuses on trade-offs between the size of the pattern collection and its accuracy to describe the data. The chapter suggests to order the patterns by their abilities to describe the whole pattern collection with respect to a given loss function and an estimation method. The obtained ordering is a refining description of the pattern collection and it requires only a loss function and an estimation method.

We show that for several pairs of loss functions and estimation methods, the most informative k-subcollection of the patterns can be approximated within a constant factor by the k-prefix of the pattern ordering for all values of k *simultaneously*. We illustrate the pattern orderings by refining approximations closed itemsets and tilings of transaction databases. We evaluate the condensation abilities of the pattern orderings empirically by computing refining approximations of closed frequent itemsets. The results show that already short pre-

fixes of the orderings of the frequent itemsets are sufficient to provide reasonably accurate approximations.

Chapter 5 is motivated by the fact that a pattern collection has usually some structure apart from the quality values of the patterns. Virtually all pattern collections have non-trivial partial orders over the patterns. In this chapter we suggest the use of minimum chain and antichain partitions of partially ordered pattern collections to figure out the essence of a given pattern collection.

For an arbitrary pattern collection, its chain and antichain partitions provide clusterings of the collection. The benefit from the chain partition can be even greater: for many known pattern collections, each chain in the partition can be described as a single pattern. The chain partitions give a partially negative answer to the question whether a random sample of the data is essentially the best one can hope. We evaluate empirically the ability of pattern chains to condense pattern collections in the case of closed frequent itemset collections.

Chapter 6 introduces a novel approach to relate patterns in a pattern collection to each other: patterns are considered similar if their change profiles are similar, i.e., if their quality values change similarly with respect to their common neighbors in a given neighborhood relation. This can be seen as an attempt to bridge the gap between local and global descriptions of the data.

A natural way of using similarities is the clustering of patterns. Unfortunately, clustering based on change profiles turns out to be computationally very difficult. Because of that, we discuss advantages and disadvantages of different heuristic approaches to cluster patterns using change profiles. Furthermore, we demonstrate that change profiles can determine meaningful (hierarchical) clusterings. In addition to examining the suitability of change profiles for comparing patterns, we propose two algorithms for estimating the quality values of the patterns from their approximate change profiles. To see how the approximate change profiles affect the estimation

of the quality values of the patterns, the stability of the frequency estimates of the frequent itemsets is empirically evaluated with respect to different kinds of noise.

Chapter 7 studies the problems of inverse pattern discovery, i.e., finding data that could have generated the patterns. In particular, the main task considered in the chapter is to decide whether there exists a database that has the correct frequencies for a given itemset collection. This question is relevant in, e.g., privacy-preserving data mining, in quality evaluation of pattern collections, and in inductive databases. We show that many variants of the problem are NP-hard but some non-trivial special cases have polynomial-time algorithms.

Chapter 8 concludes this dissertation.

CHAPTER 2

Pattern Discovery

This chapter provides an introduction to pattern discovery, one of the two main sub-disciplines of data mining, and its central concepts that are used through and through this dissertation. A general framework is derived for pattern discovery, the most important condensed representations of pattern collections are introduced and the purpose of patterns in shortly discussed.

2.1 The Pattern Discovery Problem

The goal in pattern discovery is to find interesting patterns from given data [Han02, Man02]. The task can be defined more formally as follows:

Problem 2.1 (pattern discovery). Given a class $\mathcal{P}$ of patterns and an interestingness predicate $q : \mathcal{P} \to \{0, 1\}$ for the pattern class, find the collection

$$\mathcal{P}_q = \{p \in \mathcal{P} : q(p) = 1\}$$

of *interesting* patterns. Its complement $\mathcal{P}_{\bar{q}} = \mathcal{P} \setminus \mathcal{P}_q$ is called the collection of *uninteresting patterns* in $\mathcal{P}$ with respect to q.

The pattern discovery problem as defined above consists of only two parts: the collection $\mathcal{P}$ of possibly interesting patterns and the interestingness predicate q.

The pattern collection $\mathcal{P}$ constitutes *a priori* assumptions of which patterns could be of interest. The collection $\mathcal{P}$ is usually

not represented explicitly since its cardinality can be very large, sometimes even infinite. For example, the collection of patterns could consist of all regular expressions over a given alphabet Σ. (For an introduction to regular expressions, see e.g. [HMU01].) This collection is infinite even for the unary alphabet.

The absence of data from the definition might be a bit confusing at first. It is omitted on purpose: Often the interestingness predicate depends on data and the data is usually given as a parameter for the predicate. This is not true in every case, however, since the *interestingness* (or, alternatively, the *quality*) of a pattern can be determined by an expert who has specialized to some particular data set and the interestingness predicate might be useless for any other data set regardless of its form. For example, a company offering industrial espionage that is specialized to investigate power plants can be rather poor detecting interesting patterns from gardening data.

Defining a reasonable interestingness predicate is usually a highly non-trivial task: the interestingness predicate should capture most truly interesting patterns and only few uninteresting ones.

Due to these difficulties, a relaxation of an interestingness predicate, an *interestingness measure*

$$\phi : \mathcal{P} \to [0, 1]$$

expressing the quantitative value $\phi(p)$ of the interestingness (or the quality) for each pattern $p \in \mathcal{P}$ is used instead of an interestingness predicate. In this dissertation the value $\phi(p)$ of $p \in \mathcal{P}$ is called the *quality* value of p with respect to the interestingness measure ϕ, or in short: the quality of p. Many kinds of interestingness measures have been studied in the literature, see e.g. [TKS02].

Example 2.1 (an interestingness measure). Let the data set consist of names of recently born children and their ages (that are assumed to be strictly positive), i.e., let $\mathcal{D}$ be a set of pairs $\langle name, age \rangle \in \Sigma^* \times \mathbb{R}_+$.

An interestingness measure ϕ for the pattern class $\mathcal{P}_{\text{regexp}}$ consisting of all regular expressions could be defined as follows. Let $\mathcal{D}|_p$ be the group of children whose names satisfy the regular expression $p \in \mathcal{P}_{\text{regexp}}$. The quality of a pattern $p \in \mathcal{P}_{\text{regexp}}$ is the smallest age of any child in $\mathcal{D}$ divided by the average ages the children whose

names belong to the regular language p, i.e.,

$$\phi(p, \mathcal{D}) = \frac{\min\{age : \langle name, age\rangle \in \mathcal{D}\}}{\left(\sum_{\langle name, age\rangle \in \mathcal{D}|_p} age\right) / |\{age : \langle name, age\rangle \in \mathcal{D}|_p\}|}.$$

□

There are many reasons why interestingness measures are favored over interestingness predicates. An important reason is that it is often easier to suggest some degrees of interestingness for the patterns in the given collection than to partition the patterns into the groups of strictly interesting and uninteresting ones. In fact, using an interestingness measure, instead of an interestingness predicate, partially postpones the difficulty of fixing a suitable interestingness predicate, since an interestingness measure implicitly determines an infinite number of interestingness predicates:

$$q(p) = \begin{cases} 1 & \text{if } \phi(p) \geq \sigma \\ 0 & \text{otherwise.} \end{cases}$$

In addition to these practical reasons, there are also some more foundational arguments that support the use of interestingness measures instead of predicates. Namely, it can be argued that the actual goal in pattern discovery is not merely to find a collection of interesting patterns but to rank the patterns with respect to their quality values [Mie04a]. Also, due to the exploratory nature of data mining, it might not be wise to completely discard the patterns that seem to be uninteresting, since you never can tell what patterns may come in handy. Instead, it could be more useful just to list the pattern in decreasing order with respect to their quality values.

Also the interestingness predicates can be defended against the interestingness measures. The interestingness predicates determine collections of patterns whereas the interestingness measures determine rankings (or gradings). On one hand, the interestingness predicates can be manipulated and combined by boolean connectives. Furthermore, the manipulations have direct correspondents in the pattern collections. Combining rankings corresponding to interestingness measures, on the other hand, is not so straightforward.

Thus, the interestingness predicates and the interestingness measures have both strong and weak points. Due to this, the majority

of pattern discovery research has been focused on the combination of interestingness measures and predicates: they consider discovering collections of interesting patterns augmented by their quality values.

2.2 Frequent Itemsets and Association Rules

The most prominent example of pattern discovery is discovering (or mining) *frequent itemsets* from *transaction databases* [AIS93, Man02].

Definition 2.1 (items and itemsets). A set of possible *items* is denoted by $\mathcal{I}$. An *itemset* X is a subset of $\mathcal{I}$. For brevity, an itemset X consisting items $A_1, A_2, \ldots, A_{|X|}$ can be written $A_1 A_2 \ldots A_{|X|}$ instead of $\{A_1, A_2, \ldots, A_{|X|}\}$.

Definition 2.2 (transactions and transaction databases). A *transaction* t is a pair $\langle i, X \rangle$ where i is a *transaction identifier (tid)* and X is an itemset. The number of items in the itemset X of a transaction $t = \langle i, X \rangle$ is denoted by $|t|$.

A *transaction database* $\mathcal{D}$ is a set of transactions. Each transaction in $\mathcal{D}$ has a unique transaction identifier. The number of transactions in the transaction database $\mathcal{D}$ is denoted by $|\mathcal{D}|$ and the set of transaction identifiers in $\mathcal{D}$ by $tid(\mathcal{D}) = \{i : \langle i, X \rangle \in \mathcal{D}\}$. In the context of this dissertation it is assumed, without loss of generality, that $tid(\mathcal{D}) = \{1, \ldots, |\mathcal{D}|\}$.

The set of occurrences of an itemset X in $\mathcal{D}$ is the set

$$occ(X, \mathcal{D}) = \{i : \langle i, X \rangle \in \mathcal{D}\}$$

of transaction identifiers of the transactions $\langle i, X \rangle \in \mathcal{D}$. The number of occurrences of X in $\mathcal{D}$ is denoted by $count(X, \mathcal{D}) = |occ(X, \mathcal{D})|$.

Another important aspect for frequent itemsets is the definition of what it means that an itemset is frequent with respect to a transaction database.

Definition 2.3 (covers, supports and frequencies). A transaction $t = \langle i, Y \rangle$ in a transaction database $\mathcal{D}$ is said to *cover* or

support an itemset X if $X \subseteq Y$. The cover of an itemset X in $\mathcal{D}$ is the set

$$cover(X, \mathcal{D}) = \{i : \langle i, Y \rangle \in \mathcal{D}, X \subseteq Y\}$$

of transaction identifiers of the transactions in $\mathcal{D}$ that *cover* X. The *support* of X in $\mathcal{D}$ is denoted by $supp(X, \mathcal{D})$ and it is equal to the cardinality of the *cover* of X in $\mathcal{D}$, i.e.,

$$supp(X, \mathcal{D}) = |cover(X, \mathcal{D})| .$$

The frequency of X in $\mathcal{D}$ is its support divided by the number of transactions in $\mathcal{D}$, i.e.,

$$fr(X, \mathcal{D}) = \frac{supp(X, \mathcal{D})}{|\mathcal{D}|}.$$

The database $\mathcal{D}$ can be omitted from the parameters of these functions when $\mathcal{D}$ is not known or needed. If there are several itemset collections $\mathcal{F}_1, \ldots, \mathcal{F}_m$ with different covers, supports or frequencies, we denote the cover, the support and the frequency of an itemset X in the collection $\mathcal{F}_i$ $(1 \leq i \leq m)$ by $cover(X, \mathcal{F}_i)$, $supp(X, \mathcal{F}_i)$ and $fr(X, \mathcal{F}_i)$, respectively.

Based on these definitions, the frequent itemset mining problem can be formulated as follows:

Problem 2.2 (frequent itemset mining [AIS93]). Given a transaction database $\mathcal{D}$ and a minimum frequency threshold $\sigma \in (0, 1]$, find all σ-frequent itemsets in $\mathcal{D}$, i.e., all itemsets such that $fr(X, \mathcal{D}) \geq \sigma$. The collection of σ-frequent itemsets is denoted by $\mathcal{F}(\sigma, \mathcal{D})$.

Example 2.2 (frequent itemsets). Let the transaction database $\mathcal{D}$ consist of transactions $\langle 1, ABC \rangle$, $\langle 2, AB \rangle$, $\langle 3, ABCD \rangle$ and $\langle 4, BC \rangle$. Then the frequencies of itemsets in $\mathcal{D}$ are as shown in Table 2.1. For example, the collection $\mathcal{F}(2/4, \mathcal{D})$ of 2/4-frequent itemsets in $\mathcal{D}$ is $\{\emptyset, A, B, C, AB, AC, BC, ABC\}$.

□

Probably the most well-known example of frequent itemset mining tasks is the *market basket analysis*. In that case the items are products available for sale. Each transaction consists of a transaction identifier and a subset of the products that typically corresponds to items bought in a single purchase, i.e., the transactions

Table 2.1: Itemsets and their frequencies in $\mathcal{D}$.

X	$fr(X, \mathcal{D})$
$\emptyset$	1
A	3/4
B	1
C	3/4
AB	3/4
AC	2/4
BC	3/4
ABC	2/4
$ABCD$	1/4

are market baskets. (Alternatively each transaction can correspond to all items bought by a single customer, possibly as several shopping events.) Thus, frequent itemsets are the sets of products that people tend to buy together as a single purchase event.

The frequent itemsets are useful also in text mining. An important representation of text documents is the so-called bag-of-words model where a document is represented as a set of stemmed words occurring in the document. Thus, items correspond to the stemmed words and each document is a transaction. The frequent itemsets are the sets of stemmed words that occur frequently together in the documents of the document collection.

Web mining is yet another application of frequent itemsets. There each item could be, for example, a link pointing at (from) a certain web page and each transaction could the correspond to the links pointing from (at) a web page. Then the frequent itemsets correspond to groups of web pages that are referred concurrently by (that refer concurrently) the same web pages.

2.2.1 Real Transaction Databases

The purpose of data mining is to analyze data. Without data there is not much data mining. Also the methods described in this dissertation are demonstrated using real data and the patterns discovered from the data. More specifically, in this dissertation, we use the (frequent) itemsets mined from three transaction databases as running examples of pattern collections (of interesting patterns).

The two main reasons for this are that many data analysis tasks can be modeled as frequent itemset mining and frequent itemset mining has been studied very actively for more than a decade.

We use a course completion database of the computer science students at the University of Helsinki to illustrate the methods described in this dissertation. Each transaction in that database corresponds to a student and items in a transaction correspond to the courses the student has passed. As data cleaning, we removed from the database the transactions corresponding to students without any passed courses in computer science. The cleaned database consists of 2405 transactions corresponding to students and 5021 different items corresponding to courses.

The number of students that have passed certain number of courses and the the number of students passed each course are shown in Figure 2.1. The courses that at least a 0.20-fraction of the student in the course completion database have passed (i.e., the 34 most popular courses) are shown as Table 2.2. The 0.20-frequent itemsets in the course completion database are illustrated by Example 2.3.

Example 2.3 (0.20**-frequent itemsets in the course completion database).** Let us denote by $\mathcal{F}(\sigma, \mathcal{D})[i]$ the σ-frequent itemsets in $\mathcal{D}$ with cardinality i. Then the cardinality distributions of the 0.20-frequent itemsets in the course completion database and the most frequent itemsets of each cardinality in that collection are as shown in Table 2.3.

□

The condensation approaches described in Chapters 3–6 are quantitatively evaluated using two data sets from UCI KDD Repository (`http://kdd.ics.uci.edu/`): Internet Usage data consisting of 10104 transactions and 10674 items, and IPUMS Census data consisting of 88443 transactions and 39954 items.

The transaction database Internet Usage is an example of dense transaction databases and the transaction database IPUMS Census is a sparse one: in the Internet Usage database only few σ-frequent itemsets are contained exactly in the same transactions whereas in the IPUMS Census databases many σ-frequent itemsets are contained in exactly the same transactions. (This holds for many different values of σ). This means also that most of the frequent

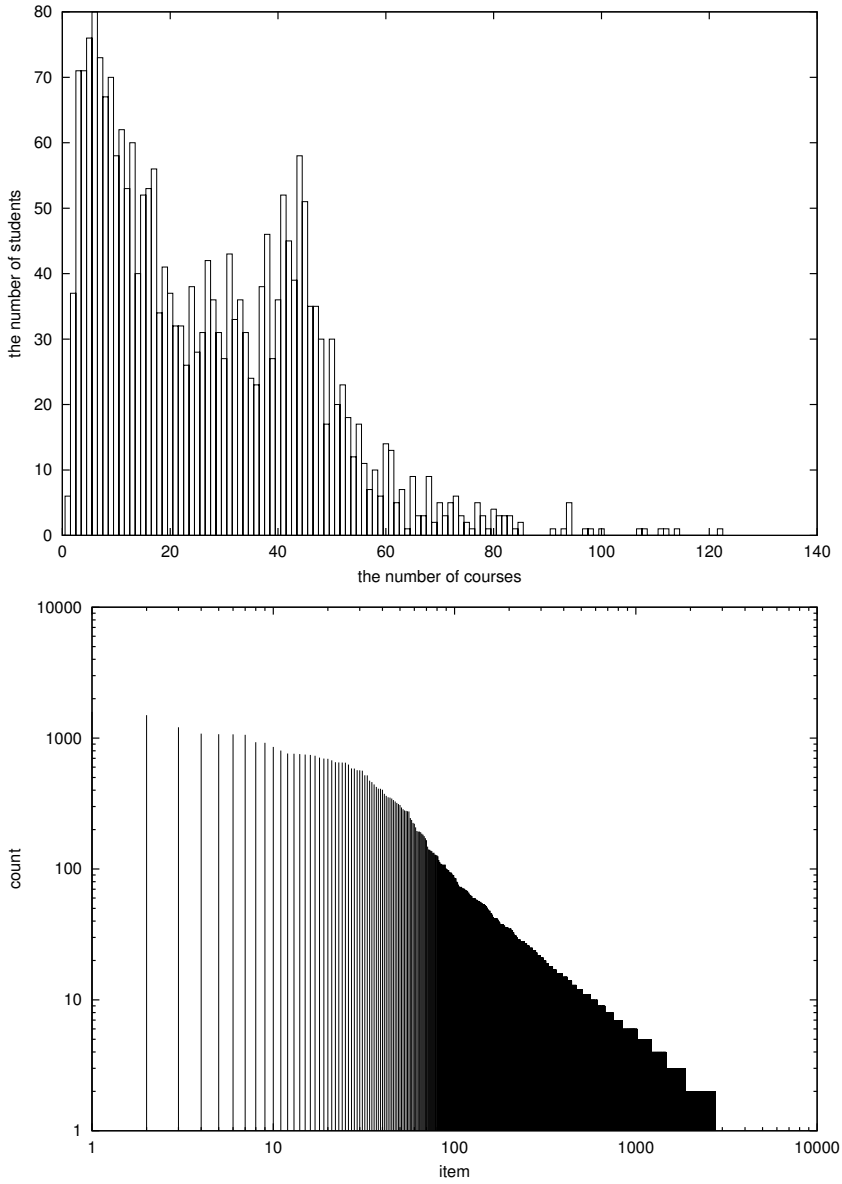

Figure 2.1: The number of transactions of different cardinalities (top) and the item counts in the course completion database (bottom).

Table 2.2: The courses in the course completion database that at least a 0.20-fraction of the students in the database has passed. The columns are the rank of the course with respect to the support, the number of students that have passed the course, the official course code and the name of the course, respectively.

rank	count	code	name
0	2076	50001	Orientation Studies
1	1587	99270	Reading Comprehension in English
2	1498	58160	Programming Project
3	1210	58123	Computer Organization
4	1081	58128	Introduction to UNIX
5	1071	58125	Information Systems
6	1069	58131	Data Structures
7	1060	58161	Data Structures Project
8	931	99280	English Oral Test
9	920	58127	Programming in C
10	856	58122	Programming (Pascal)
11	803	99291	Oral and Written Skills in the Second Official Language, Swedish
12	763	58162	Information Systems Project
13	760	58132	Concurrent Systems
14	755	58110	Scientific Writing
15	748	58038	Database Systems I
16	744	57031	Approbatur in Mathematics I
17	733	581259	Software Engineering
18	709	57019	Discrete Mathematics I
19	697	581330	Models for Programming and Computing
20	695	50028	Maturity Test in Finnish
21	677	581327	Introduction to Application Design
22	655	581326	Programming in Java
23	651	581328	Introduction to Databases
24	650	581325	Introduction to Programming
25	649	581256	Teacher Tutoring
26	628	57013	Linear Algebra I
27	586	57274	Logic I
28	585	580212	Introduction to Computing
29	568	581324	Introduction to the Use of Computers
30	567	58069	Data Communications
31	564	581260	Software Engineering Project
32	520	57032	Approbatur in Mathematics II
33	519	581329	Database Application Project

Table 2.3: The number of the 0.20-frequent itemsets of each cardinality in the course completion database, the most frequent itemsets of each cardinality and their supports.

i	$\lvert\mathcal{F}(\sigma,\mathcal{D})[i]\rvert$	the largest $X \in \mathcal{F}(\sigma,\mathcal{D})[i]$	$supp(X,\mathcal{D})$
0	1	$\emptyset$	2405
1	34	$\{0\}$	2076
2	188	$\{0,1\}$	1345
3	474	$\{2,3,5\}$	960
4	717	$\{0,2,3,5\}$	849
5	626	$\{0,2,3,4,5\}$	681
6	299	$\{0,2,3,5,7,12\}$	588
7	72	$\{2,3,5,7,12,13,15\}$	547
8	8	$\{0,2,3,5,7,12,13,15\}$	512

itemsets in Internet Usage are closed whereas most of the frequent itemsets in IPUMS Census are not. (See Definition 2.10 for more details on itemsets being closed.)

2.2.2 Computing Frequent Itemsets

The frequent itemset mining problem has been studied extensively for more than a decade and several efficient search strategies have been developed, see e.g. [AIS93, AMS+96, GZ03, HPYM04, Zak00]. Most of the techniques follow the generate-and-test approach: the collection of frequent itemsets is initialized to consist of the empty itemset with support equal to the number of transactions in the database. (This is due to the fact that the empty itemset is contained in each transaction which means also that its frequency is one.) Then the collections of itemsets that might be frequent are generated and tested repeatedly until it is decided that there are no more itemsets that are not tested but could still be frequent. The most important property of frequent itemsets for search space pruning and candidate generation is the anti-monotonicity of the supports with respect to the set inclusion relation.

Observation 2.1. *If $X \subseteq Y$, then $supp(X,\mathcal{D}) \geq supp(Y,\mathcal{D})$. Thus, all subitemsets of frequent itemsets are frequent and all superitemsets of infrequent itemsets are infrequent.*

This observation is largely responsible for the computational feasibility of the famous frequent itemset mining algorithm Apriori [AMS+96] in practice. It or some of its variant is extensively used in virtually all frequent itemset mining methods.

2.2.3 Association Rules

The itemsets that are frequent in the database are itself summaries of the database but they can be considered also as side-products of finding association rules.

Definition 2.4 (Association rules). Let $\mathcal{D}$ be a transaction database. An *association rule* is an implication of form $X \Rightarrow Y$ such that $X, Y \subseteq \mathcal{I}$. The itemset X is called the *body* (or the *antecedent*) of the rule and the itemset Y is known as the *head* (or the *consequent*) of the rule.

The *accuracy* of the association rule $X \Rightarrow Y$ is denoted by

$$acc(X \Rightarrow Y, \mathcal{D}) = \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})},$$

its support $supp(X \Rightarrow Y, \mathcal{D})$ is equal to $supp(X \cup Y, \mathcal{D})$ and the frequency of the association rule $X \Rightarrow Y$ is

$$fr(X \Rightarrow Y, \mathcal{D}) = \frac{supp(X \Rightarrow Y, \mathcal{D})}{|\mathcal{D}|}.$$

An association rule is called *simple* if the head is a singleton.

To avoid generating redundant association rules, it is usually assumed that the body X and the head Y of the rule $X \Rightarrow Y$ are disjoint. Instead of all association rules, typically only the σ-frequent association rules, i.e., the association rules with frequency at least σ are computed. The intuition behind this restriction is that the support of the association rule immediately tells how many transactions in the database the association rule concerns. Another reason for concentrating only to σ-frequent association rules is that they can be computed from the σ-frequent itemsets by a straightforward algorithm (Algorithm 2.1) [AIS93].

Algorithm 2.1 Association rule mining.

Input: A collection $\mathcal{F}(\sigma, \mathcal{D})$ of σ-frequent itemsets in a transaction database $\mathcal{D}$.

Output: The collection $\mathcal{R}$ of association rules over the collection $\mathcal{F}(\sigma, \mathcal{D})$.

```
1: function ASSOCIATION-RULES(F(σ, D))
2:     R ← ∅
3:     for all Y ∈ F(σ, D) do
4:         for all X ⊂ Y do
5:             R ← R ∪ {X ⇒ Y \ X}
6:         end for
7:     end for
8:     return R
9: end function
```

2.3 From Frequent Itemsets to Interesting Patterns

The definition of frequent itemsets readily generalizes to arbitrary pattern collections $\mathcal{P}$ and databases $\mathcal{D}$ such that the frequency of a pattern $p \in \mathcal{P}$ in the database $\mathcal{D}$ can be determined. Also association rules can be defined for a pattern collection $\mathcal{P}$ if there is a suitable partial order $\preceq$ over the collection.

Definition 2.5 (partial order). A *partial order* $\preceq$ is a *transitive*, *antisymmetric* and *reflexive* binary relation, i.e., a relation $\preceq \subseteq \mathcal{P} \times \mathcal{P}$ such that $p \preceq p' \wedge p' \preceq p'' \Rightarrow p \preceq p''$, $p \preceq p' \wedge p' \preceq p \Rightarrow p = p'$ and $p \preceq p$ for all $p, p', p'' \in \mathcal{P}$. (Note that $p \preceq p'$ is equivalent to $\langle p, p' \rangle \in \preceq$.) We use the shorthand $p \prec p'$ when $p \preceq p'$ but $p' \not\preceq p$.

Elements $p, p' \in \mathcal{P}$ are called *comparable* with respect to the partial order $\preceq$ if and only if $p \preceq p'$ or $p' \preceq p$. If the elements are not comparable, then they are *incomparable*. A partial order is a *total order* in $\mathcal{P}$ if and only if all $p, p' \in \mathcal{P}$ are comparable.

Association rules can be defined over the pattern collection $\mathcal{P}$ and the partial order $\preceq$ over $\mathcal{P}$ if $p \preceq p'$ implies $fr(p, \mathcal{D}) \geq fr(p', \mathcal{D})$ for all $p, p' \in \mathcal{P}$. For example, itemsets are a special case of this: one such partial order $\preceq$ over the collection of all itemsets $X \subseteq \mathcal{I}$

is naturally defined by the set inclusion relation

$$X \preceq Y \iff X \subseteq Y$$

holding for all $X, Y \subseteq \mathcal{I}$. Then, by the definition of the frequency of itemsets (Definition 2.3), $X \subseteq Y$ implies $fr(X, \mathcal{D}) \geq fr(Y, \mathcal{D})$ for all $X, Y \subseteq \mathcal{I}$.

Definition 2.6 (frequent patterns and their association rules). Let $\mathcal{P}$ be a pattern collection, $\mathcal{D}$ a database, σ a positive value in the interval $[0, 1]$, and for each pattern $p \in \mathcal{P}$, let $fr(p, \mathcal{D})$ denote the frequency of p in $\mathcal{D}$. The collection $\mathcal{F}(\sigma, \mathcal{D})$ of σ-frequent patterns consists of the patterns $p \in \mathcal{P}$ such that $fr(p, \mathcal{D}) \geq \sigma$.

Let $\preceq$ be a partial order over the pattern collection $\mathcal{P}$ and let $p \preceq p'$ imply $fr(p, \mathcal{D}) \geq fr(p', \mathcal{D})$ for all $p, p' \in \mathcal{P}$. Then an association rule is a rule $p \Rightarrow p'$ where $p, p' \in \mathcal{P}$ and $p \preceq p'$. The accuracy of an association rule $p \Rightarrow p'$ is

$$acc(p \Rightarrow p', \mathcal{D}) = \frac{fr(p', \mathcal{D})}{fr(p, \mathcal{D})}.$$

The association rules can be generalized also for incomparable patterns $p, p' \in \mathcal{P}$ by defining

$$acc(p \Rightarrow p', \mathcal{D}) = \frac{fr(p'', \mathcal{D})}{fr(p, \mathcal{D})}.$$

where p'' is such a pattern in $\mathcal{P}$ that $p, p' \preceq p''$ and $fr(p'', \mathcal{D}) \geq fr(p''', \mathcal{D})$ for all $p''' \in \mathcal{P}$ with such that $p, p' \preceq p'''$.

Example 2.4 (frequent substrings and association rules). Let s be a string over an alphabet Σ and let the frequency of a string $p = p_1 \dots p_{|p|} \in \Sigma^*$ in s be the number of its occurrences in s divided by the length of s, i.e.,

$$fr(p, s) = \frac{\left|\left\{i : p = s_{i+1} \dots s_{i+|p|}\right\}\right|}{|s| - |p| + 1}.$$

Furthermore, let the partial order $\preceq$ over the strings in Σ^* be the substring relation, i.e.,

$$s \preceq t \iff \exists i \in \{0, \dots, |t| - |s|\} : s = t_{i+1} \dots t_{i+|s|}$$

for all $s, t \in \Sigma^*$. As $p \preceq p'$ then implies $fr(p, s) \geq fr(p', s)$, the association rules can be defined for substrings.

The frequencies of all strings in s can be computed in time $\mathcal{O}(|s|)$ by constructing a suffix tree or a suffix array of s. (For details on linear-time suffix tree and array constructions, see e.g. [FCFM00, GK97, KS03].) □

The previously outlined search strategies to find σ-frequent itemsets and association rules have been adapted to many kinds of patterns such as sequences [WH04, Zak01], episodes [CG03b, GAS03, MTV97], trees [XYLD03, Zak02], graphs [IWM03, KK01, WWS+02, YH02] and queries [DT01, GVdB02, MS03].

The interestingness predicate obtained by a minimum frequency threshold determines a *downward closed* pattern collection for many kinds of patterns.

Definition 2.7 (downward closed pattern collections). A pattern collection $\mathcal{P}$ is *downward closed* with respect to a partial order $\preceq$ and an interestingness predicate q if and only if $p \in \mathcal{P}_q$ implies that $p' \in \mathcal{P}_q$ for all $p' \preceq p$.

Many of the pattern discovery techniques are adaptations of the general levelwise search strategy for downward closed collections of interesting patterns [MT97]. The search procedure repeatedly evaluates all patterns whose all subpatterns are recognized to be interesting. The procedure is described by Algorithm 2.2 (which is an adaptation from [MT97]).

Algorithm 2.2 can be modified in such a way that the requirement of having downward closed pattern collection can be relaxed. Specifically, it is sufficient to require that the collection of potentially interesting patterns that has to be evaluated in the levelwise search is downward closed in the sense that there is a way to neglect other patterns in the collection. (For an example, see subsection 2.4.2.)

2.4 Condensed Representations of Pattern Collections

A major difficulty in pattern discovery is that the pattern collections tend to be too large to understand. Fortunately, the pattern

Algorithm 2.2 The levelwise algorithm for discovering interesting patterns.

Input: A pattern collection $\mathcal{P}$, a partial order $\preceq$ over $\mathcal{P}$ and an interestingness predicate $q : \mathcal{P} \to \{0, 1\}$ such that $p \preceq p'$ implies $q(p) \geq q(p')$ for all $p, p' \in \mathcal{P}$.

Output: The collection $\mathcal{P}_q$ of interesting patterns in $\mathcal{P}$.

```
1: function LEVELWISE(P, ⪯, q)
2:     P_q ← ∅                    ▷ No pattern is known to be interesting.
3:     P' ← P                     ▷ All patterns are potentially interesting.
4:     repeat                     ▷ Find the minimal still potentially interesting
           patterns and check whether they are interesting.
5:         K ← {p ∈ P' : p' ∈ P, p' ≺ p ⇒ p' ∈ P_q}
6:         P_q ← P_q ∪ {p ∈ K : q(p) = 1}
7:         P' ← P' \ K
8:     until K = ∅
9:     return P_q
10: end function
```

collections contain often redundant information and many patterns can be inferred from the other patterns. That is, the pattern collection can be described by its subcollection of irredundant patterns. The irredundancy of a pattern does not always depend only on the pattern collection and the interestingness predicate but also on the other irredundant patterns and the method for inferring all patterns in the collection from the interesting ones.

In pattern discovery literature such collections of irredundant patterns are known as *condensed (or concise) representations of pattern collections* [CG03a], although the condensed representations in the context of data mining were introduced in a slightly more general sense as small representations of data that are accurate enough with respect to a given class of queries [MT96].

2.4.1 Maximal and Minimal Patterns

Sometimes it is sufficient, for representing the pattern collection, to store only the *maximal patterns* in the collection [GKM$^+$03].

Definition 2.8 (maximal patterns). A pattern $p \in \mathcal{P}$ is *maximal* in the collection $\mathcal{P}$ with respect to the partial order $\prec$ if and only

if $p \not\preceq p'$ for all $p' \in \mathcal{P}$. The collection of maximal patterns in $\mathcal{P}$ is denoted by $Max(\mathcal{P}, \preceq)$.

It can be shown that the maximal interesting patterns in the collection determine the whole collection of interesting patterns if the interesting patterns form a downward closed pattern collection.

Proposition 2.1. *The collection $Max(\mathcal{P}_q, \preceq)$ of the maximal interesting patterns determines the collection $\mathcal{P}_q$ of interesting patterns if and only if $\mathcal{P}_q$ is downward closed.*

Proof. If the collection $\mathcal{P}_q$ is downward closed, then by the definition of maximality, for each pattern $p \in \mathcal{P}_q$ there is the maximal pattern in $p' \in \mathcal{P}_q$ such that $p \preceq p'$. Furthermore, for each maximal pattern $p' \in \mathcal{P}_q$ it holds $p \preceq p' \Rightarrow p \in \mathcal{P}_q$ if $\mathcal{P}_q$ is downward closed.

If the collection $\mathcal{P}_q$ is not downward closed, then there is a non-maximal pattern p such that $p \notin \mathcal{P}_q$ but $p \prec p'$ for some $p' \in \mathcal{P}_q$. The maximal patterns in $\mathcal{P}_q$ are not sufficient to point out that pattern. □

The maximal patterns in the collection of σ-frequent itemsets, i.e., the maximal σ-frequent itemsets in $\mathcal{D}$, are denoted by $\mathcal{FM}(\sigma, \mathcal{D})$. Representing a downward closed collection of patterns by the maximal patterns in the collection can reduce the space consumption drastically. For example, the number $|\mathcal{FM}(\sigma, \mathcal{D})|$ of maximal frequent itemsets can be exponentially smaller than the number $|\mathcal{F}(\sigma, \mathcal{D})|$ of all frequent itemsets.

Example 2.5 (the number of σ-frequent itemsets versus the number of maximal σ-frequent itemsets). Let us consider a transaction database $\mathcal{D}$ consisting only of one tuple $\langle 1, \mathcal{I} \rangle$. For this database and all possible minimum frequency thresholds $\sigma \in [0, 1]$ we have: $|\mathcal{FM}(\sigma, \mathcal{D})| = 1$ and $|\mathcal{F}(\sigma, \mathcal{D})| = 2^{|\mathcal{I}|}$. □

Example 2.6 (maximal 0.20-frequent itemsets in the course completion database). Let us denote the collection of the maximal σ-frequent itemsets in $\mathcal{D}$ with cardinality i by $\mathcal{FM}(\sigma, \mathcal{D})[i]$. Then the cardinality distributions of the maximal 0.20-frequent itemsets in the course completion database (see Subsection 2.2.1) and the most frequent itemsets of each cardinality are as shown in Table 2.4

Table 2.4: The number of the maximal 0.20-frequent itemsets of each cardinality in the course completion database, the most frequent itemsets of each cardinality and their supports.

i	$\lvert\mathcal{FM}(\sigma,\mathcal{D})[i]\rvert$	the largest $X \in \mathcal{FM}(\sigma,\mathcal{D})[i]$	$supp(X,\mathcal{D})$
0	0	-	-
1	1	$\{33\}$	519
2	21	$\{2,26\}$	547
3	41	$\{0,2,19\}$	529
4	58	$\{0,2,7,17\}$	553
5	38	$\{2,3,5,9,10\}$	511
6	66	$\{0,1,2,3,4,5\}$	550
7	20	$\{0,2,3,5,7,14,20\}$	508
8	8	$\{0,2,3,5,7,12,13,15\}$	512

□

Due to the potential reduction in the number of itemsets needed to find, several search strategies for finding only the maximal frequent itemsets have been developed [BCG01, BGKM02, BJ98, GZ01, GZ03, GKM+03, SU03].

It is not clear, however, whether the maximal interesting patterns are the most concise subcollection of patterns to represent the interesting patterns. The collection could be represented also by the *minimal* uninteresting patterns.

Definition 2.9 (minimal patterns). A pattern $p \in \mathcal{P}$ is *minimal* in the collection $\mathcal{P}$ with respect to the partial order $\preceq$ if and only if $p' \not\preceq p$ for all $p' \in \mathcal{P}$. The collection of minimal patterns in $\mathcal{P}$ is denoted by $Min(\mathcal{P},\preceq)$.

As in the case of the maximal interesting patterns, it is easy to see that the minimal uninteresting patterns uniquely determine the collection of the interesting patterns if the pattern collection is downward closed.

The collection of minimal σ-infrequent itemsets in $\mathcal{D}$ is denoted by $\mathcal{IM}(\sigma,\mathcal{D})$. It is much more difficult to relate the number of minimal uninteresting patterns to the number of interesting patterns, even when the collection of interesting patterns is downward closed. In fact, for a downward collection $\mathcal{P}_q$ of interesting patterns patterns the number $\lvert Min(\mathcal{P}_{\bar{q}},\preceq)\rvert$ of uninteresting patterns cannot

be bounded very well in general from above nor from below by the number $|\mathcal{P}_q|$ of interesting patterns and the number $|Max(\mathcal{P}_q, \preceq)|$ of maximal interesting patterns.

Bounding the number of the minimal infrequent itemsets by the number of frequent itemsets is also slightly more complex than bounding the number of maximal frequent itemsets.

Example 2.7 (the number of σ-frequent itemsets versus the number of minimal σ-infrequent itemsets). The number $|\mathcal{IM}(\sigma, \mathcal{D})|$ of minimal infrequent itemsets can be $|\mathcal{I}|$ times larger than the number of $|\mathcal{F}(\sigma, \mathcal{D})|$ frequent itemsets.

Namely, let the transaction database consists of transaction $\langle 1, \emptyset \rangle$. Then $\mathcal{F}(\sigma, \mathcal{D}) = \{\emptyset\}$ but $\mathcal{IM}(\sigma, \mathcal{D}) = \{\{A\} : A \in \mathcal{I}\}$. This is also the worst case since each frequent itemset $X \in \mathcal{F}(\sigma, \mathcal{D})$ can have at most $|\mathcal{I}|$ superitemsets in $\mathcal{IM}(\sigma, \mathcal{D})$.

If the collection $\mathcal{IM}(\sigma, \mathcal{D})$ is empty, then $|\mathcal{F}(\sigma, \mathcal{D})| > c\,|\mathcal{IM}(\sigma, \mathcal{D})|$ for all values $c \in \mathbb{R}$. Otherwise, let the transaction database $\mathcal{D}$ consist of one transaction with itemset $\mathcal{I} \setminus \{A\}$ for each $A \in \mathcal{I}$ and let $\sigma = 1/|\mathcal{D}|$. Then $|\mathcal{IM}(\sigma, \mathcal{D})|$ is exponentially smaller than $|\mathcal{F}(\sigma, \mathcal{D})|$. □

It is known that the number $|\mathcal{FM}(\sigma, \mathcal{D})|$ of maximal itemset can be bounded from above by $(|\mathcal{I}| - \sigma\,|\mathcal{D}| + 1)\,|\mathcal{IM}(\sigma, \mathcal{D})|$ if $\mathcal{IM}(\sigma, \mathcal{D})$ is not empty [BGKM02]. Furthermore, it is clear that $|\mathcal{IM}(\sigma, \mathcal{D})| \leq |\mathcal{I}|\,|\mathcal{FM}(\sigma, \mathcal{D})|$ for all minimum frequency thresholds $\sigma \in [0, 1]$.

The collection $\mathcal{IM}(\sigma, \mathcal{D})$ can be obtained from $\mathcal{FM}(\sigma, \mathcal{D})$ by generating all minimal hypergraph transversals in the hypergraph

$$\{\mathcal{I} \setminus X : X \in \mathcal{FM}(\sigma, \mathcal{D})\},$$

i.e., in the hypergraph consisting of the complements of the maximal σ-frequent itemsets in $\mathcal{D}$ [MT97].

The slack in the bounds between the number of the maximal frequent and the number of the minimal infrequent itemsets implies that it cannot be decided in advance without seeing the data which of the representations — $\mathcal{FM}(\sigma, \mathcal{D})$ or $\mathcal{IM}(\sigma, \mathcal{D})$ — is better. In practice, the smaller of the collections $\mathcal{FM}(\sigma, \mathcal{D})$ and $\mathcal{IM}(\sigma, \mathcal{D})$ can be chosen. Each maximal frequent and each minimal infrequent itemset determines its subitemsets to be frequent and superitemsets to be infrequent. Sometimes one can obtain a representation for

$\mathcal{F}(\sigma, \mathcal{D})$ that is smaller than $\mathcal{FM}(\sigma, \mathcal{D})$ or $\mathcal{IM}(\sigma, \mathcal{D})$ by choosing some itemsets from $\mathcal{FM}(\sigma, \mathcal{D})$ and some from $\mathcal{IM}(\sigma, \mathcal{D})$ in such a way that the chosen itemsets determine the collection $\mathcal{F}(\sigma, \mathcal{D})$ uniquely [Mie04c].

Sometimes it is not sufficient to represent only the collection of interesting patterns but also the quality values for the patterns are needed as well. For example, the accuracy of an association rule $X \Rightarrow Y$ depends on the frequencies of the frequent itemsets X and $X \cup Y$. One solution is to determine the pattern collection as described above and describe the quality values in the collection of interesting patterns separately. The quality values can be represented, e.g., by a simplified database [Mie03c] or by a random sample of transactions from the database [Mie04c]. In these approaches, however, the condensed representation is not a subcollection of the patterns anymore. Thus, a different approach is required if the condensed representation of the pattern collection is required to consist of patterns.

2.4.2 Closed and Free Patterns

For the rest of the chapter we shall focus on interestingness measures ϕ such that $p \preceq p'$ implies $\phi(p) \geq \phi(p')$ for all $p, p' \in \mathcal{P}$, i.e., to anti-monotone interestingness measures. Then maximal interesting patterns and their quality values determine lower bounds for all other interesting patterns as well. The highest lower bound obtainable for the quality value of a pattern p from the quality values of the maximal patterns is

$$\max\left\{\phi(p') : p \preceq p' \in Max(\mathcal{P}_q, \preceq)\right\}.$$

The patterns p with the quality value matching with the maximum quality value of the maximal interesting patterns that are superpatterns of p can be removed from the collection of potentially irredundant patterns if the maximal interesting patterns are decided to be irredundant. An exact representation for the collection of interesting patterns can be obtained by repeating these operations. The collection of the irredundant patterns obtained by the previous procedure is called the collection of *closed* interesting patterns [ZO98].

Definition 2.10 (closed patterns). A pattern $p \in \mathcal{P}$ is *closed* in the collection $\mathcal{P}$ with respect to the partial order $\prec$ and the interestingness measure ϕ if and only if $p \prec p'$ implies $\phi(p) > \phi(p')$ for all $p' \in \mathcal{P}$. The collection of closed patterns in $\mathcal{P}$ is denoted by $Cl(\mathcal{P}, \preceq, \phi)$. For brevity, $\preceq$ and ϕ can be omitted when they are clear from the context.

The collection of closed σ-frequent itemsets in $\mathcal{D}$ is denoted by $\mathcal{FC}(\sigma, \mathcal{D})$. One procedure for detecting the closed patterns (for a given pattern collection $\mathcal{P}$, a partial order $\preceq$ and an interestingness measure ϕ) is described as Algorithm 2.3.

Algorithm 2.3 Detection of closed patterns

Input: A collection $\mathcal{P}$ of patterns, a partial order $\preceq$ over $\mathcal{P}$ and an interestingness measure ϕ.

Output: The collection $Cl(\mathcal{P}, \preceq, \phi)$ of patterns in $\mathcal{P}$ that are closed with respect to ϕ.

```
1: function CLOSED-PATTERNS(P, ≺, φ)
2:     K ← P
3:     while K ≠ ∅ do
4:         K' ← Max(K, ⪯)
5:         Cl(P, ⪯, φ) ← Cl(P, ⪯, φ) ∪ K'
6:         K ← K \ K'
7:         K ← {p ∈ K : p' ∈ K', p ≺ p' ⇒ φ(p) > φ(p')}
8:     end while
9:     return Cl(P, ⪯, φ)
10: end function
```

Example 2.8 (closed frequent itemsets). Let the transaction database $\mathcal{D}$ be the same as in Example 2.2, i.e.,

$$\mathcal{D} = \{\langle 1, ABC\rangle, \langle 2, AB\rangle, \langle 3, ABCD\rangle, \langle 4, BC\rangle\}.$$

Then

$$\mathcal{FC}(2/4, \mathcal{D}) = \{B, AB, BC, ABC\} = \mathcal{F}(2/4, \mathcal{D}) \setminus \{\emptyset, A, C, AC\}.$$

□

Example 2.9 (closed 0.20-frequent itemsets in the course completion database). Let us denote the collection of the closed

σ-frequent itemsets in $\mathcal{D}$ with cardinality i by $\mathcal{FC}(\sigma, \mathcal{D})[i]$. Then the cardinality distributions of the closed 0.20-frequent itemsets in the course completion database (see Subsection 2.2.1) and the most frequent closed itemsets of each cardinality are as shown in Table 2.5.

Table 2.5: The number of the closed 0.20-frequent itemsets of each cardinality in the course completion database, the most frequent closed itemsets of each cardinality and their supports.

i	$\lvert\mathcal{FC}(\sigma, \mathcal{D})[i]\rvert$	the largest $X \in \mathcal{FC}(\sigma, \mathcal{D})[i]$	$supp(X, \mathcal{D})$
0	1	$\emptyset$	2405
1	34	$\{0\}$	2076
2	186	$\{0, 1\}$	1345
3	454	$\{2, 3, 5\}$	960
4	638	$\{0, 2, 3, 5\}$	849
5	519	$\{0, 2, 3, 4, 5\}$	681
6	238	$\{0, 2, 3, 5, 7, 12\}$	588
7	58	$\{2, 3, 5, 7, 12, 13, 15\}$	547
8	8	$\{0, 2, 3, 5, 7, 12, 13, 15\}$	512

□

It is a natural question whether the closed interesting patterns could be discovered immediately without generating all interesting patterns. For many kinds of frequent closed patterns this question has been answered positively; there exist methods for mining directly, e.g., closed frequent itemsets [PBTL99, PCT+03, WHP03, ZH02], closed frequent sequences [WH04, YHA03], and closed frequent graphs [YH03] from data. Recently it has been shown that frequent closed itemsets can be found in time polynomial in the size of the output [UAUA04].

The number $\lvert Cl(\mathcal{P}_q)\rvert$ of closed interesting patterns is at most the number $\lvert\mathcal{P}_q\rvert$ of all interesting patterns and at least the number $\lvert Max(\mathcal{P}_q)\rvert$ of maximal interesting patterns, since $\mathcal{P}_q \supseteq Cl(\mathcal{P}_q) \supseteq Max(\mathcal{P}_q)$. Tighter bounds for the number of closed interesting patterns depend on the properties of the pattern collection $\mathcal{P}$.

Example 2.10 (the number of σ-frequent itemsets versus the number of closed σ-frequent itemsets). Similarly to the maximal frequent itemsets, the number of closed frequent itemsets

in the transaction database $\mathcal{D} = \{\langle 1, \mathcal{I}\rangle\}$ is exponentially smaller than the number of all frequent itemsets for all minimum frequency thresholds $\sigma \in (0, 1]$. □

However, the number of closed frequent sets can be exponentially larger than the number of maximal itemsets.

Example 2.11 (the number of maximal σ-frequent itemsets versus the number of closed σ-frequent itemsets). Let $\mathcal{D}$ consist of one transaction for each subset of size $|\mathcal{I}| - 1$ of $\mathcal{I}$ and $\lceil \sigma/(1-\sigma) \rceil |\mathcal{I}|$ transactions consisting of the itemset $\mathcal{I}$. Then $\mathcal{FM}(\sigma, \mathcal{D}) = \{\mathcal{I}\}$ but $\mathcal{FC}(\sigma, \mathcal{D}) = \{X \subseteq \mathcal{I}\} = 2^{\mathcal{I}}$. □

Example 2.12 (comparing all, closed and maximal σ-frequent itemsets in the course completion database). Let us consider the course completion database (see Subsection 2.2.1). In that transaction database, the number of all, closed and maximal σ-frequent itemsets for several different minimum frequency thresholds σ are as shown in Table 2.6.

Table 2.6: The number of all, closed and maximal σ-frequent itemsets in the course completion database for several different minimum frequency thresholds σ.

σ	all	closed	maximal
0.50	7	7	3
0.40	18	18	10
0.30	103	103	28
0.25	363	360	80
0.20	2419	2136	253
0.15	19585	12399	857
0.10	208047	82752	4456
0.05	5214764	918604	43386
0.04	12785998	1700946	80266
0.03	38415247	3544444	172170
0.02	167578070	8486933	414730
0.01	1715382996	23850242	1157338

The number of maximal σ-frequent itemsets is quite low compared even to the number of closed σ-frequent itemsets. The number of closed σ-frequent itemsets is also often considerably smaller

than the number of all σ-frequent itemsets, especially for low values of σ. □

A desirable property of closed frequent itemsets is that they can be defined by closures of the itemsets. A *closure* of an itemset X in a transaction database $\mathcal{D}$ is the intersection of the transactions in $\mathcal{D}$ containing X, i.e.,

$$cl(X, \mathcal{D}) = \bigcap_{\langle i,Y\rangle \in \mathcal{D}, Y \supseteq X} Y.$$

Clearly, there is unique closure $cl(X, \mathcal{D})$ in the transaction database $\mathcal{D}$ for each itemset X. It can be shown that each closed itemset is its own closure [GW99, Kry01, PBTL99]. Thus, the collection of closed σ-frequent itemsets can be expressed alternatively as

$$\mathcal{FC}(\sigma, \mathcal{D}) = \{X \subseteq \mathcal{I} : cl(X, \mathcal{D}) = X, fr(X, \mathcal{D}) \geq \sigma\}.$$

In fact, this is often used as a definition of a closed itemset. In this dissertation, however, the closed patterns are not defined using closures; the reason is that it is not clear in the case of other pattern collections than frequent itemsets whether the closure can be defined in a natural way and when it is unique.

The levelwise algorithm (Algorithm 2.2) can be adapted to mine also closed itemsets: Let $\mathcal{FI}(\sigma, \mathcal{D})$ denote the collection of all σ-frequent items in $\mathcal{D}$ and let $\mathcal{FC}_k$ be the collection of the closed frequent itemsets at level k. The level for closed itemsets is the length of the shortest path from the itemset to the closure of the empty itemset in the partial order defined by the set inclusion relation. Thus, the zeroth level consists of the closure of the empty itemset. The collection of potentially frequent closed itemsets at level k ($k \geq 1$) consists of closures of $X \cup \{A\}$ for each frequent closed itemset X in level $k-1$ and each frequent item $A \notin X$. The adaptation of Algorithm 2.2 for frequent closed itemset mining is described as Algorithm 2.4.

The collection of closed interesting patterns can be seen a refinement of the collection of maximal interesting patterns: a closed interesting pattern p is a maximal interesting pattern for the minimum quality value thresholds in the interval

$$\left(\max\left\{\phi(p') : p \prec p'\right\}, \phi(p)\right].$$

Algorithm 2.4 The levelwise algorithm for discovering frequent closed itemsets in a transaction database.

Input: A transaction database $\mathcal{D}$ and a minimum frequency threshold $\sigma \in (0, 1]$.

Output: The collection $\mathcal{FC}(\sigma, \mathcal{D})$ of σ-frequent closed itemsets in $\mathcal{D}$.

1: **function** CLOSURES($\sigma, \mathcal{D}$)
2: $\quad\mathcal{I} \leftarrow \bigcup_{X \in \mathcal{D}} X$
3: $\quad\mathcal{FI} \leftarrow \{A \in \mathcal{I} : fr(A, \mathcal{D}) \geq \sigma\}$
4: $\quad i \leftarrow 0$
5: $\quad\mathcal{FC}_i \leftarrow \{cl(\emptyset, \mathcal{D})\}$
6: $\quad\mathcal{FC}(\sigma, \mathcal{D}) \leftarrow \mathcal{FC}_0$
7: $\quad$**repeat**
8: $\quad\quad i \leftarrow i + 1$
9: $\quad\quad\mathcal{K} \leftarrow \{cl(X \cup \{A\}, \mathcal{D}) : X \in \mathcal{FC}_{i-1}, A \in \mathcal{FI} \setminus X\}$
10: $\quad\quad\mathcal{FC}_i \leftarrow \{X \in \mathcal{K} : fr(X, \mathcal{D}) \geq \sigma\} \setminus \mathcal{FC}(\sigma, \mathcal{D})$
11: $\quad\quad\mathcal{FC}(\sigma, \mathcal{D}) \leftarrow \mathcal{FC}(\sigma, \mathcal{D}) \cup \mathcal{FC}_i$
12: $\quad$**until** $\mathcal{K} = \emptyset$
13: $\quad$**return** $\mathcal{FC}(\sigma, \mathcal{D})$
14: **end function**

A natural relaxation of the closed interesting patterns is to store maximal interesting patterns for several minimum quality value thresholds. For example, the collections

$$\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{FM}(\sigma + \epsilon, \mathcal{D}), \ldots, \mathcal{FM}(\sigma + (\lceil (1 - \sigma) / \epsilon \rceil - 1)\, \epsilon, \mathcal{D})$$

of the maximal frequent itemsets are sufficient for estimating the frequency of any σ-frequent itemset in $\mathcal{D}$ by the maximum absolute error at most ϵ. Furthermore, the frequencies of the maximal frequent itemsets are not needed: it is sufficient to know in which of the collections $\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{FM}(\sigma + \epsilon, \mathcal{D}), \ldots$ the maximal pattern belongs to and what is the minimum frequency threshold for that collection. Then the frequency of an itemset X can be estimated to be the maximum of the minimum frequency thresholds of the maximal itemset collections that contain an itemset containing the itemset X.

Algorithm 2.3 can be modified to solve this task of approximating the collection of σ-frequent closed itemsets. To approximate especially the collections of the frequent itemsets, many maximal

frequent itemset mining techniques can be adapted for mining the maximal frequent itemset collections for several minimum frequency thresholds, see e.g. [PDZH02].

An alternative notion of approximating closed frequent itemsets is proposed in [BB00]. The approach readily generalizes to any collection of interesting patterns with an anti-monotone interestingness measure: a pattern is considered to be ϵ-closed if the absolute difference between its quality value and the largest quality value of its superpatterns is more than ϵ.

Finally, an approach based on simplifying interestingness values to approximate closed interesting patterns is described in Chapter 3 of this dissertation and another approximation based on pattern ordering with respect to the informativeness of the prefixes of the ordering is proposed in Chapter 4.

Instead of defining irredundant patterns to be those that have strictly higher quality values than any of their superpatterns, the irredundant patterns could be defined to be those that have strictly lower quality values than any of their subpatterns. The latter patterns are called *free patterns* [BBR03], *generators* [PBTL99] or *key patterns* [BTP$^+$00].

Definition 2.11 (free patterns). A pattern $p \in \mathcal{P}$ is *free* in the collection $\mathcal{P}$ with respect to the partial order $\prec$ and the interestingness measure ϕ if and only if $p' \prec p$ implies $\phi(p) < \phi(p')$ for all $p' \in \mathcal{P}$. The collection of free patterns in $\mathcal{P}$ is denoted by $Gen(\mathcal{P})$.

The collection of free σ-frequent itemsets in $\mathcal{D}$ is denoted by $\mathcal{FG}(\sigma, \mathcal{D})$. Unfortunately, the free interesting patterns $Gen(\mathcal{P}_q)$ are not always a sufficient representation for all interesting patterns but also minimal free uninteresting patterns, i.e., the patterns in the collection $Min(Gen(\mathcal{P}_{\bar{q}}))$ are needed.

Example 2.13 (free frequent itemsets). Let the transaction database $\mathcal{D}$ be the same as in Example 2.2, i.e.,

$$\mathcal{D} = \{\langle 1, ABC\rangle, \langle 2, AB\rangle, \langle 3, ABCD\rangle, \langle 4, BC\rangle\}.$$

Then

$$\mathcal{FG}(1/4, \mathcal{D}) = \{\emptyset, A, C, AC\} = \mathcal{F}(1/4, \mathcal{D}) \setminus \{B, AB, BC, ABC\}$$

This is not, however, sufficient to determine the collection of 1/4-frequent itemsets in $\mathcal{D}$ since there is no information about B nor D. The item B is frequent but not free, whereas the item D is free but not frequent. □

As in the case of closed interesting patterns, the number of free interesting patterns is at most the number of all interesting patterns. The number of free interesting itemsets can be smaller than even the number of maximal interesting or minimal uninteresting patterns.

In the case of frequent itemsets, the number of free frequent itemsets is always at least as large as the number of closed frequent itemsets since each free itemset has a only one closure but several free itemsets can share the same one. Although the free frequent itemsets seem to have many disadvantages, they have one major advantage compared to closed frequent itemsets: collections of free frequent itemsets are downward closed [BBR03]. Thus, closed frequent itemsets can be discovered from free frequent itemsets by computing the closures for all free frequent itemsets. Notice that if free frequent itemsets are used only to compute the closed frequent itemsets, the minimal free infrequent itemsets are not needed for the representation since for each closed frequent itemset X there is at least one free frequent itemset Y such that $X = cl(Y, \mathcal{D})$.

Similarly to closed frequent itemsets, also mining the approximate free itemset collections based on a few different notions of approximation has been studied [BBR03, PDZH02].

2.4.3 Non-Derivable Itemsets

Taking the maximum or the minimum of the quality values of the super- or subpatterns are rather simple methods of inferring the unknown quality values but not much more complex inference techniques are useful with arbitrary anti-monotone interestingness measures. (Note that this is the case even with arbitrary frequent pattern collections since the only requirement for frequency is the anti-monotonicity.) For some pattern collections with suitable interestingness measures it is possible to find more concise representations.

For example, several more sophisticated condensed representations have been developed for frequent itemsets [BR01, CG03a,

Kry01]. This line of work can be seen to be culminated on *non-derivable itemsets* [CG02]. The idea of non-derivable itemsets is to deduce lower and upper bounds for the frequency of the itemset from the frequencies of its subitemsets.

Definition 2.12 (non-derivable itemsets). Let $\overline{fr}$ and $\underline{fr}$ denote mappings that give upper and lower bounds for the frequency of any itemset over $\mathcal{I}$. An itemset $X \subseteq \mathcal{I}$ is *non-derivable* with respect to the transaction database $\mathcal{D}$ (and functions $\overline{fr}$ and $\underline{fr}$) if and only if the lower bound $\underline{fr}(X, \mathcal{D})$ is strictly smaller than the upper bound $\overline{fr}(X, \mathcal{D})$. The collection of non-derivable itemsets is denoted by $\mathcal{N}(\mathcal{D})$.

One bound for the frequencies can be computed using inclusion-exclusion [CG02]. (An alternative to inclusion-exclusion would be to use (integer) linear programming [BSH02, Cal04a]. However, if the bounds for the frequencies are computed from the frequencies of all subitemsets, then inclusion-exclusion leads to the best possible solution [Cal04b].) From the inequality

$$\sum_{Y \subseteq Z \subseteq X} (-1)^{|Z \setminus Y|} fr(Z, d) \geq 0$$

holding for all X and Y, it is possible to derive upper and lower bounds for the frequency of the itemsets X in $\mathcal{D}$ [CG03a]:

$$\overline{fr}(X, \mathcal{D}) = \min_{Y \subset X} \left\{ \sum_{Y \subseteq Z \subset X} (-1)^{|X \setminus Z|+1} fr(Z, \mathcal{D}) : |X \setminus Y| \text{ is odd} \right\}$$

$$\underline{fr}(X, \mathcal{D}) = \max_{Y \subset X} \left\{ \sum_{Y \subseteq Z \subset X} (-1)^{|X \setminus Z|+1} fr(Z, \mathcal{D}) : |X \setminus Y| \text{ is even} \right\}$$

The collection of non-derivable itemsets is downward closed. The largest non-derivable itemset is at most of size $\lfloor \log_2 |\mathcal{D}| \rfloor$ [CG02]. To represent frequent itemsets it is sufficient to store the frequent non-derivable itemsets and the minimal infrequent non-derivable itemsets with upper bounds to the frequency at least the minimum frequency threshold.

Example 2.14 (non-derivable itemsets). Let the transaction database $\mathcal{D}$ be the same as in Example 2.2, i.e.,

$$\mathcal{D} = \{\langle 1, ABC \rangle, \langle 2, AB \rangle, \langle 3, ABCD \rangle, \langle 4, BC \rangle\}.$$

Then $\mathcal{N}(\mathcal{D}) = \{\emptyset, A, B, C, AC\}$. □

The approach of non-derivable itemsets is essentially different from the other condensed representations described, as no additional assumptions are made about the itemsets with unknown frequencies: their frequencies can be determined uniquely using, e.g., inclusion-exclusion. In contrast, using closed and free itemsets, each unknown frequency is assumed to be determined exactly as the maximum frequency of its superitemsets and the minimum frequency of its subitemsets, respectively.

The problem of finding non-derivable representations for essentially other pattern classes than itemsets is a very important and still largely open problem.

2.5 Exploiting Patterns

The real goal in pattern discovery is rarely just to obtain the patterns themselves but to use the discovered patterns.

One indisputable use of patterns is to disclose interesting aspects of the data. The suitability of different ways to represent pattern collections for the disclosure depends crucially on the actual application and the goals of data mining in the task at hand. However, at least the number of patterns and their complexity affect the understandability of the collection.

In practice, the number of patterns in the representation is strongly affected by the application and the database. For example, when represented explicitly, the itemset collection consisting only of the itemset $\mathcal{I}$ is probably easier to understand than the collection $2^{\mathcal{I}}$ of all subsets of $\mathcal{I}$. The explicit representation, however, is not always to most suitable.

Example 2.15 (represeting a collection implicitly). Sometimes the database $\mathcal{D}$ can be expected to be so dense that all frequent itemsets are also closed, i.e., $\mathcal{F}(\sigma, \mathcal{D}) = \mathcal{FC}(\sigma, \mathcal{D})$ under normal circumstances (with respect to the assumptions). If only the closed frequent itemsets are being represented, then it is most convenient to describe the collection by its maximal itemsets and those non-maximal itemsets that are not closed. Thus, it would be very surprising if the database $\mathcal{D}$ happens to be such that the

only closed itemset would be $\mathcal{I}$, and recognizing exactly that fact from the representation of the collection, i.e., the collection $2^{\mathcal{I}} \setminus \{\mathcal{I}\}$, would be quite arduous. □

Also the complexity of the representation can have a significant influence to the understandability. For example, the smallest Turing machine generating the pattern collection is probably quite an unintuitive representation. (The length of the encoding of such a Turing machine is called the Kolmogorov complexity or algorithmic information of the pattern collection [Cal02, LV97].) Similar situations occur also with the condensed representations. For example, although the number of non-derivable itemsets is usually less than the number of free frequent itemsets, the collection of the free frequent itemsets might still be more understandable since for most of us choosing the minimum value is more natural operation than computing all possible inclusion-exclusion truncations.

Data mining is an exploratory process to exploit the data. The data or the patterns derived from the data might not be understandable as whole and the right questions to be asked about the data are not always known in advance. Thus, it would be useful to be able to answer (approximately) to several queries to patterns and data. (A database capable to support data mining by means of that kind of queries is often called an *inductive database* [Bou04, DR03, IM96, Man97].) Three most important aspects of approximate query answering are the following:

Representation size. The size of the summary structure needed for answering the queries is very important. In addition to the actual space required for the storage, the size can affect also the efficiency of query answering: it is much more expensive to retrieve patterns from, e.g., tertiary memory than doing small computations based on patterns in main memory. For example, if all σ-frequent itemsets and their frequencies fit into main memory, then the frequency queries can be answered very efficiently for σ-frequent itemsets compared to computing the frequency by scanning through the complete transaction database that might reside on an external server with heavy load. There are many ways how pattern collections can be stored concisely. For example, representing the pattern collection and their quality values by listing just the

quality values leads to quite concise representations [Mie05b].

The efficiency of query answering. It is not always known in advance what should be asked about the data. Also, the pattern collections can be too large to digest completely in one go. Thus, different viewpoints to data and patterns might be helpful. The efficient query answering can be provided by efficient index structures. For example, although the number of closed frequent itemsets is often considerably smaller than the number of all frequent itemsets, retrieving the frequency of a given frequent itemset can be more difficult. If all frequent itemsets are stored, then answering the frequency query $fr(X, \mathcal{D})$ can be implemented as a membership query: the frequencies of the frequent itemsets can be stored in a trie and thus the frequency of a frequent itemset X can be found in time linear in $|X|$. Answering the same query when storing only the closed frequent itemsets in a trie is much more difficult: in the worst case the whole trie has to be transversed. This problem can be relieved by inserting some additional links to the trie. The trie representations can be generalized to deterministic automata representations [Mie05a].

The accuracy of the answers. Sometimes approximate answers to queries are sufficient if they can be provided substantially faster than the exact answers. Furthermore, it might be too expensive to store all data (or patterns) and thus exact answers might be impossible [BBD+02]. A simple approach to answer quite accurately to many queries is to store a random sample of the data. For example, storing a random subset $\mathcal{D}'$ of a transactions in the transaction database $\mathcal{D}$ gives good approximations to frequency queries [Toi96, Mie04c]. Another alternative is to store some subset of itemsets and estimate the unknown frequencies from them [KS02, MT96, PMS03]. A natural fusion of these two approaches is use both patterns and data to represent the structure facilitating the possible queries [GGM03]. When the query answers are inaccurate, it is often valuable to obtain some bounds to the errors. The frequencies of the frequent itemsets, for example, can be bounded below and above by, e.g., linear programming and (truncated) inclusion-exclusion [BSH02, CG02].

CHAPTER 3

Frequency-Based Views to Pattern Collections

It is a highly non-trivial task to define an (anti-monotone) interestingness measure ϕ such that there is a minimum quality value threshold σ capturing almost all truly interesting and only few uninteresting patterns in the collection. One way to augment the interestingness measure is to define additional constraints for the patterns. The use of constraints is a very important research topic in pattern discovery but the research has been concentrated mostly on structural constraints on patterns and pattern collections [BGMP03, BJAG00, DRJLM02, GVdB00, KGBW03, LLN03, Mie03c, SVA97]. Typical examples of structural constraints for patterns are constraints for items and itemsets: an interesting itemset can be required or forbidden to contain certain items or itemsets. Other typical constraints for pattern collections are monotone and anti-monotone constraints such as minimum and maximum frequency thresholds, or minimum and maximum cardinality constraints for the itemsets.

Example 3.1 (constraints in itemset mining). Let the set $\mathcal{I}$ of items be products sold in a grocery store. The transaction database $\mathcal{D}$ could then consist of transactions corresponding to purchases of customers that have bought something from the shop at least three times. As a constrained itemset mining task, we could be interested to find itemsets that

1. do not contain garlic,

2. consist of at least seven products,
3. contain at least two vegetables or bread and sour milk, and
4. cost at most ten euros.

These constraints attempt to characterize global travelers that are likely to become low-profit regular customers.

The first and the third constraint are examples of constraints for items or itemsets. The second and the fourth constraints are examples of anti-monotone and monotone constraints, respectively.

Clearly, all constraints could be expressed as boolean combinations of item constraints, since that is sufficient for defining any subcollection of $2^{\mathcal{I}}$ and all constraints define a subcollection of $2^{\mathcal{I}}$. However, that would not be very intuitive and also it could be computationally very demanding to find all satisfying truth assignments (corresponding to itemsets) for an arbitrary boolean formula. □

In this chapter we propose a complementary approach to further restrict and sharpen the collection of interesting patterns. The approach is based on simplifying the quality values of the patterns and it can be seen as a natural generalization of characterizing the interesting patterns by a minimum quality value threshold σ for the quality values of the patterns. The quality value simplifications can be adapted easily to pattern classes of various kind since they depend only on the quality values of the interesting patterns and not on the structural properties of the patterns. Simplifying the quality values is suitable for interactive pattern discovery as post-processing of a pattern collection containing the potentially interesting patterns. For example, in the case of itemsets, the collection of potentially interesting patterns usually consists of the σ-frequent itemsets for the smallest possible minimum frequency threshold σ such that the frequent itemset mining is still feasible in practice.

In addition to making the collection more understandable in general, the simplifications of the quality values can be used to reduce the number of interesting patterns by discretizing the quality values and removing the patterns whose discretized quality values can be inferred (approximately) from the quality values of the patterns that are not removed. Although there might be more powerful ways to condense the collection of interesting patterns, the great virtue of discretization is its conceptual simplicity: it is relatively

understandable how the discretization simplifies the structure of the quality values in the collection of interesting patterns.

This chapter is based on the article "Frequency-Based Views to Pattern Collections" [Mie03d]. For brevity, we consider for the rest of the chapter frequencies instead of arbitrary quality values.

3.1 Frequency-Based Views

A *simplification of frequencies* is a mapping $\psi : [0,1] \to I$, where I is a collection of non-overlapping intervals covering the interval $[0,1]$, i.e.,

$$I \subset \{[a,b], [a,b), (a,b], (a,b) \subseteq [0,1]\}$$

such that $\bigcup I = [0,1]$ and $i \cap j = \emptyset$ for all $i, j \in I$.

Example 3.2 (frequent patterns). The collection $\mathcal{F}(\sigma, \mathcal{D})$ of σ-frequent patterns can be defined using frequency simplifications as follows:

$$\psi(fr(p,\mathcal{D})) = \begin{cases} fr(p,\mathcal{D}) & \text{if } fr(p,\mathcal{D}) \geq \sigma \text{ and} \\ [0,\sigma) & \text{otherwise.} \end{cases}$$

□

There are several immediate applications of frequency simplifications. They can be used, for example, to focus on some particular frequency-based property of the pattern class.

Example 3.3 (focusing on some frequencies). First, example 3.2 is an example of focusing on some frequencies.

As a second example, the data analyst might be interested only in very frequent (e.g., the frequency is at least $1-\epsilon$) and very infrequent (e.g., the frequency is at most ϵ) patterns. Then the patterns in the interval $(\epsilon, 1-\epsilon)$ could be neglected or their frequencies could be mapped all to the interval $(\epsilon, 1-\epsilon)$. Thus, the corresponding frequency simplification is the mapping

$$\psi(fr(p,\mathcal{D})) = \begin{cases} (\epsilon, 1-\epsilon) & \text{if } fr(p,\mathcal{D}) \in (\epsilon, 1-\epsilon) \text{ and} \\ fr(p,\mathcal{D}) & \text{otherwise.} \end{cases}$$

As a third example, let us consider association rules. The data analyst might be interested in the rules with accuracy close to 1/2

(e.g., within some positive constant ϵ), i.e., the association rules $p \Rightarrow p'$ $(p, p' \in \mathcal{P}, p \preceq p')$ with no predictive power. Thus, in that case the frequency simplification $\psi(acc(p'', \mathcal{D}))$ of the association rule $p \Rightarrow p'$ (denoted by a shorthand p'') is

$$\psi(acc(p'', \mathcal{D})) = \begin{cases} [0, 1/2 - \epsilon) & \text{if } acc(p'', \mathcal{D}) < 1/2 - \epsilon, \\ (1/2 + \epsilon, 1] & \text{if } acc(p'', \mathcal{D}) > 1/2 + \epsilon \text{ and} \\ acc(p'', \mathcal{D}) & \text{otherwise.} \end{cases}$$

□

Frequency simplifications are useful also in condensing collections of frequent patterns. For an example of this, see Section 3.3. Other potential applications are speeding up the pattern discovery algorithms, hiding confidential information about the data from the pattern users, correcting or indicating errors in data and in frequent patterns, and examining the stability of the collection of frequent patterns.

Although the frequency simplifications in general may require a considerable amount of interaction, defining simple mappings from the unit interval $[0, 1]$ to a collection of its subintervals and applying the simplification in pattern discovery is often more tractable than defining complex structural constraints with respect to definability and computational complexity. Here are some examples of simple mappings:

- Points in a subinterval of $[0, 1]$ can be replaced by the subinterval itself.
- The points can be discretized by a given discretization function.
- Affine transformations, logarithms and other mappings can be applied to the points.

Note that the simplification does not have to be applicable to all points in $[0, 1]$ but only to the finite number of different frequencies $fr(p, \mathcal{D})$ of the patterns at hand.

The frequency simplifications have clearly certain limitations, as they focus just on frequencies, neglecting the structural aspects of the patterns and the pattern collection (although the structure of the pattern collection can be taken into account indirectly when

defining the simplification). For example, sometimes interesting and uninteresting patterns can have the same frequency. Nevertheless, the frequency simplifications can be useful in constrained pattern discovery as a complementary approach to structural constraints. Furthermore, the simplifications could be used to aid in the search for advantageous constraints by revealing properties that cannot be expressed by the frequencies.

3.2 Discretizing Frequencies

Discretization is an important special case of simplifying frequencies. In general, discretizations are used especially for two purposes: reducing noise and decreasing the size of the representation. As an example of these, let us look at k-means clusterings.

Example 3.4 (k-means clustering). The k-means clustering of a (finite) point set $P \subseteq \mathbb{R}^d$ tries to find a set O of k points in $\mathbb{R}^d$ that minimize the cost

$$\sum_{p \in P} \min_{o \in O} \sum_{i=1}^{d} (p_i - o_i)^2 .$$

This objective can be interpreted as trying to find the centers of k Gaussian distributions that would be the most likely to generate the point set P. Thus, each point in P can be considered as a cluster center plus some Gaussian noise.

The representation of the set P by the cluster centers is clearly smaller than the original point set P. Furthermore, if the centers of the Gaussian distributions are far enough from each other, then the points in P can be encoded in smaller space by expressing for each point $p \in P$ the cluster $o \in O$ where it belongs and the vector $p - o$.

Note that in practice, the k-means clusterings are not always correct ones, even if the assumption of k Gaussian distributions generating the set P is true, because the standard algorithm used for k-means clustering (known as *the k-means algorithm*) is a greedy heuristic. Furthermore, even if it were known to which cluster each of the points in P belongs to, the points in each cluster rarely

provide the correct estimate for the cluster center. (For more details on k-means clustering, see e.g. [HMS01, HTF01].) □

A discretization of frequencies can be defined as follows:

Definition 3.1 (discretization of frequencies). A *discretization of frequencies* is a mapping γ from $[0,1]$ to a (finite) subset of $[0,1]$ that preserves the order of the points. That is, if $x, y \in [0,1]$ and $x \leq y$ then $\gamma(x) \leq \gamma(y)$. Points in the range of the discretization function γ are called the *discretization points* of γ.

Example 3.5 (discretization of frequencies). Probably the simplest example of discretization functions is the mapping γ that maps all frequencies in $[0,1]$ to some constant $c \in [0,1]$. Clearly, such γ is a mapping from $[0,1]$ to a finite subset of $[0,1]$ and $x \leq y \Rightarrow \gamma(x) \leq \gamma(y)$ for all $x, y \in [0,1]$. □

One often very important requirement for a good discretization function is that it should not introduce much error, i.e., the discretized values should not differ too much from the original values. In the next subsections we prove data-independent bounds for the errors in accuracies of association rules with respect to certain discretization functions of frequencies and give algorithms to minimize the empirical loss of several loss functions.

To simplify the considerations, the frequencies of the patterns are assumed to be strictly positive for the rest of the chapter.

3.2.1 Loss Functions for Discretization

The loss functions considered in this section are absolute error and approximation ratio.

The absolute error for a point $x \in (0,1]$ with respect to a discretization function γ is

$$\ell_a(x, \gamma) = |x - \gamma(x)|$$

and the maximum absolute error with respect to a discretization function γ for a finite set $P \subset (0,1]$ of points is

$$\ell_a(P, \gamma) = \max_{x \in P} \ell_a(x, \gamma). \tag{3.1}$$

In addition to the absolute error, also the relative error, i.e., the approximation ratio is often used to evaluate goodness of the approximation. The approximation ratio for a point $x \in (0, 1]$ is

$$\ell_r(x, \gamma) = \frac{\gamma(x)}{x}$$

and the maximum approximation ratio interval with respect to a discretization function for a finite set $P \subset (0, 1]$ is

$$\ell_r(P, \gamma) = \left[\min_{x \in P} \ell_r(x, \gamma), \max_{x \in P} \ell_r(x, \gamma)\right]. \tag{3.2}$$

Let $\ell(x, \gamma)$ denote the loss for a point $x \in P$ with respect to a given discretization γ. Sometimes the most appropriate error for a point set is not the maximum error $\max_{x \in P} \ell(x, \gamma)$ but a weighted sum of the errors of the points in P. If the weight function is $w : P \to \mathbb{R}$ then the weighted sum of errors is

$$\ell_w(P, \gamma) = \sum_{x \in P} w(x)\ell(x, \gamma). \tag{3.3}$$

In the next few subsections we derive efficient algorithms for minimizing these loss functions defined by Equation 3.1, Equation 3.2 and Equation 3.3.

3.2.2 Data-Independent Discretization

In this subsection we show that the discretization functions

$$\gamma_a^\epsilon(x) = \epsilon + 2\epsilon \left\lfloor \frac{x}{2\epsilon} \right\rfloor \tag{3.4}$$

and

$$\gamma_r^\epsilon(x) = (1 - \epsilon)^{1+2\lfloor (\ln x)/(2 \ln(1-\epsilon)) \rfloor} \tag{3.5}$$

are the worst case optimal discretization functions with respect to the maximum absolute error and the maximum approximation ratio interval, respectively. Furthermore, we bound the maximum absolute error and the intervals for approximation ratios for the accuracies of association rules computed using the discretized frequencies.

Let us first study the optimality of the discretization functions. The discretization function γ_a^ϵ is optimal in the following sense:

Theorem 3.1. *Let $P \subset (0, 1]$ be a finite set. Then*

$$\ell_a(P, \gamma_a^\epsilon) \leq \epsilon.$$

Furthermore, for any other data-independent discretization function γ with less discretization points, $\ell(P', \gamma) > \epsilon$ for some point set $P' \subset (0, 1]$ such that $|P| = |P'|$.

Proof. For any point $x \in (0, 1]$, the absolute error $\ell_a(x, \gamma_a^\epsilon)$ with respect to the discretization function γ_a^ϵ is at most ϵ since

$$2\epsilon \left\lfloor \frac{x}{2\epsilon} \right\rfloor \leq x < 2\epsilon + 2\epsilon \left\lfloor \frac{x}{2\epsilon} \right\rfloor$$

and

$$\gamma_a^\epsilon(x) = \epsilon + 2\epsilon \left\lfloor \frac{x}{2\epsilon} \right\rfloor.$$

Any discretization function γ can be considered as a collection γ^{-1} of intervals covering the interval $(0, 1]$. Each discretization point can cover an interval of length at most 2ϵ when the maximum absolute error is allowed to be at most ϵ. Thus, at least $\lceil 1/(2\epsilon) \rceil$ discretization points are needed to cover the whole interval $(0, 1]$. The discretization function γ_a^ϵ uses exactly that number of discretization points. □

It can be observed from the proof of Theorem 3.1 that some maximum error bounds ϵ are unnecessary high. Thus, the maximum absolute error bound ϵ can be decreased without increasing the number of discretization points.

Corollary 3.1. *The bound ϵ for the maximum absolute error can be decreased to*

$$\frac{1}{2\left\lceil \frac{1}{2\epsilon} \right\rceil}$$

without increasing the number of discretization points when discretizing by the function γ_a^ϵ.

The worst case optimality of the discretization function γ_r^ϵ can be shown as follows:

Theorem 3.2. *Let $P \subset (0, 1]$ be a finite set. Then*

$$\ell_r(P, \gamma_r^\epsilon) \subseteq \left[1 - \epsilon, \frac{1}{1 - \epsilon}\right].$$

Furthermore, for any other data-independent discretization function γ with less discretization points we have

$$\ell(P', \gamma) \not\subseteq \left[1 - \epsilon, \frac{1}{1-\epsilon}\right]$$

for some point set $P' \subset (0, 1]$ such that $|P| = |P'|$.

Proof. Clearly,

$$\left\lfloor \frac{\ln x}{2 \ln (1-\epsilon)} \right\rfloor \leq \frac{\ln x}{2 \ln (1-\epsilon)} \leq 1 + \left\lfloor \frac{\ln x}{2 \ln (1-\epsilon)} \right\rfloor$$

holds for all $x > 0$ and we can write

$$x = (1-\epsilon)^{(\ln x)/(\ln(1-\epsilon))} = (1-\epsilon)^{2(\ln x)/(2\ln(1-\epsilon))}.$$

Thus,

$$\begin{aligned} 1 - \epsilon &= \frac{(1-\epsilon)^{1+2(\ln x)/(2\ln(1-\epsilon))}}{x} \\ &\leq \frac{(1-\epsilon)^{1+2\lfloor(\ln x)/(2\ln(1-\epsilon))\rfloor}}{x} \\ &= \frac{(1-\epsilon)^{-1+2+2\lfloor(\ln x)/(2\ln(1-\epsilon))\rfloor}}{x} \\ &\leq \frac{(1-\epsilon)^{-1+2(\ln x)/(2\ln(1-\epsilon))}}{x} = \frac{1}{1-\epsilon}. \end{aligned}$$

The discretization function γ_r^ϵ is the worst case optimal for any interval $[x, 1] \subset (0, 1]$, since it defines a partition of $[x, 1]$ with maximally long intervals. □

Furthermore, the discretization function with the maximum absolute and the maximum relative approximation error guarantees gives guarantees for the maximum relative and the maximum absolute errors, respectively, as follows.

Theorem 3.3. *A discretization function with the maximum absolute error ϵ guarantees that a discretization of a point $x \in (0, 1]$ has the relative error in the interval $[1 - \epsilon/x, 1 + \epsilon/x]$.*

Proof. By definition, the minimum and the maximum discretization errors of a discretization function with the maximum absolute error at most ϵ are $(x - \epsilon)/x = 1 - \epsilon/x$ and $(x + \epsilon)/x = 1 + \epsilon/x$. □

Theorem 3.4. *A discretization function with the maximum relative error in the interval* $[1-\epsilon, 1+\epsilon]$ *guarantees that a point* $x \in (0,1]$ *has the maximum absolute error at most* ϵx*.*

Proof. The discretization $\gamma(x)$ of x with the maximum relative error in the interval $[1-\epsilon, 1+\epsilon]$ is in the interval $[(1-\epsilon)\,x, (1+\epsilon)\,x]$. Thus, the maximum absolute error is

$$\max\{x - (1-\epsilon)\,x, (1+\epsilon)\,x - x\} = \epsilon x$$

as claimed. □

An important use of frequent patterns is to discover accurate association rules. Thus, it would be very useful to be able to bound the errors for the accuracies of the association rules. For simplicity, we consider association rules over itemsets although all following results hold for any pattern collections and quality values.

Let us first study how well the maximum absolute error guarantees for frequency discretizations transfer to the maximum absolute error guarantees for the accuracies of association rules.

Theorem 3.5. *Let* γ^ϵ *be a discretization function with the maximum absolute error* ϵ*. The maximum absolute error for the accuracy of the association rule* $X \Rightarrow Y$ *when the frequencies* $fr(X \cup Y, \mathcal{D})$ *and* $fr(X, \mathcal{D})$ *are discretized by* γ^ϵ *is at most*

$$\min\left\{1, \frac{2\epsilon}{fr(X, \mathcal{D})}\right\}.$$

Proof. By definition, a discretization function preserves the order of points in the discretizations. Because $fr(X \cup Y, \mathcal{D}) \leq fr(X, \mathcal{D})$, we have $\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) \leq \gamma^\epsilon(fr(X, \mathcal{D}))$.

Since the correct accuracies are always in the interval $[0,1]$, the maximum absolute error is at most 1.

The two extreme cases are

1. when $fr(X \cup Y, \mathcal{D}) = fr(X, d) - \delta > 0$ for arbitrary small $\delta > 0$, but $\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = fr(X \cup Y, \mathcal{D}) - \epsilon$ and $\gamma^\epsilon(fr(X, \mathcal{D})) = fr(X \cup Y, \mathcal{D}) + \epsilon$, and

2. when $fr(X \cup Y, \mathcal{D}) = fr(X, d) - 2\epsilon + \delta > 0$ for arbitrary small $\delta > 0$, but $\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = \gamma^\epsilon(fr(X, \mathcal{D}))$.

In the first case, the worst case absolute error is at most

$$\begin{aligned}
& \left| \frac{fr(X \cup Y, \mathcal{D}) - \epsilon}{fr(X \cup Y, \mathcal{D}) + \epsilon} - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| \\
\leq & \left| \frac{fr(X \cup Y, \mathcal{D}) - \epsilon}{fr(X, \mathcal{D}) + \epsilon} - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| \\
= & \frac{\epsilon fr(X, \mathcal{D}) + \epsilon fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})^2 + \epsilon fr(X, \mathcal{D})} \\
\leq & \frac{2\epsilon fr(X, \mathcal{D})}{fr(X, \mathcal{D})^2 + \epsilon fr(X, \mathcal{D})} \\
\leq & \frac{2\epsilon}{fr(X, \mathcal{D}) + \epsilon}.
\end{aligned}$$

In the second case, the absolute error in the worst case is at most

$$1 - \frac{fr(X, \mathcal{D}) - 2\epsilon}{fr(X, \mathcal{D})} = \frac{2\epsilon}{fr(X, \mathcal{D})}$$

when $fr(X, \mathcal{D}) \geq 2\epsilon$.

Thus, the second case is larger and gives the upper bound. □

Note that in the worst case the maximum absolute error can indeed be 1 as shown by Example 3.6.

Example 3.6 (the tightness of the bound for γ_a^ϵ and any $\epsilon > 0$). Let $fr(X \cup Y, \mathcal{D}) = \delta$ and $fr(X, \mathcal{D}) = 2\epsilon - \delta$. Then $\gamma_a^\epsilon(fr(X \cup Y, \mathcal{D})) = \gamma_a^\epsilon(fr(X, \mathcal{D})) = \epsilon$. Thus,

$$\left| 1 - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| = \left| 1 - \frac{\delta}{2\epsilon - \delta} \right| \to 1$$

when $\delta \to 0$. □

When the maximum absolute error for the frequency discretization function is bounded, also the maximum relative error for the accuracies of the association rules computed from discretized and original frequencies can bounded as follows:

Theorem 3.6. *Let γ^ϵ be a discretization function with the maximum absolute error ϵ. The approximation ratio for the accuracy of the association rule $X \Rightarrow Y$, when the frequencies $fr(X \cup Y, \mathcal{D})$ and $fr(X, \mathcal{D})$ are discretized using the function γ^ϵ, is in the interval*

$$\left[\max\left\{0, \frac{fr(X \cup Y, \mathcal{D}) - \epsilon}{fr(X \cup Y, \mathcal{D}) + \epsilon}\right\}, \frac{fr(X, \mathcal{D})}{fr(X \cup Y, \mathcal{D})} \right].$$

Proof. The smallest approximation ratio is obtained when $fr(X, \mathcal{D}) = fr(X \cup Y, \mathcal{D}) + \delta$ where δ is an arbitrary small positive value, but $\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = fr(X \cup Y, \mathcal{D}) - \epsilon$ and $\gamma^\epsilon(fr(X, \mathcal{D})) = fr(X, \mathcal{D}) + \epsilon$. Then the approximation ratio is

$$\begin{aligned} & \frac{fr(X \cup Y, \mathcal{D}) - \epsilon}{fr(X, \mathcal{D}) + \epsilon} \left(\frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right)^{-1} \\ = & \frac{fr(X \cup Y, \mathcal{D}) fr(X, \mathcal{D}) - \epsilon fr(X, \mathcal{D})}{fr(X \cup Y, \mathcal{D}) fr(X, \mathcal{D}) + \epsilon fr(X \cup Y, \mathcal{D})} \\ = & \frac{fr(X \cup Y, \mathcal{D})^2 + \delta fr(X \cup Y, \mathcal{D}) - \epsilon fr(X \cup Y, \mathcal{D}) - \delta\epsilon}{fr(X \cup Y, \mathcal{D})^2 + \delta fr(X \cup Y, \mathcal{D}) + \epsilon fr(X \cup Y, \mathcal{D})}. \end{aligned}$$

If $\delta \to 0$, then

$$\frac{fr(X \cup Y, \mathcal{D}) - \epsilon}{fr(X, \mathcal{D}) + \epsilon} \left(\frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right)^{-1} \to \frac{fr(X \cup Y, \mathcal{D}) - \epsilon}{fr(X \cup Y, \mathcal{D}) + \epsilon}.$$

Note that in that inequality, we assume that $fr(X, \mathcal{D}) \geq fr(X \cup Y, \mathcal{D}) \geq \epsilon$ because, by Definition 3.1, all discretized values are non-negative. Hence, we get the claimed lower bound.

By the definition of the approximation ratio, the upper bound is obtained when $fr(X, \mathcal{D}) \neq fr(X \cup Y, \mathcal{D})$ but $\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = \gamma^\epsilon(fr(X, \mathcal{D}))$. The greatest approximation ratio is obtained when $fr(X \cup Y, \mathcal{D}) = fr(X, \mathcal{D}) - 2\epsilon + \delta$ for arbitrary small $\delta > 0$ but $\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = \gamma^\epsilon(fr(X, \mathcal{D}))$. Then the approximation ratio is

$$\frac{\gamma^\epsilon(fr(X \cup Y, \mathcal{D}))}{\gamma^\epsilon(fr(X, \mathcal{D}))} \left(\frac{fr(X, \mathcal{D}) - 2\epsilon + \delta}{fr(X, \mathcal{D})} \right)^{-1} \to \frac{fr(X, \mathcal{D})}{fr(X, \mathcal{D}) - 2\epsilon}$$

when $\delta \to 0$. If $fr(X, \mathcal{D}) \to 2\epsilon$, then the ratio increases unboundedly. □

The worst case the relative error bounds for the discretization function γ_a^ϵ are the following.

Example 3.7 (the worst case relative error bounds of γ_a^ϵ). The smallest ratio is achieved when $fr(X, \mathcal{D}) = 2k\epsilon$ and $fr(X \cup Y, \mathcal{D}) = 2k\epsilon - \delta$ for arbitrary small $\delta > 0$ and some $k \in \{1, \ldots, \lfloor 1/\epsilon \rfloor\}$. The ratio

$$\frac{(2k-1)\epsilon / (2k+1)\epsilon}{(2k\epsilon - \delta)/2\epsilon}$$

is minimized by choosing $k = 1$. Thus, the lower bound for the relative error is $1/3$.

The relative error cannot be bounded above since the frequencies $fr(X, \mathcal{D}) = 2\epsilon - \delta$ and $fr(X \cup Y, \mathcal{D}) = \delta$ with discretizations $\gamma(fr(X, \mathcal{D})) = \gamma(fr(X \cup Y, \mathcal{D}))$ give the ratio

$$\frac{\epsilon/\epsilon}{\delta/(2\epsilon - \delta)} = \frac{2\epsilon}{\delta} - 1 \to \infty$$

when $\delta \to 0$ and $\epsilon > 0$. □

The relative error for the accuracies of the association rules can be bounded much better when discretizing by the discretization function γ^ϵ having the approximation ratio guarantees instead of the maximum absolute error guarantees.

Theorem 3.7. *Let γ^ϵ be a discretization function with the approximation ratio in the interval $\left[(1-\epsilon), (1-\epsilon)^{-1}\right]$. The approximation ratio for the accuracy of the association rule $X \Rightarrow Y$ when the frequencies $fr(X \cup Y, \mathcal{D})$ and $fr(X, \mathcal{D})$ are discretized by γ^ϵ is in the interval*

$$\left[(1-\epsilon)^2, (1-\epsilon)^{-2}\right].$$

Proof. By choosing

$$\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = (1-\epsilon)\, fr(X \cup Y, \mathcal{D})$$

and

$$\gamma^\epsilon(fr(X, \mathcal{D})) = (1-\epsilon)^{-1}\, fr(X, \mathcal{D})$$

we get

$$\frac{(1-\epsilon)\, fr(X \cup Y, \mathcal{D})}{(1-\epsilon)^{-1}\, fr(X, \mathcal{D})} = (1-\epsilon)^2 \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})}.$$

By choosing

$$\gamma^\epsilon(fr(X \cup Y, \mathcal{D})) = (1-\epsilon)\, fr(X \cup Y, \mathcal{D})$$

and

$$\gamma^\epsilon(fr(X, \mathcal{D})) = (1-\epsilon)^{-1}\, fr(X, \mathcal{D})$$

we get

$$\frac{(1-\epsilon)^{-1}\, fr(X \cup Y, \mathcal{D})}{(1-\epsilon)\, fr(X, \mathcal{D})} = (1-\epsilon)^{-2} \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})}.$$

It is easy to see that these are the worst case instances. □

Note that these bounds are tight also for the discretization function γ_r^ϵ.

The discretization functions with the maximum absolute error guarantees give also some guarantees for the approximation ratios of accuracies.

Theorem 3.8. *Let γ^ϵ be a discretization function with the approximation ratio in the interval $\left[(1-\epsilon), (1-\epsilon)^{-1}\right]$. Then the maximum absolute error for the accuracy of the association rule $X \Rightarrow Y$ when the frequencies $fr(X \cup Y, \mathcal{D})$ and $fr(X, \mathcal{D})$ are discretized by γ^ϵ is at most*

$$1 - (1-\epsilon)^2 = 2\epsilon(1-\epsilon).$$

Proof. There are two extreme cases. First, the frequencies $fr(X, \mathcal{D})$ and $fr(X \cup Y, \mathcal{D})$ can be almost equal but be discretized as far as possible from each other, i.e.,

$$\begin{aligned} & \left| \frac{(1-\epsilon)\, fr(X \cup Y, \mathcal{D})}{(1-\epsilon)^{-1} fr(X, \mathcal{D})} - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| \\ = & \left| (1-\epsilon)^2 \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| \\ = & \left(1 - (1-\epsilon)^2\right) \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \end{aligned}$$

The maximum value is achieved by setting $fr(X \cup Y, \mathcal{D}) = fr(X, \mathcal{D}) - \delta$ for arbitrary small $\delta > 0$.

In the second case, the frequencies $fr(X, \mathcal{D})$ and $fr(X \cup Y, \mathcal{D})$ are discretized to have the same value although they are as apart from each other as possible. That is,

$$\begin{aligned} & \left| \frac{(1-\epsilon)^{-1} fr(X \cup Y, \mathcal{D})}{(1-\epsilon)\, fr(X, \mathcal{D})} - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| \\ = & \left| \frac{fr(X \cup Y, \mathcal{D})}{(1-\epsilon)^2 fr(X, \mathcal{D})} - \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \right| \\ = & \left(\frac{1}{(1-\epsilon)^2} - 1 \right) \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})} \\ = & \frac{1 - (1-\epsilon)^2}{(1-\epsilon)^2} \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})}. \end{aligned}$$

However, in that case $fr(X \cup Y, \mathcal{D}) \leq (1-\epsilon)^2 fr(X, \mathcal{D})$. Thus, the maximum absolute error is again at most $1 - (1-\epsilon)^2$. □

In this subsection we have seen that data-independent discretization of frequencies with approximation guarantees can provide approximation guarantees also for the accuracies of the association rules computed from the discretized frequencies without any *a priori* information about the frequencies (especially when the frequencies are discretized using a discretization function with the approximation ratio guarantees).

3.2.3 Data-Dependent Discretization

In practice, taking the actual data into account usually improves the performance of the approximation methods. Thus, it is natural to consider also data-dependent discretization techniques. The problem of discretizing frequencies by taking the actual frequencies into account can be formulated as a computational problem as follows:

Problem 3.1 (frequency discretization). Given a finite subset P of $(0, 1]$, a maximum error threshold ϵ and a loss function ℓ, find a discretization γ for P such that $|\gamma(P)|$ is minimized and the error $\ell(P, \gamma)$ is at most ϵ.

Example 3.8 (frequency discretization). Let the set $P \subset (0, 1]$ consist of points 1/10, 3/10, 7/10 and 9/10, let the maximum error threshold ϵ be 1/10, and let the loss function ℓ be the maximum absolute error (Equation 3.1).

Then the discretization function γ with smallest number of discretization points and maximum absolute error at most ϵ is the mapping

$$\gamma = \left\{ \frac{1}{10} \mapsto \frac{1}{5}, \frac{3}{10} \mapsto \frac{1}{5}, \frac{7}{10} \mapsto \frac{4}{5}, \frac{9}{10} \mapsto \frac{4}{5} \right\}.$$

If $\epsilon = 1/9$ instead, then there are several mappings with the maximum absolute error at most ϵ and two discretization points. Namely, all mappings

$$\gamma = \left\{ \frac{1}{10} \mapsto a, \frac{3}{10} \mapsto a, \frac{7}{10} \mapsto b, \frac{9}{10} \mapsto b \right\}$$

where $a \in [1/5 - 1/90, 1/5 + 1/90]$ and $b \in [4/5 - 1/90, 4/5 + 1/90]$. □

In this subsection we derive sub-quadratic algorithms for discretizing with respect to the maximum absolute error and polynomial-time solutions for also many other classes of loss functions.

Maximum absolute error

A discretization of a point set $P \subseteq (0, 1]$ without exceeding the maximum absolute error ϵ can be interpreted as an interval cover of the point set P with intervals of length 2ϵ, i.e., a collection of length 2ϵ sub-intervals of $[0, 1]$ that together cover all points in P.

A simple solution for the frequency discretization problem with the loss function being the maximum absolute error is to repeatedly choose the minimum uncovered point $d \in P$ and discretize all the previously uncovered points of P in the interval $[d, d + 2\epsilon]$ to the value $d + \epsilon$. This is described as Algorithm 3.1.

Algorithm 3.1 A straightforward algorithm for discretization with respect to the maximum absolute error.

Input: A finite set $P \subset [0, 1]$ and a real value $\epsilon \in [0, 1]$.
Output: A discretization function γ with $\ell_a(P, \gamma) \leq \epsilon$.

```
1:  function IntervalCover(P, ε)
2:      while P ≠ ∅ do
3:          d ← min P
4:          I ← {x ∈ P : d ≤ x ≤ d + 2ε}
5:          for all x ∈ I do
6:              γ(x) ← d + ε
7:          end for
8:          P ← P \ I
9:      end while
10:     return γ
11: end function
```

Theorem 3.9. *Algorithm 3.1 finds a discretization function γ such that the error $\ell_a(P, \gamma)$ is at most ϵ and for all discretizations γ' with a smaller number of discretization points than $|\gamma(P)|$ the error $\ell_a(P, \gamma')$ is greater than ϵ.*

Proof. The maximum absolute error is at most ϵ since all points are covered by intervals of length 2ϵ and the distance to the center of any covering interval is at most ϵ.

To see that a smaller number of discretization points would have a larger error, let $x_1, \ldots, x_m$ be the discretization points of the discretization γ found by Algorithm 3.1 for the point set P. By construction, there is a point $x_i - \epsilon \in P$ for each $1 \leq i \leq m$. Furthermore, $|x_i - x_j| > 2\epsilon$ for all discretization points x_i and x_j of γ such that $1 \leq i < j \leq m$, since otherwise the point $x_j - \epsilon \in P$ is contained in the interval $[x_i - \epsilon, x_i + \epsilon]$ or the point $x_i - \epsilon \in P$ is contained in the interval $[x_j - \epsilon, x_j + \epsilon]$. Thus, no two points $x_i - \epsilon, x_j - \epsilon \in P$ such that $1 \leq i < j \leq m$ can share the same discretization point x_k where $1 \leq k \leq m$. □

The straightforward implementation of Algorithm 3.1 runs in time $\mathcal{O}(|P|^2)$. The bound is tight in the worst case as shown by Example 3.9.

Example 3.9 (The worst case running time of Algorithm 3.1). Let $P = \{1/|P|, 2/|P|, \ldots, 1 - 1/|P|, 1\}$ and $\epsilon < 1/(2|P|)$. Then at each iteration only one point is removed but all other points are inspected. There are $|P|$ iterations and the iteration i takes time $\mathcal{O}(i)$. Thus, the total time complexity is $\mathcal{O}(|P|^2)$. □

In the special case of ϵ being a constant, the time complexity of the algorithm is linear in $|P|$ because each iteration takes at most time $\mathcal{O}(|P|)$ and there can be at most constant number of iterations: At each iteration, except possibly the last one, at least length 2ϵ subinterval of $[0, 1]$ is covered. Thus, the number of iterations can be bounded above by $\lceil 1/(2\epsilon) \rceil = \mathcal{O}(1)$ and the total time needed is $\mathcal{O}(|P|)$.

The worst case time complexity of the algorithm can be reduced to $\mathcal{O}(|P| \log |P|)$ by constructing a heap for the point set P. A minimum element in the heap can be found in constant time and insertions and deletions to the heap can be done in time logarithmic in $|P|$ [Knu98].

The time complexity $\mathcal{O}(|P| \log |P|)$ is not optimal, especially if preprocessing of the point set P is allowed. For example, if the set P is represented as a sorted array, i.e., an array P such that $P[i] \leq P[j]$ for all $1 \leq i < j \leq |P|$, then the problem can be solved in linear time in $|P|$ by Algorithm 3.2.

The efficiency of Algorithm 3.2 depends crucially on the efficiency of sorting. In the worst case sorting real-valued points takes

Algorithm 3.2 A linear-time algorithm for discretizing a sorted point set with respect to maximum absolute error.

Input: A finite set $P \subset [0,1]$ as an array in ascending order and a real value $\epsilon \in [0,1]$.
Output: A discretization function γ with $\ell_a(P,\gamma) \leq \epsilon$.

```
1: function PREFIX-COVER(P, ε)
2:     for i = 1, ..., |P| do
3:         if d < P[i] − ε then
4:             d ← P[i] + ε
5:         end if
6:         γ(P[i]) ← d
7:     end for
8:     return γ
9: end function
```

time $\mathcal{O}(|P| \log |P|)$ but sometimes, for example when the points are almost in order, the points can be sorted faster. For example, the frequent itemset mining algorithm APRIORI [AMS+96] finds the frequent itemsets in partially descending order in their frequencies. Note that also the generalization of the algorithm APRIORI, the levelwise algorithm (Algorithm 2.2) can easily be implemented in such a way that it outputs frequent patterns in descending order in frequencies.

However, it is possible to find in time $\mathcal{O}(|P|)$ a discretization function with maximum absolute error at most ϵ and the minimum number of discretization points, even if the points in P are not in ordered in some specific way in advance. This can be done by first discretizing the frequencies using the discretization function γ_a^ϵ (Equation 3.4) and then repairing the discretization. The high-level idea of the algorithm is as follows:

1. Put the points in P into bins $0, 1 \ldots, \lfloor 1/(2\epsilon) \rfloor$ corresponding to intervals $(0, 2\epsilon], (2\epsilon, 4\epsilon], \ldots, (2\epsilon \lfloor 1/(2\epsilon) \rfloor, 1]$. Let B be the set of bins such that $B[i]$ corresponds to bin i.

2. Find a minimal non-empty bin i in B. (A non-empty bin i is called minimal if $i = 0$ or the bin $i - 1$ is empty.)

3. Find the smallest point x in the bin i, replace the interval corresponding to the bin i by interval $[x, x + 2\epsilon]$ and move

the points of the bin $i+1$ that are in the interval $[x, x+2\epsilon]$ into the bin i.

4. Remove bin i from B.

5. Go to step 2 if there are still non-empty bins.

The algorithm can be implemented to run in linear time in $|P|$: The discretization to bins can be computed in time $\mathcal{O}(|P|)$ using a hash table for the set B [Knu98]. A minimal non-empty bin can be found in amortized constant time by processing the consecutive runs of non-empty bins consecutively.

If the points in P are given in an arbitrary order, then Algorithm 3.3 is asymptotically optimal for minimizing the number of discretization points with respect to the given maximum absolute discretization error threshold ϵ as shown by Theorem 3.10.

Theorem 3.10. *No (deterministic) algorithm can find a discretization γ with the minimum number of discretization points without inspecting all points in $P \subset (0,1]$ when $2\epsilon + \delta \leq 1$ for any $\delta > 0$.*

Proof. Let P consist of points in the interval $(0, \delta)$ and possibly the point 1. Furthermore, let the points examined by the algorithm be in the interval $(0, \delta)$. Based on that information, the algorithm cannot decide for sure whether or not the point 1 is in P. □

If the set P is given in ascending or descending order, however, then it is possible to find a set $\gamma(P)$ of discretization points of minimum cardinality among those that determine a discretization of P with the maximum absolute error at most ϵ, in time $\mathcal{O}(|\gamma(P)| \log |P|)$ see Algorithm 3.4. Although $\gamma(P)$ is only an implicit representation of the discretization function $\gamma : P \to \gamma(P)$, the discretization of any $x \in P$ can be found in time $\mathcal{O}(\log |\gamma(P)|)$ if the set $\gamma(P)$ is represented, e.g., as a sorted array.

Note that the proposed techniques for discretizing with respect to the maximum absolute error guarantees (i.e., Algorithms 3.1, 3.2, 3.3 and 3.4) generalize to maximum error functions that are strictly increasing transformations of the maximum absolute error function. Furthermore, the algorithms can be modified to minimize the maximum absolute error instead of the number of discretization points by a simple application of binary search.

Algorithm 3.3 A linear-time algorithm for discretization with respect to maximum absolute error.

Input: A finite set $P \subset [0,1]$ and a real value $\epsilon \in [0,1]$.
Output: A discretization function γ with $\ell_a(P,\gamma) \leq \epsilon$.

```
1: function BIN-COVER(P, ε)
2:     for all x ∈ P do
3:         i ← ⌊x/(2ε)⌋
4:         B[i] ← B[i] ∪ {x}
5:     end for
6:     for all B[i] ∈ B, B[i] ≠ ∅ do
7:         while i > 0 and B[i − 1] ≠ ∅ do
8:             i ← i − 1
9:             d ← min B[i]
10:        end while
11:        while B[i] ≠ ∅ do
12:            I ← {x ∈ B[i] : d ≤ x ≤ d + 2ε}
13:            for all x ∈ I do
14:                γ(x) ← d + ε
15:            end for
16:            B[i] ← B[i] \ I
17:            if min B[i + 1] < d + 2ε then
18:                i ← i + 1
19:            else
20:                d ← min B[i]
21:            end if
22:        end while
23:    end for
24:    return γ
25: end function
```

Weighted sums of errors

Sometimes it would be more natural to valuate the quality of discretizations using a weighted sum

$$\sum_{x \in P} w(x)\ell(x,\gamma)$$

of errors $\ell(x,\gamma)$ instead of the maximum error $\max_{x \in P} \ell(x,\gamma)$. In that case, the algorithms described previously in this chapter do not

Algorithm 3.4 A sublinear-time algorithm for discretization a sorted point set with respect to the maximum absolute error.

Input: A finite set $P \subseteq [0,1]$ as an array in ascending order and a real value $\epsilon \in [0,1]$.

Output: A discretization points $\gamma(P)$ with $\ell_a(P,\gamma) \leq \epsilon$.

```
 1: function LOG-COVER(P, ε)
 2:     i ← 1
 3:     while i ≤ |P| do
 4:         d ← P[i] + ε
 5:         γ(P) ← γ(P) ∪ {d}
 6:         j ← |P| + 1
 7:         while j > i + 1 do
 8:             k ← ⌊(i + j) /2⌋
 9:             if P[k] ≤ d + ε then
10:                 i ← k
11:             else
12:                 j ← k
13:             end if
14:         end while
15:         i ← j
16:     end while
17:     return γ(P)
18: end function
```

find the optimal solutions. Fortunately, the problem can be solved optimally in time polynomial in $|P|$ by dynamic programming; see e.g. [Fis58, JKM+98].

To describe the solution, we have to first define some notation. Let the point set P be represented as an array in ascending order, i.e., $P[i] \leq P[j]$ for all $1 \leq i < j \leq |P|$, and let $P[i,j]$ denote the subarray $P[i] \ldots P[j]$. The best discretization point to represent the array $P[i,j]$ is denoted by $\mu_{i,j}$ and its error by $\varepsilon_{i,j}$. The loss of the best discretization $P[1,i]$ with k discretization points with respect to the sum of errors is denoted by Δ_i^k and the $k-1$th discretization point in that discretization is denoted by ω_i^k.

The optimal error for $P[1,i]$ using k discretization points can be

defined by the following recursive formula:

$$\Delta_i^k = \begin{cases} \varepsilon_{1,i} & \text{if } k = 1 \text{ and} \\ \min_{k \le j \le i} \left\{ \Delta_{j-1}^{k-1} + \varepsilon_{j,i} \right\} & \text{otherwise.} \end{cases}$$

The optimal sum-of-errors discretization by dynamic programming can be divided into two subtasks:

1. Compute the matrices μ of discretization points and ε of their errors: $\mu_{i,j}$ is the discretization point for the subset $P[i,j]$ and $\varepsilon_{i,j}$ is its error.

2. Find the optimal discretizations for $P[1,i]$ with k discretization points for all $1 \le k \le i \le |P|$ from the matrices μ and ε using dynamic programming.

The optimal discretization function for P can be found from any matrix $\varepsilon \in \mathbb{R}^{|P| \times |P|}$ of errors and any matrix $\mu \in \mathbb{R}^{|P| \times |P|}$ of discretization points (although not all matrices ε and μ make sense nor are they computable). For example, the matrices can be given by an expert.

Simple examples of error and discretization point matrices computable in polynomial time in $|P|$ are the matrices ε and μ for the weighted sums of absolute errors. They can be computed in time $\mathcal{O}(|P|^3)$ as described by Algorithm 3.5. (Function MEDIAN computes the weighted median of $P[i,j]$.)

The discretization points $\mu_{i,j}$ and the errors $\varepsilon_{i,j}$ of $P[i,j]$ for all $1 \le i \le j \le |P|$ can already be informative summaries of the set P. Besides of that, it is possible to extract from the matrices ε and μ the matrices Δ and ω corresponding to the partial sums of errors and the discretizations. This can be done by Algorithm 3.6. (The matrices Δ and ω determine the optimal discretizations for each number of discretization points and each prefix $P[1,i]$ of P.)

The time complexity of Algorithm 3.6 is $\mathcal{O}(|P|^3)$. The time consumption can be reduced to $\mathcal{O}(k\,|P|^2)$ if we are interested only on discretizations with at most k discretization points. Furthermore, the method can be adapted to other kinds of loss functions, too. For some loss functions, the dynamic programming can be implemented with asymptotically better efficiency guarantees [ER01, JKM+98]. There are several ways to speed up the search in practice. For example, it is not necessary to compute the parts of the matrices that are detected to be not needed in the best solutions.

Algorithm 3.5 An algorithm to compute the loss and discretization matrices ε and μ for the point set P and a weight function w.

Input: A finite set $P \subset [0,1]$ and a weight function $w : P \to \mathbb{R}$.
Output: Matrices ε and μ.

```
1:  function VALUATE-ABS(P, w)
2:      for i = 1, ..., |P| do
3:          for j = i, ..., |P| do
4:              μ_{i,j} ← MEDIAN(P[i, j], w)
5:              ε_{i,j} ← 0
6:              for k = i, ..., j do
7:                  ε_{i,j} ← ε_{i,j} + w(P[k]) |P[k] − μ_{i,j}|
8:              end for
9:          end for
10:     end for
11:     return ⟨ε, μ⟩
12: end function
```

Although the matrices Δ and ω contain the information about the optimal discretizations of all prefixes of P for each number of discretization points, usually the actual goal is to extract the optimal discretizations from these matrices.

The optimal discretizations of k discretization points can be found in time $\mathcal{O}(|P|)$ by Algorithm 3.7. It can be adapted to find discretization with minimum number of discretization points and the error less than ϵ in time linear in $|P|$. Note that if it is sufficient to obtain just the set $\gamma(P)$ of k discretization points, then the task can be conducted in time $\mathcal{O}(k)$ by Algorithm 3.8.

Instead of finding the best discretization with a certain number of discretization points, one could search for a hierarchical discretization suggesting a good discretization of k discretization points for all values of k.

Example 3.10 (hierarchical discretizations). Let the point set P be $\{0.1, 0.2, 0.5, 0.6, 0.9, 1.0\}$ and let us consider hierarchical discretizations with respect to the maximum absolute error. Two standard approaches to define hierarchical clusterings are divisive (or top-down) and agglomerative (or bottom-up) clusterings.

Divisive hierarchical clustering starts from the whole point set

Algorithm 3.6 An algorithm to compute matrices Δ and ω from P, ε and μ.

Input: A finite set $P \subset [0,1]$, and matrices ε and μ.
Output: Matrices Δ and ω.

```
1: function TABULATOR(P, ε, μ)
2:     for all i ∈ {1, ..., |P|} do          ▷ Initialize the errors Δ_i^k.
3:         Δ_i^1 ← ε_{1,i}
4:     end for
5:     for all k, i ∈ {2, ..., |P|}, k ≤ i do
6:         Δ_i^k ← ∞
7:     end for
8:     for k = 1, ..., |P| do          ▷ Find the best discretization of
           P[1, i] with k discretization points.
9:         Δ' ← ∞
10:        for all j, i ∈ {k, ..., |P|}, j ≤ i do
11:            if Δ' < Δ_i^k then
12:                Δ_i^k ← Δ'
13:                ω_i^k ← j − 1
14:            end if
15:        end for
16:    end for
17:    return ⟨Δ, ω⟩
18: end function
```

and recursively divides it in such a way that the division always improves the solution as much as possible. For example, the divisive clustering of P would be the following:

- The first level of the clustering consists of only one cluster, namely $\{0.1, 0.2, 0.5, 0.6, 0.9, 1.0\}$.
- The maximum absolute error is decreased as much as possible by splitting the set into two parts $\{0.1, 0.2, 0.5\}$ and $\{0.6, 0.9, 1.0\}$
- In the third level no split improves the maximum absolute error. However, splitting $\{0.1, 0.2, 0.5\}$ to $\{0.1, 0.2\}$ and $\{0.5\}$, or splitting $\{0.6, 0.9, 1.0\}$ to $\{0.6\}$ and $\{0.9, 1.0\}$ decreases most the maximum absolute error for one of the clusters with the maximum absolute error.

Algorithm 3.7 An algorithm to extract the best discretization of k discretization points from the matrices Δ and ω.

Input: A finite set $P \subset [0,1]$, matrices Δ, μ and ω, and an integer $k \in \{1, \ldots, |P|\}$.

Output: The discretization γ of k discretization points with the smallest error $\Delta^k_{|P|}$.

```
1: function FIND-DISCRETIZATION(P, μ, ω, k)
2:     i ← |P|
3:     for l = k, ..., 1 do
4:         for j = i, ..., ω_i^l + 1 do
5:             γ(P[i]) ← μ_{ω_i^j, i}
6:         end for
7:         i ← ω_i^l
8:     end for
9:     return γ
10: end function
```

Algorithm 3.8 An algorithm to extract the best k discretization points from the matrices Δ and ω.

Input: A finite set $P \subset [0,1]$, matrices Δ, μ and ω, and an integer $k \in \{1, |P|\}$.

Output: The set $\gamma(P)$ of k discretization points with points with the smallest error $\Delta^k_{|P|}$.

```
1: function FIND-DISCRETIZATION-POINTS(P, μ, ω, k)
2:     γ(P) ← ∅
3:     i ← |P|
4:     for l = k, ..., 1 do
5:         j ← ω_i^l + 1
6:         γ(P) ← γ(P) ∪ {μ_{j,i}}
7:         i ← ω_i^l
8:     end for
9:     return γ(P)
10: end function
```

- The fourth level consists of the clusters $\{0.1, 0.2\}$, $\{0.5\}$, $\{0.6\}$, and $\{0.9, 1.0\}$.

- In the fifth level we have again two equally good splitting possibilities: $\{0.1, 0.2\}$ to $\{0.1\}$ and $\{0.2\}$, or $\{0.9, 1.0\}$ to

{0.9} and {1.0}.

- The last level consists of singletons {0.1}, {0.2}, {0.5}, {0.6}, {0.9}, and {1.0}.

Agglomerative hierarchical clustering starts from the singletons and merges the clusters by minimizing the error introduced by the merges. Thus, the agglomerative clustering of would be the following: First level consists of singletons {0.1}, {0.2}, {0.5}, {0.6}, {0.9}, and {1.0}. In the next three levels {0.1} and {0.2}, {0.5} and {0.6}, and {0.9} and {1.0} are merged in some order. Thus, the level four consists of clusters $\{0.1, 0.2\}$, $\{0.5, 0.6\}$, and $\{0.9, 1.0\}$. In the level five either $\{0.1, 0.2\}$ is merged with $\{0.5, 0.6\}$, or $\{0.5, 0.6\}$ is merged with $\{0.9, 1.0\}$. The last level consists of the set P.

It depends on the actual use of the discretized values which one of these two approaches to hierarchical clustering is better. □

In addition to standard divisive and agglomerative hierarchical discretizations, it is possible to find hierarchical discretizations that are optimal with respect to a given permutation $\pi : \{1, \ldots, |P|\} \to \{1, \ldots, |P|\}$ in the following sense: The discretization with $\pi(1)$ discretization points has the minimum error among all discretizations with $\pi(1)$ discretization points. The discretization with $\pi(2)$ discretization points is the one that has the minimum error among all discretizations compatible with the discretization with $\pi(1)$ discretization points. In general, the discretization with $\pi(i)$ discretization points has the minimum error among the discretizations with $\pi(i)$ discretization points that are compatible with the chosen discretizations with $\pi(1), \pi(2), \ldots, \pi(i-1)$ discretization points.

The time complexity of the straightforward dynamic programming implementation of this idea by modifying Algorithm 3.6 is $\mathcal{O}(|P|^4)$. Furthermore, for certain loss functions it is possible to construct hierarchical discretizations that are close to optimal for all values of the number of discretization points simultaneously [Das02].

The discretizations could be applied to association rules instead of frequent patterns. In that case, there are two values to be discretized for each association rule: the frequency and the accuracy of the rule. This can be generalized for patterns with d-dimensional vectors of quality values. The problem is equivalent to clustering, and thus in general, the problem is NP-hard but

many known approximation algorithms for clustering can be applied [BHPI02, dlVKKR03, FG88, KMN+04, KSS04, KVV04].

3.3 Condensation by Discretization

Discretization of frequencies can be used to simplify the collections of frequent patterns. The high-level schema is the following:

1. Discretize the frequencies of the frequent patterns.
2. Find a condensed representation for the pattern collection with the discretized frequencies.

For example, the collection of closed frequent itemsets can be approximated by the closed frequent itemsets with respect to discretized frequencies.

Example 3.11 (condensation by discretization and closed itemsets). Let $\mathcal{I} = \{1, \ldots, \lfloor (1-\sigma)n \rfloor\}$, $\sigma \in (0,1)$ and

$$\begin{aligned} \mathcal{D} &= \{\langle 1, \{1\}\rangle, \ldots, \langle \lfloor (1-\sigma)n \rfloor, \{\lfloor (1-\sigma)n \rfloor\}\rangle\} \\ &\quad \cup \{\langle \lfloor (1-\sigma)n \rfloor + i, \mathcal{I}\rangle : i \in \{1, \ldots, \lceil \sigma n \rceil\}\}. \end{aligned}$$

Then

$$\mathcal{FC}(\sigma, \mathcal{D}) = \{\{1\}, \ldots, \{n\}, \mathcal{I}\}$$

with $fr(\mathcal{I}, \mathcal{D}) = \lceil \sigma |\mathcal{D}| \rceil / |\mathcal{D}|$ and $fr(\{A\}, \mathcal{D}) = \lceil \sigma |\mathcal{D}| + 1 \rceil / |\mathcal{D}|$ for each $A \in \mathcal{I}$.

If we allow error $1/|\mathcal{D}|$ in the frequencies, then we can discretize all frequencies of the non-empty σ-frequent closed itemsets in $\mathcal{D}$ to $\lceil \sigma |\mathcal{D}| \rceil / |\mathcal{D}|$, i.e., $(\gamma \circ fr)(\emptyset, \mathcal{D}) = 1$ and $(\gamma \circ fr)(X, \mathcal{D}) = \lceil \sigma |\mathcal{D}| \rceil / |\mathcal{D}|$ for all other $X \subseteq \mathcal{I}$.

Then the collection $\mathcal{FC}(\sigma, \mathcal{D}, \gamma)$ of σ-frequent closed itemsets with respect to the discretization γ consists only of two itemsets $\emptyset$ and $\mathcal{I}$ with frequencies 1 and $\lceil \sigma |\mathcal{D}| \rceil / |\mathcal{D}|$. □

Note that if the original transaction database is available, then a slightly similar approach to condense the collection of frequent itemsets is to take a random sample of the transactions and compute the closed frequent itemsets in the sample. This reduces the

number of closed itemsets but still results relatively good approximation for the frequencies of the frequent itemsets [Mie04c, Toi96]. A major advantage of computing the closed frequent itemsets in a sample of transactions is that computing the closed frequent itemsets in the sample is potentially much faster than computing the collection of (closed) itemsets in the original data and discretizing the frequencies. Disadvantages of this sampling approach are that the outcome of the closed itemset mining from the sample is also a random variable depending on the sample, and that the quality of the approximation provided by the closed frequent itemsets in the sample is at most as good as the quality of the optimal approximation provided by the optimal discretization of the frequencies. Of course, the sampling and discretization could be used in conjunction, by first taking a relatively large sample of transactions for obtaining the closed frequent itemsets efficiently and then discretizing the frequencies of the closed frequent itemsets in the sample. This should provide the computational efficiency and the approximation quality in between of sampling and discretizing. We focus, however, solely on discretizations.

Example 3.12 (closed itemsets disappearing in the course completion database). Let us consider the collection $\mathcal{FC}(\sigma, \mathcal{D})$ of closed 0.20-frequent itemsets in the course completion database (see Subsection 2.2.1). Recall (Example 2.9) that the number $|\mathcal{FC}(\sigma, \mathcal{D})|$ of the closed 0.20-frequent itemsets in the course completion database is 2136.

If the supports are discretized with the maximum absolute error 2 (that is less than 0.1 percent of the number of transactions in the database), then the number of closed itemsets with respect to the discretized supports is only 567, i.e., less than 24 percent of $|\mathcal{FC}(\sigma, \mathcal{D})|$.

In some parts of the itemset collection $\mathcal{FC}(\sigma, \mathcal{D})$ the reduction in the number of the closed itemsets can be even greater than the average. For example, there are eight subsets of the itemset $X = \{3, 5, 7, 13, 14, 15, 20\}$ than are closed with respect to exact supports but that have the same discretized support as X. These itemsets are shown in Table 3.1.

□

Table 3.1: The itemsets with the same discretized support as $\{3, 5, 7, 13, 14, 15, 20\}$ in the course completion database. The column are as follows: $supp(X, \mathcal{D})$ is the exact support of the itemset X in the completion database $\mathcal{D}$, $\gamma_a^2(supp(X, \mathcal{D}))$ is $supp(X, \mathcal{D})$ discretized with maximum absolute error 2, and $X \in \mathcal{FC}(\sigma, \mathcal{D})$ is the itemset X.

$supp(X, \mathcal{D})$	$\gamma_a^2(supp(X, \mathcal{D}))$	$X \in \mathcal{FC}(\sigma, \mathcal{D})$
488	490	$\{3, 5, 7, 13, 14, 15, 20\}$
489	490	$\{3, 5, 7, 13, 15, 20\}$
489	490	$\{3, 5, 13, 14, 15, 20\}$
490	490	$\{3, 5, 13, 15, 20\}$
490	490	$\{3, 7, 13, 14, 15, 20\}$
491	490	$\{3, 13, 14, 15, 20\}$
492	490	$\{3, 7, 13, 15, 20\}$
492	490	$\{5, 7, 13, 14, 15, 20\}$

The number of discretization points determines the quality of the approximation: On one extreme — a discretization using only one discretization point — the frequent itemsets that are closed with respect to the discretized frequencies correspond to maximal frequent itemsets. When the number of discretization points increases, also the number of closed frequent itemsets increase, the other extreme case being the collection of frequent closed itemsets without any discretization.

If the condensed representation depends on testing whether the frequencies of the patterns are equal (such condensed representations are, for example, the closed and the free patterns), then the number of discretization points can be used as an estimate of the effectiveness of the discretization. In addition to simplifying the collections of frequent patterns, discretization can be used to make the discovery of some patterns more efficient.

We evaluated the condensation abilities of discretizations by discretizing the frequencies of the frequent itemsets in the Internet Usage and IPUMS Census databases (see Subsection 2.2.1), and then computing which of the frequent itemsets are closed also with respect to the discretized frequencies. (In these experiments, we omitted the empty itemset from the itemset collections since its frequency is always 1.)

In the first series of experiments we were interested whether data-dependent discretizations yield to smaller collections of closed itemsets than their data-independent counterparts. We discretized the frequencies using discretization function γ_a^ϵ (Equation 3.4) and the algorithm Prefix-Cover (Algorithm 3.2) with different maximum absolute error thresholds ϵ and removed the itemsets that were not closed with respect to the discretized frequencies.

The results for Internet Usage database with the minimum frequency threshold 0.05 are shown in Table 3.2. The number of the 0.05-frequent itemsets, the number of the closed 0.05-frequent itemsets and the number of the maximal 0.05-frequent itemsets in Internet Usage database are 143391, 141568, and 23441, respectively.

Table 3.2: The number of closed itemsets in the collection of 0.05-frequent itemsets in the Internet Usage database with discretized frequencies for different maximum absolute error guarantees. The columns of the table are the maximum absolute error ϵ allowed, the number of σ-frequent itemsets that are closed with respect to the frequencies discretized using Equation 3.4 and the number of σ-frequent itemsets that are closed with respect to the frequencies discretized using Algorithm 3.2.

ϵ	fixed discretization	empirical discretization
0.0010	123426	123104
0.0050	72211	71765
0.0100	54489	45944
0.0200	34536	31836
0.0400	31587	25845
0.0600	26087	24399
0.0800	24479	23916
0.1000	23960	23705

The results for IPUMS Census database with the minimum frequency threshold 0.2 are shown in Table 3.3. The results were similar to other minimum frequency thresholds. The number of the 0.2-frequent itemsets, the number of the 0.2 frequent closed itemsets and the number of the maximal 0.2-frequent itemsets in IPUMS Census database are 86879, 6689, and 578, respectively.

Clearly, the number of closed σ-frequent itemsets is an upper bound and the number of maximal σ-frequent itemsets is a lower

Table 3.3: The number of closed itemsets in the collection of 0.2-frequent itemsets in the IPUMS Census database with discretized frequencies for different maximum absolute error guarantees. The columns of the table have the same interpretation as the columns of Table 3.2.

ϵ	fixed discretization	empirical discretization
0.0010	3226	3242
0.0050	2362	2375
0.0100	1776	1772
0.0200	1223	1225
0.0400	1014	841
0.0600	932	725
0.0800	711	661
0.1000	627	627

bound for the number of frequent itemsets that are closed with respect to the discretized frequencies. The maximum absolute error is minimized in the case of just one discretization point by choosing its value to be the average of the maximum and the minimum frequencies. The maximum absolute error for the best discretization with only one discretization point for the 0.05-frequent itemsets in the Internet Usage database is 0.4261. This is due to the fact that the highest frequency in the collection of the 0.05-frequent itemsets in the Internet Usage database (excluding the empty itemset) is 0.9022. The maximum absolute error for the best discretization with one discretization point for the 0.2-frequent itemsets in the IPUMS Census database is 0.4000. That is, there is an itemset with frequency equal to 0.2 and an itemset with frequency equal to 1 in the collection of 0.2-frequent itemsets in the IPUMS Census database.

In addition to minimizing the maximum absolute error, we computed the optimal discretizations with respect to the average absolute error using dynamic programming (Algorithms 3.5, 3.6 and 3.7). In particular, we computed the optimal discretizations for each possible number of discretization points. The practical feasibility of the dynamic programming discretization depends crucially on the number $N = |fr(\mathcal{F}(\sigma, \mathcal{D}), \mathcal{D})|$ of different frequencies as its time complexity is $\mathcal{O}(N^3)$. Thus, the tests were conducted using

smaller collections of frequent itemsets than in the case of discretization with respect to the maximum absolute error.

For the average absolute error, a uniform weighting over the frequent itemsets were used. That is, the error of the discretization γ was

$$\frac{1}{|\mathcal{F}(\sigma, \mathcal{D})|} \sum_{X \in \mathcal{F}(\sigma, \mathcal{D})} |fr(X, \mathcal{D}) - \gamma(fr(X, \mathcal{D}))| .$$

The results are shown in Figure 3.1 and in Figure 3.2. The figures can be interpreted as follows. The labels of the curves are the minimum frequency thresholds σ for the collections of σ-frequent itemsets they correspond to. The upper figures show the number of frequent itemsets that are closed with respect to the discretized frequencies against the average absolute error of the discretization. The lower figures show the number of the closed σ-frequent itemsets for discretized frequencies against the number of discretization points.

On the whole, the results are encouraging, especially as the discretizations do not exploit directly the structure of the pattern collection but only the frequencies. Although there are differences between the results on different databases, it is possible to observe that even with a quite small number of closed frequent itemsets and discretization points, the frequencies of the frequent itemsets were approximated adequately.

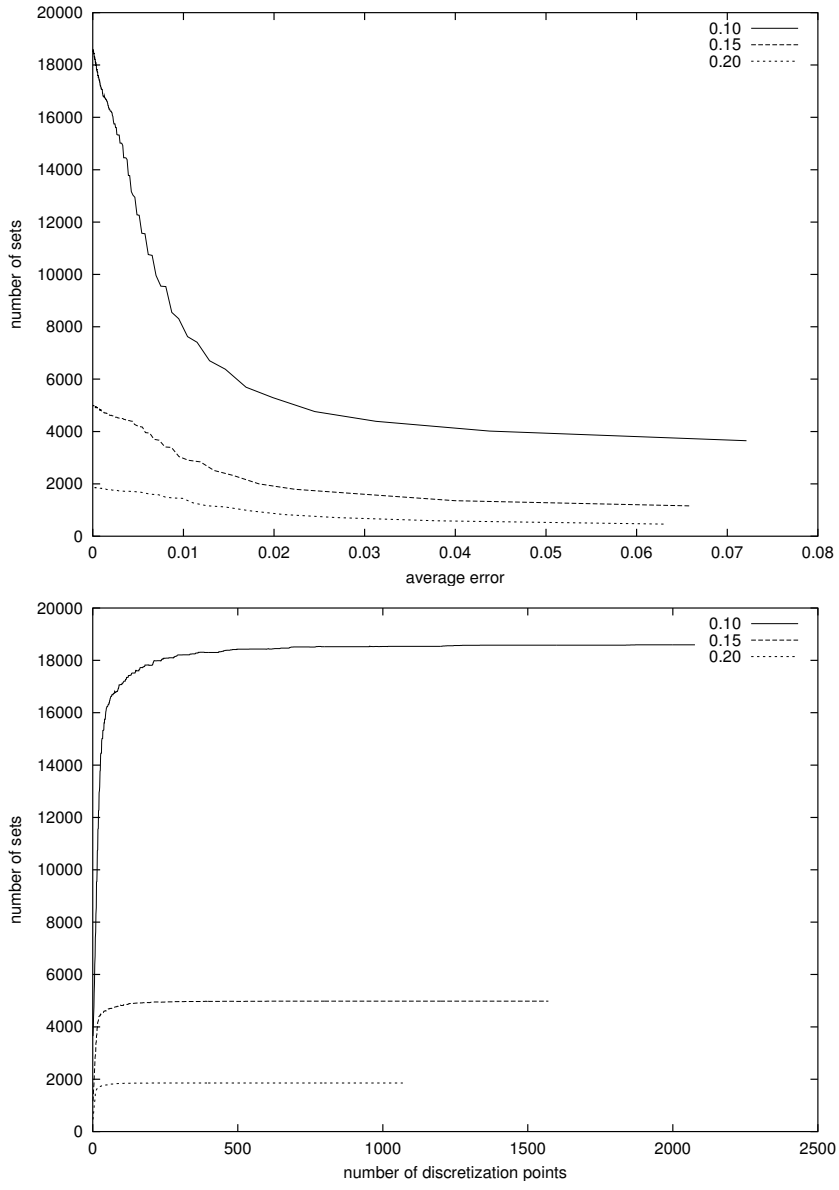

Figure 3.1: The best average absolute error discretizations for Internet Usage data.

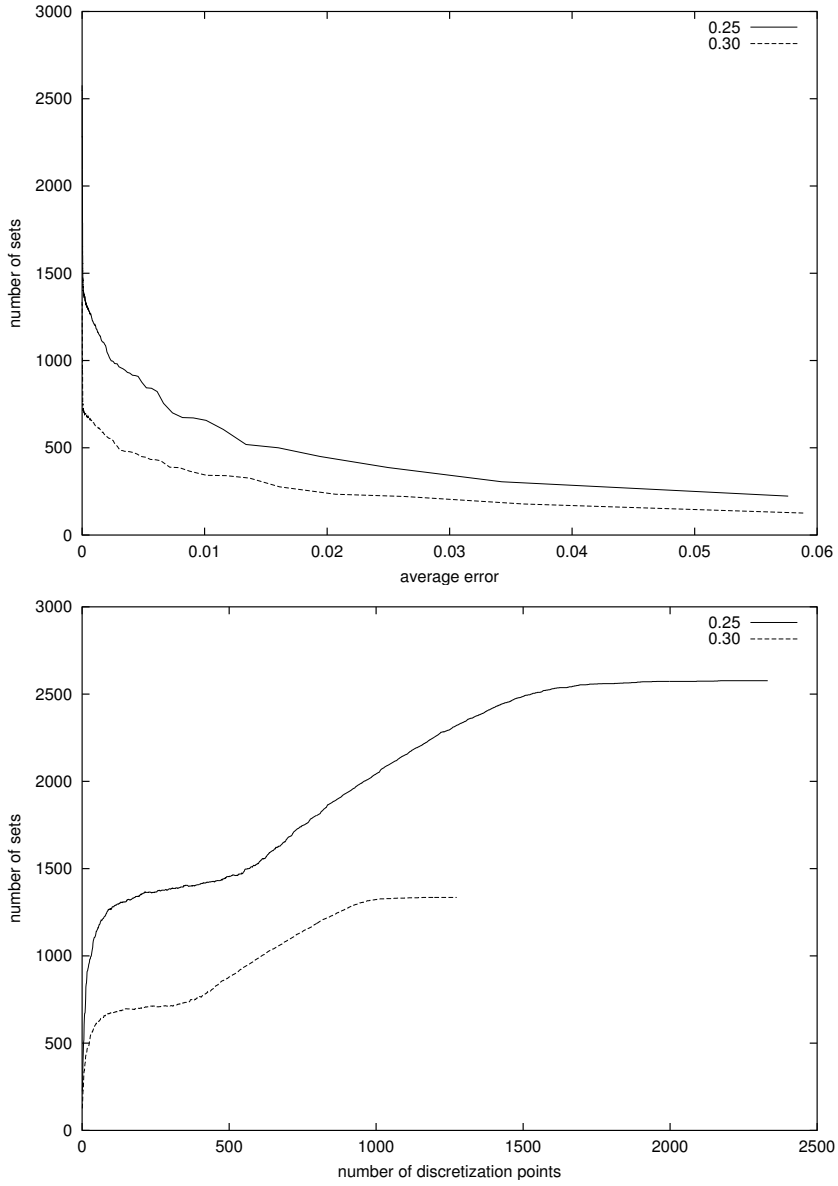

Figure 3.2: The best average absolute error discretizations for IPUMS Census data.

CHAPTER 4

Trade-offs between Size and Accuracy

There are trade-offs between the understandability of the pattern collection and its ability to describe the data at hand:

- If the pattern collection is small, then there is a chance that it could eventually be understandable.
- If the pattern collection is large, then it might describe the data underlying the pattern collection adequately.

Sometimes a very small collection of patterns can be both understandable and accurate description the data. In general, however, characterizing the data accurately requires many patterns assuming that there are many different relevant data sets.

The trade-offs between understandability and accuracy have been studied in pattern discovery mainly by comparing the cardinality of the pattern collection to a quantitative measure of how well the pattern collection describes the relevant aspects of the data.

Typically, one obtains smaller pattern collections by using sufficiently high minimum quality value thresholds. Finding a minimum quality value threshold that captures most of the interesting and only few uninteresting patterns is a challenging or even impossible task in practice.

Example 4.1 (discovering the backbone of a supermarket's profit). Let $\mathcal{D}$ be a transaction database of a supermarket containing purchase events, the set $\mathcal{I}$ of items consisting of the products

sold in that supermarket. Furthermore, let $w(A)$ be the profit of the item $A \in \mathcal{I}$ and let $w(X)$ be the combined profit of the items in the itemset X, i.e., $w(X) = \sum_{A \in X} w(A)$.

Suppose that we are interested in itemsets that fetch large portions of the profit of the supermarket, i.e., the itemsets $X \subseteq \mathcal{I}$ with large *(weighted) area* $w(X)supp(X, \mathcal{D})$ in the transaction database $\mathcal{D}$. Then we have to face several problems, for example the following ones.

First, there is no way to define a minimum frequency threshold that would capture most of the itemsets fetching a large profit without discovering many itemsets with less relevancy to the total profit of the supermarket, although the support of the itemset is the only data-dependent part of this interestingness measure.

Second, it is not clear what would be the right minimum area threshold. For example, why should we choose 10000 instead of 9999 or vice versa? Intuitively this should not matter. However, even a small change in the threshold might change the collection of interesting patterns considerably.

Third, we could find out that we are actually more interested in some other kinds of products, e.g., products with character and a weak brand. However, even realizing that these constraints are important for a itemset being interesting might be very difficult from a large collection of itemsets that contain also sufficiently many such itemsets. Furthermore, weak brand can perhaps be detected based on the discrepancy between the market value and the production costs of the product but determining that a product has character is highly subjective task without, e.g., extensive customer polls.

Thus, finding truly interesting patterns from data is often a challenging, iterative and interactive process. □

To reduce the discrepancy between the size and the accuracy, several condensed representations of pattern collections have been introduced. (See Section 2.4 for more details.) They share, however, the same fundamental difficulties as the other pattern collections: it is difficult to find a small pattern collection that summarizes (the relevant aspects of) the data well. Overcoming these problems with the size and the accuracy seems to be very difficult and they give rise also to a crisp need for interactive exploration of pattern collections and the trade-offs between the size and the accuracy.

If the whole pattern collection is too huge to comprehend, then a natural solution to this problem is to consider only a subcollection of patterns. There are a few properties that a good subcollection of a pattern collection should fulfill. First, the subcollection should be representative for the whole pattern collection. (This requirement is based on the assumption that if the pattern collection describes the data well, then also the representative subcollection should describe the data quite well. The reason why the requirement is not defined directly for the data, instead of the patterns, is that the data might not be available or accessing it might be very expensive. Furthermore, the methods described in this chapter can readily be adapted for measuring the quality of the subcollection using the data instead of the patterns.) Second, the representative subcollection of k patterns should not differ very much from the representative subcollections of $k+1$ and $k-1$ patterns, i.e., the representative subcollections should support interactive mining as it is presumably highly non-trivial to guess the right value of k immediately.

In this chapter we propose, as a solution to this problem, to order the patterns in such a way that the kth pattern in the ordering improves our estimate of the quality values of the patterns as much as possible, given also the $k-1$ previous patterns in the ordering. Note that this ensures that the representative subcollection of k patterns does not differ much from the representative subcollections of $k+1$ and $k-1$ patterns.

In addition to the pattern ordering problem, we study also the problem of choosing the best k-subcollection of the patterns. We show that this problem is NP-hard in general. However, for certain loss functions and estimation methods, the optimal pattern ordering provides a constant factor approximation for the best k-subcollections for all values of k *simultaneously*. That is, each length-k prefix of the ordering is a subcollection that describes the quality values of the patterns almost as well as the best subcollection of cardinality k.

The feasibility of the method depends strongly on the loss function and the estimation method at hand. To exemplify this, we describe concrete instantiations of pattern orderings in two cases. First, we use the pattern ordering to provide a refining representation of frequent patterns. The representation is based on estimating

the frequencies of the patterns by the maximum of the frequencies of the known superpatterns. Any prefix of this pattern ordering can be seen as an approximation of closed frequent patterns. Second, we show how transaction databases can be described as tilings. (Tiling is a collection of tiles. A tile consists of an itemset and a set of transaction identifiers of transactions that contain the itemset. We use the fraction of the items of the database covered by the tiles as the quality of the tiling.)

Finally, we empirically evaluate the suitability of pattern orderings to serve as condensed representations in the case of the frequent itemsets. More specifically, we estimate the frequencies of the frequent itemsets using the maximum frequencies of the known superitemsets and measure the loss by the average of the absolute differences between the correct and the estimated frequencies.

This chapter is based mostly on the article "The Pattern Ordering Problem" [MM03]. (The example of tilings described in Section 4.4 is related also to the article "Tiling Databases" [GGM04].)

4.1 The Pattern Ordering Problem

Most condensed representations of pattern collections consist of a subcollection of the patterns such that the subcollection represents the whole pattern collection well, often exactly. Representing the whole pattern collection well by its subcollection depends on two components.

First, it depends on a function for estimating the quality values of the patterns from the quality values of the patterns in its subcollection, i.e., an estimation method

$$\psi : \bigcup_{\mathcal{S} \subseteq \mathcal{P}} \mathcal{P} \times [0,1]^{\mathcal{S}} \to [0,1] \,.$$

Example 4.2 (frequency estimation). A simple estimate for the frequency $fr(X, \mathcal{D})$ of an itemset $X \subseteq \mathcal{I}$ is

$$\psi_{\mathcal{I},\delta}(X, fr|_{\mathcal{S}}) = \delta^{|\mathcal{I} \setminus X|} \prod_{A \in X, X \in \mathcal{S}} fr(X, \mathcal{D}) \tag{4.1}$$

where $\mathcal{S} \subseteq 2^{\mathcal{I}}$ and δ is the default frequency for the items whose frequencies are not known. This estimation method assumes the independence of the items.

The downside of this estimation method is that it does not make use of the other frequencies than the frequencies of the singleton itemsets. Fortunately it can be generalized to exploit also other frequencies. The idea of the generalization is to find a probability distribution over the itemsets in the transactions that has the *maximum entropy* among the probability distributions compatible with the frequency constraints. This estimation method has been applied successfully in estimating the frequencies of itemsets based on the frequencies of some other itemsets [PMS03]. □

Example 4.3 (frequency estimation). Another simple frequency estimate is

$$\psi_{Max}(X, fr|_{\mathcal{S}}) = \max\{fr(Y, \mathcal{D}) : Y \in \mathcal{S}, Y \supseteq X\}. \tag{4.2}$$

Note that in the case of the closed itemsets (Definition 2.10), the frequencies of the non-closed itemsets are obtained using this rule. □

Second, the estimation is evaluated by a function that measures the error of the estimation, i.e., a loss function

$$\ell : [0,1]^{\mathcal{P}} \times [0,1]^{\mathcal{P}} \to \mathbb{R}.$$

Example 4.4 (L_p norms). One popular class of loss functions are L_p norms

$$\ell_{L_p}(\phi, \psi(\cdot, \phi|_{\mathcal{S}})) = \left(\sum_{x \in \mathcal{P}} |\phi(x) - \psi(x, \phi|_{\mathcal{S}})|^p \right)^{1/p}. \tag{4.3}$$

For example, if $p = 2$ then the L_p norm is the euclidean distance, and if $p = 1$, then it is the sum of absolute errors. The case where $p = \infty$ corresponds to the maximum of the absolute errors. □

To simplify the considerations, we consider estimation methods and loss functions as oracles, i.e., as functions that can be evaluated in constant time regardless of their true computational complexity or even computability. (Although the loss functions are often computable in a reasonable time, there is not a necessity for that restriction in the context of this chapter.) With the aid of the estimation method and the loss function, we can formulate the problem of finding a subcollection that represents the whole pattern collection well as follows:

Problem 4.1 (the best k-subcollection patterns). Given a pattern collection $\mathcal{P}$, an interestingness measure ϕ, a positive integer k, an estimation function ψ, and a loss function ℓ, find the best k-subcollection $\mathcal{S}$ of $\mathcal{P}$, i.e., find a collection $\mathcal{S} \subseteq \mathcal{P}$ such that $|\mathcal{S}| = k$ and for all k-subcollections $\mathcal{S}'$ of $\mathcal{P}$ hold

$$\ell(\phi, \psi(\cdot, \phi|_{\mathcal{S}})) \leq \ell(\phi, \psi(\cdot, \phi|_{\mathcal{S}'})).$$

That is, $\mathcal{S}$ has the smallest error among all k-subcollections of $\mathcal{P}$.

The problem of finding the best k-subcollection of patterns depends on five parameters: the pattern collection $\mathcal{P}$, the interestingness measure ϕ, the estimation method ψ, the loss function ℓ and the number k of patterns allowed in the subcollection.

Example 4.5 (the best k-subcollection itemsets). Combining Example 4.3 and Example 4.4 we get one instance of Problem 4.1:

- The pattern collection $\mathcal{P}$ is a subcollection of the collection $2^{\mathcal{I}}$ of all subsets of the set $\mathcal{I}$ of items. For example, $\mathcal{P}$ could be the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in a given transaction database $\mathcal{D}$.
- The interestingness measure ϕ is the frequency in the transaction database $\mathcal{D}$ and it is defined for all itemsets in the pattern collection.
- The frequencies of the itemsets are estimated by taking the maximum of the frequencies of known superitemsets of the itemset whose frequency is under estimation, i.e., the estimation method ψ is as defined by Equation 4.2.
- The loss in the estimation is measured by the maximum of the absolute differences between the estimated and the correct frequencies of the itemsets in $\mathcal{P}$. That corresponds to Equation 4.3 with $p = \infty$.

These parameters together with the number k (the maximum number of patterns) form an instance of Problem 4.1. □

Problem 4.1 is an optimization problem. It can easily be transformed to a decision problem that asks whether there exists a k-subcollection of $\mathcal{P}$ with the error at most ϵ instead of looking for

the k-subcollection with the smallest error. Unfortunately, even a simple special case of the problem — the decision version of Example 4.5 — is NP-complete as shown by Theorem 4.1.

We show the NP-hardness of Problem 4.1 by reduction from Problem 4.2 which is known to be NP-complete [GJ79]. (For more details in complexity theory and NP-completeness, see [GJ79, Pap95].)

Problem 4.2 (minimum cover [GJ79]). Given a collection C of subsets of a finite set S and a positive integer k, decide whether or not C contains a cover of S of size k, i.e., whether or not there is a subset $C' \subseteq C$ with $|C'| = k$ such that every element of S belongs to at least one member of C'.

Note that we omit the empty itemset from the collection $\mathcal{F}(\sigma, \mathcal{D})$ in the proof of Theorem 4.1 to simplify the reduction, since $fr(\emptyset, \mathcal{D}) = 1$, i.e., it is never necessary to estimate $fr(\emptyset, \mathcal{D})$.

Theorem 4.1. *Given a collection $\mathcal{F}(\sigma, \mathcal{D})$ of σ-frequent itemsets in a transaction database $\mathcal{D}$, a maximum error bound ϵ and a cardinality bound k, it is* NP*-complete to decide whether or not there is a subcollection of $\mathcal{F}(\sigma, \mathcal{D})'$ with the cardinality k such that the maximum absolute error between the correct frequency and the maximum of the frequencies of the superitemsets in the subcollection is at most ϵ. That is, it is* NP*-complete to decide whether or not there is a collection $\mathcal{F}(\sigma, \mathcal{D})' \subseteq \mathcal{F}(\sigma, \mathcal{D})$ such that $|\mathcal{F}(\sigma, \mathcal{D})'| = k$ and*

$$\max_{X \in \mathcal{F}(\sigma, \mathcal{D})} \left\{ fr(X, \mathcal{D}) - \max \left\{ fr(Y, \mathcal{D}) : X \subseteq Y \in \mathcal{F}(\sigma, \mathcal{D})' \right\} \right\} \leq \epsilon.$$

Proof. The problem is in NP since we can check in time polynomial in the sum of the cardinalities of the itemsets in $\mathcal{F}(\sigma, \mathcal{D})$ whether or not the maximum absolute error is at most ϵ for all $X \in \mathcal{F}(\sigma, \mathcal{D})$. That is, we can check in polynomial time for each $X \in \mathcal{F}(\sigma, \mathcal{D})$ and for each $Y \in \mathcal{F}(\sigma, \mathcal{D})'$ such that $X \subseteq Y$ whether

$$|fr(X, \mathcal{D}) - fr(Y, \mathcal{D})| = fr(X, \mathcal{D}) - fr(Y, \mathcal{D}) \leq \epsilon.$$

Let us now describe the reduction from Problem 4.2. It is easy to see that we can assume for each instance $\langle S, C, k \rangle$ of Problem 4.2 that each element of S is contained at least in one set in C, no set in C is contained in another set in C, and the cardinality of each set in C is greater than one. Furthermore, we assume that

the cardinalities of the sets in C are all at most three; the problem remains NP-complete [GJ79].

An instance $\langle C, S, k\rangle$ of the minimum cover problem is reduced to an instance $\langle \mathcal{F}(\sigma, \mathcal{D}), fr, \psi, \ell, k, \epsilon\rangle$ as follows. The set $\mathcal{I}$ of items is equal to the set S. The pattern collection $\mathcal{F}(\sigma, \mathcal{D})$ consists of the sets in C and all their non-empty subsets. Thus, the cardinality of $\mathcal{F}(\sigma, \mathcal{D})$ is $\mathcal{O}(|S|^3)$, since we assumed that the cardinality of the largest set in C is three. The transaction database $\mathcal{D}$ consists of one transaction for each set in C and an appropriate number of transactions that are singleton subsets of S to ensure that $fr(\{A\}, \mathcal{D}) = fr(\{B\}, \mathcal{D}) > fr(X, \mathcal{D})$ for all $A, B \in S$ and $X \in C$. (Thus, the minimum frequency threshold σ is $|\mathcal{D}|^{-1}$.)

If we set $\epsilon = fr(\{A\}, \mathcal{D}) - |\mathcal{D}|^{-1}$ for any element A in S, then there is a set $C' \subset C$ such that $|C'| = k$ if and only if

$$\max_{X \in \mathcal{F}(\sigma,\mathcal{D})} \left\{ fr(X, \mathcal{D}) - \max\left\{ fr(Y, \mathcal{D}) : X \subseteq Y \in \mathcal{F}(\sigma, \mathcal{D})' \right\} \right\} \leq \epsilon$$

holds for the same collection $C' = \mathcal{F}(\sigma, \mathcal{D})' \subseteq \mathcal{F}(\sigma, \mathcal{D})$ with respect to the transaction database $\mathcal{D}$. (Note that without loss of generality, we can assume that $\mathcal{F}(\sigma, \mathcal{D})' \subseteq \mathcal{FM}(\sigma, \mathcal{D}) = C$.) Thus, the problem is NP-hard, too. □

Thus, the decision version of the special case of Problem 4.1 as described by Example 4.5 is NP-complete by Theorem 4.1. Thus, so is Problem 4.1 itself. (For an alternative example of such a special case of Problem 4.1 shown to be NP-complete, see [AGM04].)

Furthermore, the proof of Theorem 4.1 implies also the following inapproximability result for the optimization version of the problem. (For more details in approximability, see [ACK+99].)

Theorem 4.2. *Given a collection $\mathcal{P}$ of itemsets in a transaction database $\mathcal{D}$, their frequencies, the estimation method ψ_{Max} as defined by Equation 4.2 and loss function $\ell_{L_\infty}(\phi, \psi(\cdot, \phi|_{\mathcal{S}}))$ as defined by Equation 4.3, it is* NP*-hard to find a subcollection $\mathcal{S}$ of $\mathcal{P}$ such that*

$$\ell_{L_\infty}(fr|_{\mathcal{P}}, \psi_{Max}(\cdot, fr|_{\mathcal{S}})) \leq \epsilon$$

and the cardinality of $\mathcal{S}$ being within a factor $c \log |\mathcal{I}|$ (for some constant $c > 0$) from the cardinality of the smallest subcollection of $\mathcal{P}$ with error at most ϵ.

Proof. The reduction in the proof of Theorem 4.1 shows that the problem is APX-hard [ACK+99, PY91].

If the collection $\mathcal{F}(\sigma, \mathcal{D})$ is replaced by the collection $\mathcal{P} = C \cup \{\{A\} : A \in S\}$, then we can get rid of the cardinality constraints for the sets in C while still maintaining the itemset collection $\mathcal{P}$ being of polynomial size in the size of the input $\langle C, S, k\rangle$. This gives us stronger inapproximability results. Namely, it is NP-hard to find a set cover $C' \subseteq C$ of the cardinality within a logarithmic factor $c \log |S|$ (for some constant $c > 0$) from the smallest set cover of S in C [ACK+99, RS97].

If we could find a collection $\mathcal{S} \subseteq \mathcal{P}$ of the cardinality k and the error at most ϵ, then that collection could also be a set cover of S of the cardinality k. □

Even if there was a polynomial-time solution for Problem 4.1, it is not clear whether it is the right problem to solve after all. A major disadvantage of the problem is that it does not take into account the requirement that the solution consisting of k patterns should be close to the solutions consisting of $k + 1$ and $k - 1$ patterns. In general, it would be desirable that the solutions of all cardinalities would be somewhat similar.

One approach to ensure that is to order the patterns somehow and consider each length-k prefix of the ordering as the representative k-subcollection of the patterns. The ordering should be such that the prefixes of the ordering are good representative subcollections of (the quality values of) the pattern collection. For example, the patterns could be ordered based on how well the prefixes of the ordering describe the collection.

Problem 4.3 (pattern ordering). Given a pattern collection $\mathcal{P}$, an interestingness measure ϕ, an estimation function ψ and a loss function ℓ, find an ordering $p_1, \ldots, p_{|\mathcal{P}|}$ of the patterns such that

$$\ell(\phi, \psi(\cdot, \phi|_{\{p_1,\ldots,p_i\}})) \leq \ell(\phi, \psi(\cdot, \phi|_{\{p_1,\ldots,p_{i-1},p_j\}})) \qquad (4.4)$$

for all $i \in \{1, \ldots, |\mathcal{P}|\}$ and $j \in \{i, \ldots, |\mathcal{P}|\}$.

The pattern ordering can be seen as a refining approximation of the pattern collection: the first pattern in the ordering describes the pattern collection at least as well as any other pattern in the

collection, the second pattern is the the best choice if the first pattern is already chosen to the representation. In general, the kth pattern in the ordering is the best choice to improve the estimate given the first $k-1$ patterns in the ordering.

Algorithm 4.1 The pattern ordering algorithm.

Input: The collection $\mathcal{P}$ of patterns, the interestingness measure ϕ, the estimation method ψ, and the loss function ℓ.

Output: The optimal pattern ordering as defined by Equation 4.4 and the loss $\varepsilon_i = \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_i}))$ for each i-prefix $\mathcal{P}_i$ of the pattern ordering.

1: **function** ORDER-PATTERNS($\mathcal{P}, \phi, \psi, \ell$)
2: $\quad\mathcal{P}_0 \leftarrow \emptyset$
3: $\quad$**for** $i = 0, \ldots, |\mathcal{P}| - 1$ **do**
4: $\quad\quad p_{i+1} \leftarrow \arg\min_{p \in \mathcal{P} \setminus \mathcal{P}_i} \left\{\ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_i \cup \{p\}}))\right\}$
5: $\quad\quad \mathcal{P}_{i+1} \leftarrow \mathcal{P}_i \cup \{p_{i+1}\}$
6: $\quad\quad \varepsilon_{i+1} \leftarrow \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_{i+1}}))$
7: $\quad$**end for**
8: $\quad$**return** $\left\langle \left\langle p_1, \ldots, p_{|\mathcal{P}|} \right\rangle, \left\langle \varepsilon_1, \ldots, \varepsilon_{|\mathcal{P}|} \right\rangle \right\rangle$
9: **end function**

The pattern ordering and the estimation errors for all prefixes of the ordering can be computed efficiently by Algorithm 4.1. The running time of the algorithm depends crucially on the complexity of evaluating the expression $\ell(\phi(\mathcal{P}), \psi(\mathcal{P}, \phi|_{\mathcal{P}_i \cup \{p\}}))$ for each pattern $p \in \mathcal{P} \setminus \mathcal{P}_i$ and for all $i = 0, \ldots, |\mathcal{P}| - 1$. If $M(\mathcal{P})$ is the maximum time complexity of finding the pattern p_{i+1} that improves the prefix $\mathcal{P}_i$ as much as possible with respect to the estimation method and the loss function, then the time complexity of Algorithm 4.1 is bounded above by $\mathcal{O}(|\mathcal{P}|\, M(\mathcal{P}))$. Note that the algorithm requires at most $\mathcal{O}(|\mathcal{P}|^2)$ loss function evaluations since there are $\mathcal{O}(|\mathcal{P}|)$ possible patterns to be the ith pattern in the ordering.

Example 4.6 (on the efficiency of Algorithm 4.1). Let the estimation method be

$$\psi_{simple}(p, \phi|_{\mathcal{S}}) = \begin{cases} \phi(p) & \text{if } p \in \mathcal{S} \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

i.e., let the quality values be zero unless explicitly given, and let

the loss be the sum of the differences

$$\phi(p) - \psi_{simple}(p, \phi|_{\mathcal{S}}) = \begin{cases} 0 & \text{if } p \in \mathcal{S} \text{ and} \\ \phi(p) & \text{otherwise.} \end{cases}$$

Then finding the pattern that improves to solution the most can be found in time logarithmic in $|\mathcal{P}|$ by using a heap [Knu98]. More specifically, each quality value of a pattern in $\mathcal{P}$ is put into the heap in time $\mathcal{O}(|\mathcal{P}| \log |\mathcal{P}|)$. The best pattern can be found in each iteration by picking the pattern with highest quality value in the heap. Thus, the total running time of the algorithm is then $\mathcal{O}(|\mathcal{P}| \log |\mathcal{P}|)$. (Note that the optimal pattern ordering could be obtained in this case also by sorting the patterns with respect to their quality values.) □

The patterns could be ordered also by starting with the whole pattern collection $\mathcal{P}$ and repeatedly removing from the collection the pattern whose omission increases the error least, rather than starting with an empty collection and adding the pattern that decreases the error most.

If the pattern ordering and the errors for all of its prefixes are computed (as Algorithm 4.1 does), then the user can very efficiently explore the trade-offs between the size and the accuracy: If the number of patterns is overwhelming, then the user can consider shorter prefixes of the pattern ordering. If the accuracy of the estimates is not high enough, then the user can add more patterns to the prefix.

Furthermore, this exploration can be done very efficiently. Finding the prefix of length k can always be implemented to run in constant time by representing the pattern ordering as an array of patterns. The shortest prefix with error at most a given threshold ϵ can be found in time $\mathcal{O}(|\mathcal{P}|)$ by scanning the array of patterns sequentially. Similarly, the prefix of length at most k with the smallest error can be found in time linear in $|\mathcal{P}|$. If the loss function is nonincreasing, i.e., it is such that

$$\psi(\cdot, \phi|_{\mathcal{S}}) \leq \psi(\cdot, \phi|_{\mathcal{S} \setminus \{p\}})$$

for each $p \in \mathcal{S}$ and each $\mathcal{S} \subseteq \mathcal{P}$, then the time consumption of these tasks can be reduced to $\mathcal{O}(\log |\mathcal{P}|)$ by a simple application of binary search.

In addition to efficient exploration of trade-offs between the size and the accuracy, the pattern ordering can shed some light to the relationships between the patterns in the collections. For example, the prefixes of the pattern ordering suggest which patterns are complementary to each other and show which improve the quality value estimation.

4.2 Approximating the Best k-Subcollection of Patterns

On one hand, the problem of finding the best k-subcollection of patterns is NP-hard as shown by Theorem 4.1. Thus, there is not much hope for polynomial-time algorithms for finding the best k-subcollection in general. On the other hand, the optimal pattern ordering can be found by Algorithm 4.1. Furthermore, the greedy procedure (of which Algorithm 4.1 is one example) has been recognized to provide efficiently exact or approximate solutions for a wide variety of other problems [Fei98, GK99, HMS93, KKT03]. Actually, Algorithm 4.1 provides the optimal solution for some special cases. For example, the prefixes of the optimal pattern ordering for Example 4.6 are also the best subcollections. Furthermore, the optimal pattern ordering always determines the best pattern to describe the quality values of the whole collection. Unfortunately it does not provide necessarily the optimal solution for an arbitrary value of k.

Example 4.7 (the suboptimality of the optimal pattern ordering). Let the pattern collection $\mathcal{P}$ be $2^{\{A,B,C\}}$ and the interestingness measure be the support. Let the support of the itemset $\{A, B, C\}$ be 1 and the other supports be 3. Furthermore, let the estimation method be as defined by Equation 4.2 and let the loss function be the euclidean distance, i.e., Equation 4.3 with $p = 2$.

Then the initial loss is 55. The best 3-subcollection consists of itemsets $\{A, B\}$, $\{A, C\}$ and $\{B, C\}$ with the loss 1 whereas Algorithm 4.1 chooses the itemset $\{A, B, C\}$ instead of one of the 2-itemsets, resulting the loss 4. The decreases of losses are 54 and 51, respectively. □

There is still a possibility, however, that the optimal pattern ordering provides reasonable approximations for at least some k-subcollections, loss functions and estimation methods.

In fact, under certain assumptions about the estimation method ψ and the loss function ℓ, it is possible to show that each k-prefix of the pattern ordering is within a constant factor from the corresponding best k-subcollection of patterns in $\mathcal{P}$ for all $k = 1, \ldots, |\mathcal{P}|$ simultaneously.

More specifically, if the estimation method ψ and the loss function ℓ together satisfy certain conditions, then for each k-prefix $\mathcal{P}_k$ of the pattern ordering the decrease of loss

$$\ell(\phi, \psi(\cdot, \phi|_\emptyset)) - \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_k}))$$

is within the factor $1 - 1/e \geq 0.6321$ from the maximum decrease of loss

$$\ell(\phi, \psi(\cdot, \phi|_\emptyset)) - \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_k^*}))$$

for any k-subcollection of $\mathcal{P}$, i.e.,

$$\frac{\ell(\phi, \psi(\cdot, \phi|_\emptyset)) - \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_k}))}{\ell(\phi, \psi(\cdot, \phi|_\emptyset)) - \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_k^*}))} \geq \frac{e-1}{e}$$

for all $k \in \{1, \ldots, |\mathcal{P}|\}$.

To simplify the notation, we use the following shorthands:

$$\begin{aligned}
\varepsilon_i &= \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_i})) \\
\varepsilon_i^* &= \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_i^*})) \\
\delta_i &= \ell(\phi, \psi(\cdot, \phi|_\emptyset)) - \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_i})) \\
\delta_i^* &= \ell(\phi, \psi(\cdot, \phi|_\emptyset)) - \ell(\phi, \psi(\cdot, \phi|_{\mathcal{P}_i^*}))
\end{aligned}$$

The pattern collection $\mathcal{P}$, the interestingness measure ϕ, the estimation method ψ and the loss function ℓ are assumed to be clear from the context.

First we show that if the loss decreases sufficiently from the $i-1$-prefix to the i-prefix for all $i = 1, \ldots, |\mathcal{P}|$, then $\delta_k \geq (1 - 1/e)\, \delta_k^*$ holds for all $k = 1, \ldots, |\mathcal{P}|$.

Lemma 4.1. *If*

$$\delta_i - \delta_{i-1} \geq \frac{1}{k} (\delta_k^* - \delta_{i-1}) \tag{4.5}$$

holds for all i and k with $1 \leq i \leq k \leq |\mathcal{P}|$ then

$$\delta_k \geq \left(1 - \frac{1}{e}\right) \delta_k^*$$

for all $k = 1, \ldots, |\mathcal{P}|$.

Proof. From Equation 4.5 we get

$$\begin{aligned} \delta_i &\geq \frac{1}{k}\delta_k^* + \left(1 - \frac{1}{k}\right) \delta_{i-1} \\ &\geq \frac{1}{k}\delta_k^* \sum_{j=0}^{i} \left(1 - \frac{1}{k}\right)^j \\ &= \frac{1}{k}\delta_k^* \frac{(1-1/k)^i - 1}{(1-1/k) - 1} \\ &= \left(1 - \left(1 - \frac{1}{k}\right)^i\right) \delta_k^* \end{aligned}$$

since by definition $\delta_0 = \varepsilon_0 - \varepsilon_0 = 0$.

Thus,

$$\delta_k \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \delta_k^* \geq \left(1 - \frac{1}{e}\right) \delta_k^*$$

as claimed. □

The approximation with respect to the optimal loss is not so easy. In fact, the optimal pattern ordering does not provide any approximation ratio guarantees in general: there might be a collection $\mathcal{P}_k^*$ of k patterns that provide zero loss estimation of ϕ but still the k-subcollection chosen by Algorithm 4.1 can have non-zero loss. (Note that also in the Example 4.7 the ratio of losses is 4 whereas the ratio between the decreases of losses is 17/18.) Still, we can transform Lemma 4.1 to give bounds for the loss instead of the decrease of the loss.

Lemma 4.2. *If*

$$\varepsilon_{i-1} - \varepsilon_i \geq \frac{1}{k} \left(\varepsilon_{i-1} - \varepsilon_k^*\right)$$

for all i and k with $1 \leq i \leq k \leq |\mathcal{P}|$ then also

$$\varepsilon_k \leq \left(1 - \frac{1}{e}\right) \varepsilon_k^* + \frac{1}{e}\varepsilon_0$$

holds for all $k = 1, \ldots, |\mathcal{P}|$.

Proof. First note that

$$\varepsilon_{i-1} - \varepsilon_i \geq \frac{1}{k}\left(\varepsilon_{i-1} - \varepsilon_k^*\right) \iff \delta_i - \delta_{i-1} \geq \frac{1}{k}\left(\delta_k^* - \delta_{i-1}\right)$$

and second that

$$\varepsilon_k \leq \left(1 - \frac{1}{e}\right)\varepsilon_k^* + \frac{1}{e}\varepsilon_0 \iff \delta_k \geq \left(1 - \frac{1}{e}\right)\delta_k^*.$$

Thus, Lemma 4.1 gives the claimed result. □

The bound given by Lemma 4.2 is considerably weaker than the bound given by Lemma 4.1 due to the additive term of a constant fraction of the initial error, i.e., the error of our initial assumption about the quality values.

Still, the prefixes of the optimal pattern ordering serve as good representative k-subcollections of $\mathcal{P}$ for all values of k simultaneously, in addition to being a refining description of the quality values of the pattern collection.

4.3 Approximating the Quality Values

As a more concrete illustration of the approximation abilities of the pattern orderings, in this section we shall consider the orderings of patterns in downward closed collections $\mathcal{P}$ with anti-monotone interestingness measures when the quality value of a pattern is estimated to be the maximum of the quality values of its known superpatterns (the collection $\mathcal{S}$), i.e.,

$$\psi_{Max}(p, \phi|_{\mathcal{S}}) = \max\left\{\phi(p') : p \preceq p' \in \mathcal{S}\right\}. \tag{4.6}$$

Note that this estimation method was used also in Example 4.3. The next results show that the estimation method ψ_{Max} gives the correct quality values for all patterns in $\mathcal{P}$ exactly when the subcollection used in the estimation contains all closed patterns in the collection $\mathcal{P}$.

Theorem 4.3. *The collection $Cl(\mathcal{P})$ of the closed patterns in $\mathcal{P}$ is the smallest subcollection of $\mathcal{P}$ such that*

$$\phi(p) = \psi_{Max}(p, \phi|_{Cl(\mathcal{P})})$$

for all $p \in \mathcal{P}$.

Proof. By definition, for each pattern $p \in \mathcal{P}$ there is a pattern $p' \in Cl(\mathcal{P})$ such that $p \preceq p'$ and $\phi(p) = \phi(p')$. As we assume that the interestingness measure ϕ is anti-monotone, taking the maximum quality value of the superpatterns $p' \in cl(\mathcal{P})$ of a pattern $p \in \mathcal{P}$ determines to quality value of p correctly. Thus, $\phi|_{Cl(\mathcal{P})}$ is sufficient to determine $\phi|_{\mathcal{P}}$.

To see that all closed patterns in $\mathcal{P}$ are needed, notice that the quality value of a pattern $p \in Cl(\mathcal{P})$ is greater than any of the quality values of its proper superpatterns. Thus, the quality values of the patterns in $Cl(\mathcal{P})$ as $\phi|_{Cl(\mathcal{P})}$ is needed to determine even $\phi|_{Cl(\mathcal{P})}$ using the estimation method ψ_{Max}. □

Thus, the problem of finding the smallest subcollection $\mathcal{S}$ of $\mathcal{P}$ such that $\ell(\phi(p), \psi_{Max}(p, \phi|_{\mathcal{S}})) = 0$ with respect to the estimation method ψ_{Max} and any reasonable loss function ℓ (i.e., a loss function such that only the correct estimation of $\phi|_{\mathcal{P}}$ has zero loss and such that the loss can be evaluated efficiently for any $p \in \mathcal{P}$) can be solved efficiently by Algorithm 2.3.

If some error is allowed, then the complexity of the solution depends also on the loss function. Let us first consider the maximum absolute error

$$\ell_{Max}(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{S}})) = \max_{p \in \mathcal{P}} \{\phi(p) - \psi_{Max}(p, \phi|_{\mathcal{S}})\},$$

i.e., the loss defined by Equation 4.3 with $p = \infty$.

By Theorem 4.1, the problem of finding the best k-subcollection of $\mathcal{P}$ with the loss at most ϵ is NP-hard even when $\mathcal{P} = \mathcal{F}(\sigma, \mathcal{D})$ and $\phi = fr$. The maximum absolute error is not very informative loss function since it does not take into account the number of patterns with error exceeding the maximum error threshold ϵ. Still, it makes a difference whether there is one or one million patterns exceeding the maximum absolute difference threshold.

If the loss function is the number of frequencies that are not estimated with the absolute error at most ϵ, i.e.,

$$\ell_{Max,\epsilon}(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{S}})) = |\{p \in \mathcal{P} : |\phi(p) - \psi_{Max}(p, \phi|_{\mathcal{S}})| > \epsilon\}| \tag{4.7}$$

then the problem can be modeled as a special case of the maximum k-coverage problem (Problem 4.4).

Problem 4.4 (maximum k-coverage [ACK+99]). Given a collection C of subsets of a finite set S and a positive integer k, find a k-subcollection C' of C with the largest coverage of S, i.e., the collection $C' \subseteq C$ of cardinality k that maximizes the cardinality of $S \setminus \left(\bigcup_{X \in C'} X\right)$.

Theorem 4.4. *Let the estimation method be ψ_{Max} (Equation 4.6) and the loss function be $\ell_{Max,\epsilon}$ (Equation 4.7). Then Problem 4.1 is a special case of Problem 4.4.*

Proof. The reduction from an instance $\langle \mathcal{P}, \phi, \psi_{Max}, \ell_{Max,\epsilon}, k \rangle$ of Problem 4.1 to an instance $\langle C, S, k \rangle$ of Problem 4.4 is straightforward. The set S consists of all patterns in $\mathcal{P}$, and C consists of sets $\{p' \in \mathcal{P} : p' \preceq p, \phi(p') - \phi(p) \leq \epsilon\}$ for each $p \in \mathcal{P}$. □

If the sum of errors is used instead of the maximum absolute error, the following approximation bounds can be guaranteed:

Theorem 4.5. *For the length-k prefix $\mathcal{P}_k$ of the optimal solution for the pattern ordering problem of the pattern collection $\mathcal{P}$ and the best k-subcollection $\mathcal{P}_k^*$ of $\mathcal{P}$, we have*

$$\delta_k \geq \left(1 - \frac{1}{e}\right) \delta_k^*$$

for the estimation method ψ_{Max} (Equation 4.6) and for any loss function

$$\ell_f(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{S}})) = \sum_{p \in \mathcal{P}} f(\phi(p) - \psi_{Max}(p, \phi|_{\mathcal{S}})) \tag{4.8}$$

where f is an increasing function.

Proof. Using Lemma 4.1, it is sufficient to show that Equation 4.5 holds.

Let $p_1, \ldots, p_{|\mathcal{P}|}$ be to ordering of the patterns in $\mathcal{P}$ given by Algorithm 4.1 and let $\mathcal{P}_i = \{p_1, \ldots, p_i\}$. The pattern collection $\mathcal{P}$ can be partitioned into k groups $\mathcal{P}_p, p \in \mathcal{P}_k^*$, as follows:

$$\mathcal{P}_{p_i} = \left\{ p \preceq p_i : i = \min \left\{ j \in \{1, \ldots, |\mathcal{P}|\} : \phi(p_j) = \psi_{Max}(p, \phi|_{\mathcal{P}_k^*}) \right\} \right\}.$$

Note that

$$\ell_f(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{S}})) \geq \ell_f(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{S} \cup \mathcal{S}'}))$$

for all $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{P}$. This implies that also

$$\ell_f(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{P}_i})) \geq \ell_f(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{P}_i \cup \mathcal{P}_k^*}))$$

for all $i, k \in \{1, \ldots, |\mathcal{P}|\}$.

For any $i, k \in \{1, \ldots, |\mathcal{P}|\}$, the decrease of loss $\delta_k^* - \delta_{i-1}$ can be written as

$$\left(\sum_{p \in \mathcal{P}} f(\phi(p) - \psi_{Max}(p, \phi|_{\mathcal{P}_{i-1} \cup \mathcal{P}_k^*})) \right) - \delta_{i-1}.$$

and it can be further partitioned into sums

$$\sum_{p' \in \mathcal{P}_p} \left(f(\phi(p') - \psi_{Max}(p', \phi|_{\mathcal{P}_{i-1} \cup \{p\}})) - f(\phi(p') - \psi_{Max}(p', \phi|_{\mathcal{P}_{i-1}})) \right)$$

for each $p \in \mathcal{P}_k^*$. At least one of those sums must be at least $1/k$-fraction of $\delta_k^* - \delta_{i-1}$. Thus, the claim holds. □

Furthermore, the search for the optimal pattern ordering using the estimation method ψ_{Max} (Equation 4.6) can be speeded up by considering, without loss of generality, only the closed patterns:

Theorem 4.6. *For all loss functions ℓ and all subcollections $\mathcal{S}$ of the pattern collection $\mathcal{P}$ we have*

$$\ell(\phi, \psi_{Max}(\cdot, \phi|_{\mathcal{S}})) = \ell(\phi, \psi_{Max}(\cdot, \phi|_{\{cl(p): p \in \mathcal{S}\}}))$$

Proof. Any pattern $p \in \mathcal{P}$ can be replaced by its closure $cl(p, \preceq, \phi)$ since $\phi(p) = \phi(cl(p))$. Furthermore, if $p' \preceq p$ then $p' \preceq cl(p)$ for all $p, p' \in \mathcal{P}$. □

Example 4.8 (ordering 0.20-frequent itemsets in the course completion database). Let us examine how the estimation method ψ_{Max} orders the 0.20-frequent itemsets in the course completion database (see Subsection 2.2.1) when the loss function is the average of the absolute differences.

The averages of the absolute differences in the frequency estimates for all prefixes up to length 2000 of the pattern ordering are shown in Figure 4.1. It can be noticed that the error decreases quite quickly.

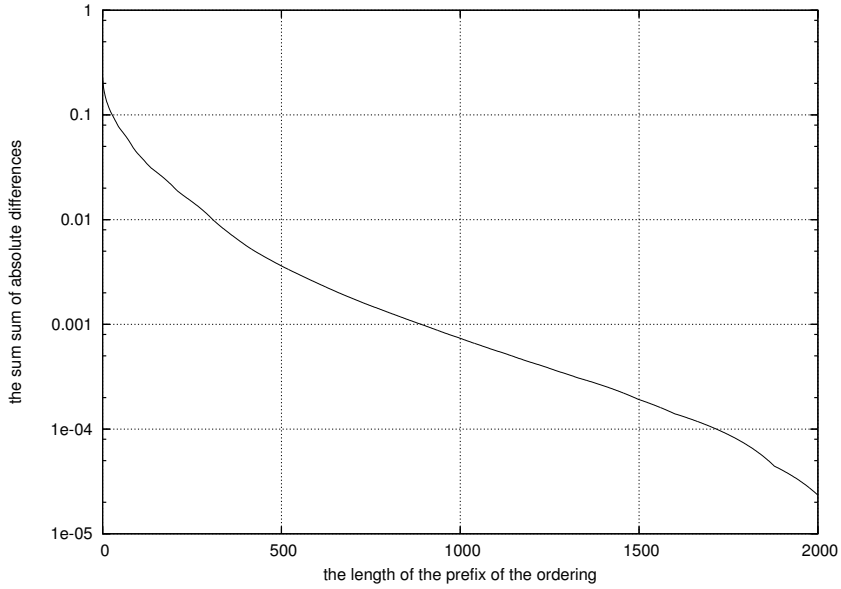

Figure 4.1: The average of the absolute differences between the frequencies estimated by Equation 4.6 and the correct frequencies for each prefix of the pattern ordering.

The decrease of the error does not tell much about the other properties of the pattern ordering. As the estimation method is taking the maximum of the frequencies of the known superitemsets, it is natural to ask whether the first itemsets in the ordering are maximal. Recall that the number of maximal 0.20-frequent itemsets is 253 and the number of closed 0.20-frequent itemsets is 2136. The first 46 itemsets in the ordering are maximal but after that there are also itemsets that are not maximal in the collection 0.20-frequent itemsets in the course completion database. The last maximal itemset appears as the 300th itemset in the ordering. The interesting part of ratios between the non-maximal and the maximal itemsets in the prefixes of the ordering is illustrated in Figure 4.2. (Note that after the 300th itemset in the ordering, the ratio changes by an additive term 1/253 for each itemset.)

One explanation for the relatively large number of maximal itemsets in the beginning of the ordering is that the initial estimate for

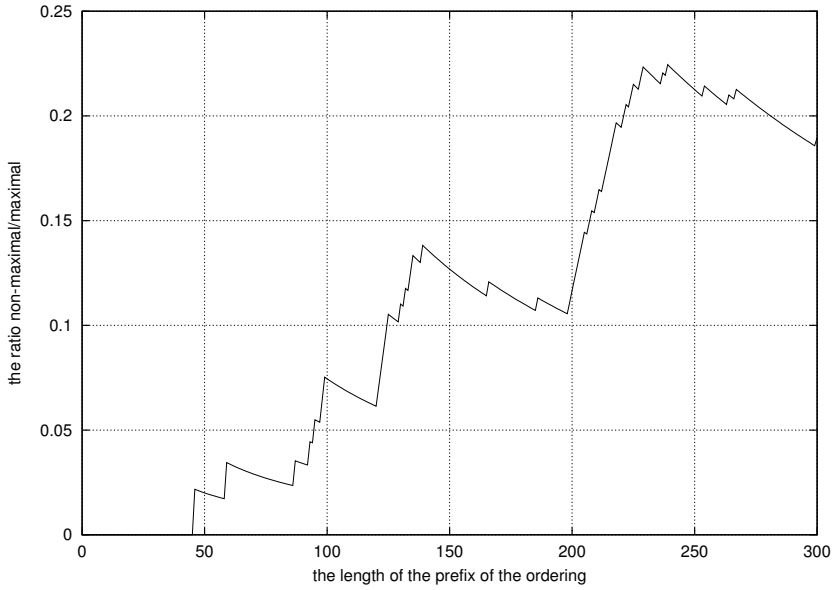

Figure 4.2: The ratio between the non-maximal and the maximal itemsets in the prefixes of the pattern ordering.

all frequencies is 0, whereas the frequency of each maximal itemset is in our case at least 0.20. Furthermore, the first maximal itemsets are quite large (thus containing a large number of at least 0.20-frequent itemsets) and the majority of the frequencies of the 0.20-frequent itemsets are within 0.025 to 0.20. Still, the itemset ordering differs considerably from first listing all maximal 0.20-frequent itemsets and then all other 0.20-frequent itemsets.

For a more refined view to the ordering, the average cardinality of the itemsets in each prefix is shown in Figure 4.3 and the number of itemsets of each cardinality in each prefix is shown in Figure 4.4.

The average cardinality of the itemsets in the prefixes drops quite quickly close to the global average cardinality. That is, after the initial major corrections in the frequencies (i.e., listing some of the largest maximal itemsets) there are both small and large itemsets in the ordering. Furthermore, the itemsets of all cardinalities are listed quite much in the same relative speed. Thus, based on these statistics, the method seems to provide some added value compared

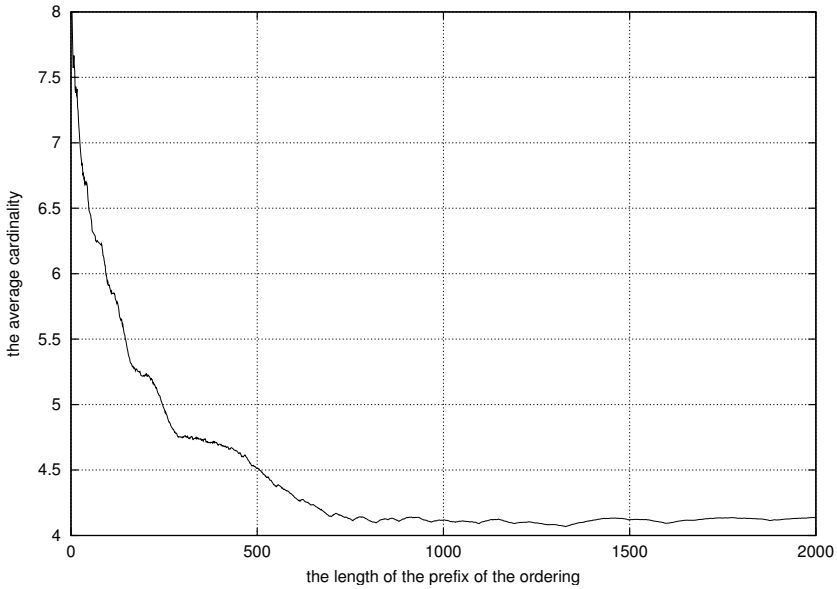

Figure 4.3: The average cardinality of the itemsets.

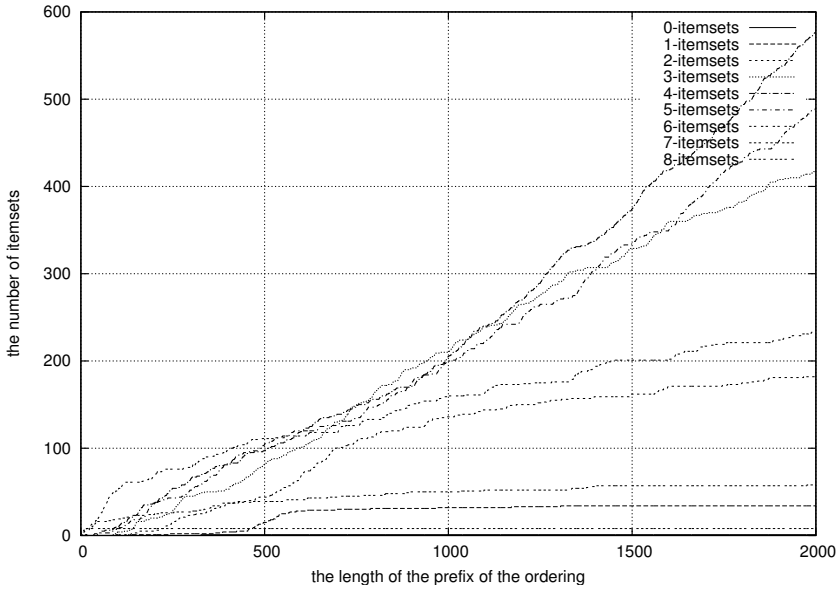

Figure 4.4: The number of itemsets of each cardinality.

to listing the itemsets levelwise from the largest to the smallest cardinality as well as listing the itemsets in the order of increasing frequency. That is, the method seems to list one itemset here and another there, giving a refining view to the itemset collection, as hoped. □

4.4 Tiling Databases

In this section we illustrate the use of pattern orderings as refining description of data, transaction databases in particular.

A transaction database $\mathcal{D}$ can be seen as an $n \times m$ binary matrix $M_{\mathcal{D}}$ such that

$$M_{\mathcal{D}}[i, A] = \begin{cases} 1 & \text{if } A \in X \text{ for some } \langle i, X \rangle \in \mathcal{D} \\ 0 & \text{otherwise.} \end{cases}$$

Viewing transaction databases as binary matrices suggests also pattern classes and interestingness measures different from itemsets and frequencies.

For example, it is not clear why itemsets (i.e., sets of column indices) would be especially suitable for describing binary matrices. Instead of sets of column indices, it could be more natural to describe the matrices by their monochromatic submatrices. Furthermore, as the transaction databases are often sparse, we shall focus on submatrices full of ones, i.e., tiles [GGM04], also known as bi-sets [BRB04], and closely related to formal concepts [GW99]. (Some other approaches to take also the transaction identifiers into account to choose a representative collection of itemsets are described in [TKR$^+$95, WK04].) As a quality measure we shall consider the areas of the tiles.

Definition 4.1 (tiles, tilings and their area). Let $\mathcal{D}$ be a transaction database over $\mathcal{I}$.

A *tile* $\tau(C, X)$ is a set $C \times X$ such that $C \subseteq \mathit{tid}(\mathcal{D})$ and $X \subseteq \mathcal{I}$. The sets C and X can be omitted when they are not of importance. A tile $\tau(C, X)$ is *contained* in $\mathcal{D}$ if for each $\langle i, A \rangle \in \tau(C, X)$ there is a transaction $\langle i, Y \rangle \in \mathcal{D}$ such that $A \in Y$ (and thus $X \subseteq Y$, too).

A tile $\tau(C, X)$ is *maximal* in $\mathcal{D}$ if it is contained in $\mathcal{D}$ and none of its supertiles is contained in $\mathcal{D}$, i.e., a tile $\tau(C, X)$ is maximal in $\mathcal{D}$ if $\tau(C, X) = \tau(\mathit{cover}(X, \mathcal{D}), \mathit{cl}(X, \mathcal{D}))$.

The *area* of a tile $\tau(C, X)$ is

$$area(\tau(C, X)) = |\tau(C, X)| = |C|\,|X|\,.$$

The area of an itemset X in $\mathcal{D}$ is the same as the area of the tile $\tau(cover(X, \mathcal{D}), X)$.

A *tiling* $\mathcal{T}$ is a collection of tiles. A tiling is *contained* in $\mathcal{D}$ if all tiles in the tiling are in $\mathcal{D}$.

The *area* of a tiling $\mathcal{T}$ is

$$area(\mathcal{T}) = \left|\bigcup_{\tau\in\mathcal{T}} \tau\right|.$$

The closure of a tiling $\mathcal{T}$ in $\mathcal{D}$ is

$$cl(\mathcal{T}, \mathcal{D}) = \{\tau(cover(X, \mathcal{D}), cl(X, \mathcal{D})) : \tau(C, X) \in \mathcal{T}\}\,.$$

The motivation to consider the area of tilings is the following. As the transaction databases $\mathcal{D}$ is typically sparse, it might be a good idea to describe $\mathcal{D}$ by indicating where are the ones in the binary matrix by the row and the column indices of submatrices full of ones. Thus, the quality of a tile or a tiling is measured by the number of ones covered by it whereas the goal in frequent itemset mining is to find as high tiles as possible.

Tiles and tilings are most suitable for pattern ordering since the area of a tiling determines a natural loss function for tilings.

More specifically, the task of tiling transaction databases can be formulated as a special case of Problem 4.3. The pattern collection $\mathcal{P}$ is the collection of tiles in $\mathcal{D}$. The interestingness measure ϕ is the area of the tile. The estimation method does not depend on the areas of the known tiles but just the tiles. The known tiles are sufficient to determine the areas of all subtiles of them. (Similarly, in the case of frequent itemsets, a subcollection of frequent itemsets would be sufficient to determine the cardinalities of those frequent itemsets and their subitemsets.) A loss function ℓ can be, e.g., the number of ones not yet covered, i.e.,

$$\ell_{area}(\mathcal{D}, \mathcal{T}) = |\{\langle i, A\rangle : \langle i, X\rangle \in \mathcal{D}, A \in X\}| - area(\mathcal{T}).$$

Thus, the number of transaction databases with $|\mathcal{D}|$ transactions over $\mathcal{I}$ that are compatible with the tiling $\mathcal{T}$ is at most

$$\binom{|\mathcal{D}|\,|\mathcal{I}| - area(\mathcal{T})}{|\{\langle i, A\rangle : \langle i, X\rangle \in \mathcal{D}, A \in X\}| - area(\mathcal{T})}. \tag{4.9}$$

Note that we use the transaction database $\mathcal{D}$ and the tiling $\mathcal{T}$ as the parameters of the loss functions, instead of the area function for tiles in $\mathcal{D}$ and the area function for tiles in $\mathcal{T}$, since they have the same information as the area function for all tiles in $\mathcal{D}$ and the area function for the tiles in the tiling $\mathcal{T}$.

Proposition 4.1. *Each k-prefix of the best ordering of tiles in $\mathcal{D}$ with respect to the loss function ℓ_{area} defines a tiling $\mathcal{T}_k$ that has area within a factor $1 - 1/e$ from the best k-tiling $\mathcal{T}_k^*$ in $\mathcal{D}$.*

Proof. Based on Lemma 4.1, it is sufficient to show that Equation 4.5 holds, i.e., that

$$area(\mathcal{T}_i) - area(\mathcal{T}_{i-1}) \geq \frac{1}{k}\left(area(\mathcal{T}_k^*) - area(\mathcal{T}_{i-1})\right)$$

for all i and k such that $1 \leq i \leq k$.

Let $\mathcal{T}_k^* = \tau_{j_1}, \ldots, \tau_{j_k}$. There must be on $\tau_i^* \in \mathcal{T}_k^*$ such that

$$area(\mathcal{T}_{i-1} \cup \{\tau_i^*\}) - area(\mathcal{T}_{i-1}) \geq \frac{1}{k}\left(area(\mathcal{T}_k^*) - area(\mathcal{T}_{i-1})\right)$$

since

$$area(\mathcal{T}_k^*) \leq \sum_{\tau \in \mathcal{T}_k^*} area(\tau).$$

Thus, there is a tile τ_i in $\mathcal{D}$ but not in $\mathcal{T}_{i-1}$ such that $area(\mathcal{T}_{i-1} \cup \{\tau_i\}) \geq area(\mathcal{T}_{i-1} \cup \{\tau_i^*\})$, i.e., the claim holds. □

Hence, each prefix of the ordering of tiles in $\mathcal{D}$ gives a good approximation for the best tiling of the same cardinality. All we have to do is to find the tiles in $\mathcal{D}$.

The first obstacle for finding the tiles in $\mathcal{D}$ is that the number of tiles can be very large. A slight relief is that we can restrict our focus to maximal tiles in $\mathcal{D}$ instead of all tiles.

Proposition 4.2. *Replacing the tiles $\tau(C, X)$ of a tiling $\mathcal{T}$ in $\mathcal{D}$ by the maximal tiles $\tau(cover(X, \mathcal{D}), cl(X, \mathcal{D}))$ in $\mathcal{D}$ does not decrease the area of the tiling.*

Proof. This is immediate since $\tau(C, X) \subseteq \tau(cover(X, \mathcal{D}), cl(X, \mathcal{D}))$ for each tile $\tau(C, X) \in \mathcal{T}$ contained in $\mathcal{D}$, and thus

$$\bigcup_{\tau(C,X) \in \mathcal{T}} \tau(C, X) \subseteq \bigcup_{\tau(C,X) \in \mathcal{T}} \tau(cover(X, \mathcal{D}), cl(X, \mathcal{D}))$$

which implies that $area(\mathcal{T}) \leq area(cl(\mathcal{T}, \mathcal{D}))$. □

The number of maximal tiles in $\mathcal{D}$ could still be prohibitive. The number of maximal tiles to be considered can be decreased by finding only large maximal tiles, i.e., the maximal tiles with area at least some threshold. Unfortunately, even the problem of finding the largest tile is NP-hard [GGM04, Pee03], but there are methods that can find the large maximal tiles in practice [BRB04, GGM04].

Nevertheless, ordering the large maximal tiles is not the same as ordering all maximal tiles. Although the number of all maximal tiles might be prohibitive, it would possible construct any prefix of the optimal ordering of the maximal tiles in $\mathcal{D}$ if we could find for any prefix $\mathcal{T}_i$ of the ordering the tile τ_{i+1} in $\mathcal{D}$ that maximizes $area(\mathcal{T}_i \cup \{\tau_{i+1}\})$. Clearly, also this problem is NP-hard but in practice such tiles can be found reasonably efficiently [GGM04].

Example 4.9 (Tiling the course completion database). Let us consider the course completion database (see Subsection 2.2.1).

We computed the greedy tilings using Algorithm 4.1 and an algorithm for discovering the tile that improves the current tiling as much as possible. We compared the greedy tiling to the tiling obtained by ordering all frequent and maximal frequent itemsets by their frequencies. The greedy tiling is able to describe the database quite adequately: the 34 first tiles (shown in Table 4.1) in the tiling cover 43.85 percent (28570/65152-fraction) of the ones in the databases. As a comparison, the 34 most frequent itemsets and the 34 most frequent closed itemsets (shown in Table 4.2) cover just 19.13 percent (12462/65152-fraction) of the database. Furthermore, already the 49 first tiles in the greedy tiling cover more than half of the ones in the database.

The relatively weak performance of frequent itemsets can be explained by the fact that frequent itemsets do not care about the other frequent itemsets and also the interaction between closed itemsets is very modest. Furthermore, all of the 34 most frequent itemsets are quite small, the three largest of them consisting of only three items, whereas 22nd tile[1] contains 23 items.

[1]The tile has the largest itemset within the 34 first tiles in the tiling and it consists almost solely of courses offered by the faculty of law. The group of students inducing the tile seem to comprise former computer science students who wanted to become lawyers and a couple of law students that have studied a couple of courses at the department of computer science.

Table 4.1: The 34 first itemset in the greedy tiling of the course completion database.

$supp(X, \mathcal{D})$	$area(X, \mathcal{D})$	X
411	4521	$\{0, 2, 3, 5, 6, 7, 12, 13, 14, 15, 20\}$
1345	2690	$\{0, 1\}$
765	3060	$\{2, 3, 4, 5\}$
367	1835	$\{2, 21, 22, 23, 24\}$
418	1672	$\{7, 9, 17, 31\}$
599	1198	$\{8, 11\}$
513	1026	$\{16, 32\}$
327	1635	$\{7, 14, 18, 20, 27\}$
706	2118	$\{0, 3, 10\}$
357	1428	$\{6, 7, 17, 19\}$
405	1215	$\{0, 24, 29\}$
362	1810	$\{7, 9, 13, 15, 30\}$
197	985	$\{2, 19, 33, 34, 45\}$
296	1184	$\{2, 3, 25, 28\}$
422	844	$\{18, 26\}$
166	830	$\{21, 23, 37, 43, 48\}$
269	538	$\{36, 52\}$
393	786	$\{3, 35\}$
329	1645	$\{6, 7, 13, 15, 41\}$
221	442	$\{40, 60\}$
735	2205	$\{2, 5, 12\}$
20	460	$\{1, 11, 162, 166, 175, 177, 189, 191, 204, 206, 208, 209, 216, 219, 223, 226, 229, 233, 249, 257, 258, 260, 272\}$
294	882	$\{14, 20, 38\}$
410	410	$\{39\}$
852	1704	$\{1, 8\}$
313	939	$\{7, 9, 44\}$
193	386	$\{42, 49\}$
577	1154	$\{21, 23\}$
649	649	$\{25\}$
1069	1069	$\{6\}$
939	1878	$\{0, 4\}$
336	336	$\{46\}$
264	528	$\{19, 50\}$
328	328	$\{47\}$

Table 4.2: The 34 most frequent (closed) itemsets in the course completion database.

$supp(X, \mathcal{D})$	$area(X, \mathcal{D})$	X
2405	0	$\emptyset$
2076	2076	$\{0\}$
1547	1547	$\{1\}$
1498	1498	$\{2\}$
1345	2690	$\{0, 1\}$
1293	2586	$\{0, 2\}$
1209	1209	$\{3\}$
1098	2196	$\{2, 3\}$
1081	1081	$\{4\}$
1071	1071	$\{5\}$
1069	1069	$\{6\}$
1060	1060	$\{7\}$
1057	2114	$\{1, 2\}$
1052	2104	$\{0, 3\}$
1004	2008	$\{3, 5\}$
992	1984	$\{2, 5\}$
983	1966	$\{2, 7\}$
971	1942	$\{2, 6\}$
960	2880	$\{2, 3, 5\}$
958	2874	$\{0, 2, 3\}$
943	1886	$\{0, 5\}$
939	1878	$\{0, 4\}$
931	931	$\{8\}$
924	1848	$\{0, 7\}$
921	1842	$\{0, 6\}$
920	920	$\{9\}$
915	2745	$\{0, 1, 2\}$
911	1822	$\{1, 3\}$
896	1792	$\{6, 7\}$
887	2661	$\{0, 3, 5\}$
880	1760	$\{3, 4\}$
875	2625	$\{0, 2, 5\}$
870	1740	$\{2, 4\}$
862	2586	$\{0, 2, 7\}$

Table 4.3: The 34 maximal itemsets with minimum support threshold 700 in the course completion database.

$supp(X, \mathcal{D})$	$area(X, \mathcal{D})$	X
748	748	$\{15\}$
744	744	$\{16\}$
733	733	$\{17\}$
707	707	$\{18\}$
709	1418	$\{0, 11\}$
700	1400	$\{7, 13\}$
721	1442	$\{7, 14\}$
732	2196	$\{0, 1, 4\}$
730	2190	$\{0, 1, 5\}$
712	2136	$\{0, 1, 7\}$
741	2223	$\{0, 1, 8\}$
749	2247	$\{0, 2, 9\}$
706	2118	$\{0, 3, 10\}$
721	2163	$\{1, 2, 6\}$
755	2265	$\{1, 2, 7\}$
750	2250	$\{2, 3, 6\}$
738	2214	$\{2, 3, 9\}$
724	2172	$\{2, 3, 10\}$
705	2115	$\{2, 5, 6\}$
716	2148	$\{2, 5, 10\}$
726	2178	$\{2, 7, 9\}$
705	2115	$\{3, 5, 6\}$
720	2160	$\{3, 5, 10\}$
704	2112	$\{3, 6, 7\}$
737	2948	$\{0, 1, 2, 3\}$
722	2888	$\{0, 2, 3, 4\}$
849	3396	$\{0, 2, 3, 5\}$
741	2964	$\{0, 2, 3, 7\}$
729	2916	$\{0, 2, 6, 7\}$
706	2824	$\{0, 3, 4, 5\}$
749	2996	$\{1, 2, 3, 5\}$
765	3060	$\{2, 3, 4, 5\}$
757	3028	$\{2, 3, 5, 7\}$
727	2908	$\{2, 3, 5, 12\}$

A slightly better performance can obtained with 34 maximal itemsets (shown in Table 4.3): the 34 maximal itemsets determine a tiling that covers 26.64 percent (17356/65152-fraction) of the database. (The 34 maximal itemsets were obtained by choosing the minimum support threshold to be 700. This is also the origin of choosing the value 34 as the number of illustrated itemsets.) Maximal itemsets depend more on each other since the maximal itemsets form an antichain (see Chapter 5 for more details).

It can be argued that we could afford a slightly larger number of frequent itemsets since they are in some sense simpler than the tiles. We tested this with the collections of the closed 0.20-frequent itemsets (2136 itemsets) and the maximal 0.20-frequent itemsets (253 itemsets) which have been used in previous real examples. They cover 43.80 percent (28535/65152-fraction) and 41.12 percent (26789/65152-fraction), respectively. That is still less than the 34 first tiles in the greedy tiling. □

Thus, sometimes the pattern ordering can be computed incrementally although generating the whole pattern collection would be infeasible. Still, there are many ways how the tilings could be improved.

First, the definitions of tiles and tilings could be adapted also to submatrices full of zeros since a submatrix full of zeros is equivalent to a submatrix full of ones in the binary matrix where all bits are flipped.

Second, the complexity of describing of a particular tile could be taken in to account. Assuming no additional information about the matrix, a $\tau(C, X)$ in $\mathcal{D}$ can be described using

$$|C| \log |\mathcal{D}| + |X| \log |\mathcal{I}|$$

bits. However, taking into account the encoding costs of the tiles, it is not sufficient to consider only maximal tiles.

Example 4.10 (Maximal tiles with costs are not optimal). Let the the transaction database $\mathcal{D}$ consist of two transactions: $\langle 1, \{A\}\rangle$ and $\langle 2, \{A, B\}\rangle$. Then the maximal tiles describing $\mathcal{D}$ are $\{\langle 1, A\rangle, \langle 2, A\rangle\}$ and $\{\langle 2, A\rangle, \langle 2, B\rangle\}$ whereas tiles $\{\langle 1, A\rangle\}$ and $\{\langle 2, A\rangle, \langle 2, B\rangle\}$ would be sufficient and slightly cheaper, too. □

Third, the bound for the number of databases given by Equation 4.9 does not take into account the fact that the tiles in the

tiling are maximal. Let $tid(\tau)$ and $\mathcal{I}_\tau$ be the transaction identifiers and the items in a tile τ, respectively. The maximality of the tile τ restricts the collection of compatible databases $\mathcal{D}$ as follows. The tile τ must be in the compatible database $\mathcal{D}$. For each transaction identifier $i \in tid(\mathcal{D}) \setminus tid(\tau)$ there must be an item $A \in \mathcal{I}_\tau$ such that $\langle i, X\rangle \in \mathcal{D}$ does not contain A. For each item $\mathcal{I} \setminus \mathcal{I}_\tau$ there must be a transaction identifier $i \in tid(\tau)$ such that $\langle i, X\rangle \in \mathcal{D}$ does contain A. The collection of the transaction databases compatible with a tiling $\mathcal{T}$ is the intersection of the collections of the transaction databases compatible with each tile in $\mathcal{T}$.

4.5 Condensation by Pattern Ordering

We evaluated the ability of the pattern ordering approach to condense the collections of the σ-frequent itemsets using the estimation method

$$\psi_{Max}(X, fr|_{\mathcal{F}(\sigma,\mathcal{D})'}) = \max\left\{fr(Y,\mathcal{D}) : X \supseteq Y \in \mathcal{F}(\sigma,\mathcal{D})'\right\}.$$

where $\mathcal{F}(\sigma,\mathcal{D})'$ is the subcollection of σ-frequent itemsets for which the frequencies are known (see also Example 4.3). The loss function used in the experiments was the average absolute error with uniform distribution over the itemset collection, i.e.,

$$\ell(fr|_{\mathcal{F}\sigma,\mathcal{D}}, \psi_{Max}(\cdot, fr|_{\mathcal{F}(\sigma,\mathcal{D})'})) = \\ \frac{1}{|\mathcal{F}(\sigma,\mathcal{D})|}\sum_{X\in\mathcal{F}(\sigma,\mathcal{D})}\left(fr(X,\mathcal{D}) - \max_{X\subseteq Y\in\mathcal{F}(\sigma,\mathcal{D})'} fr(Y,\mathcal{D})\right).$$

The pattern orderings were found by the algorithm OrderPatterns (Algorithm 4.1). Then we computed the pattern orderings for σ-frequent itemsets in the transaction databases Internet Usage and IPUMS Census for several different minimum frequency thresholds $\sigma \in [0, 1]$. The results are shown in Figure 4.5 and in Table 4.4 for Internet Usage and in Figure 4.6 and in Table 4.5 for IPUMS Census.

In the figures the axes are the following. The x-axis corresponds to the length of the prefix of the pattern ordering and the y-axis is corresponds to the average absolute error of the frequency estimation from the corresponding prefix. The labels of the curves express

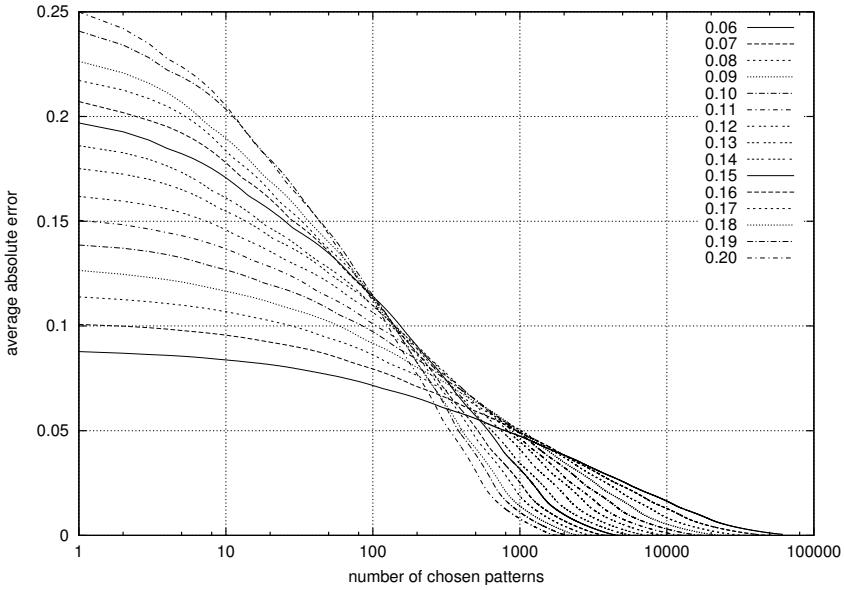

Figure 4.5: Pattern orderings for Internet Usage data.

σ	$\lvert\mathcal{F}(\sigma,\mathcal{D})\rvert$	0	0.001	0.005	0.01	0.02	0.04	0.08
0.20	1856	1856	1574	1190	900	619	418	188
0.19	2228	2228	1870	1396	1052	728	486	212
0.18	2667	2667	2217	1625	1206	820	522	211
0.17	3246	3246	2672	1925	1421	970	597	231
0.16	4013	4013	3254	2295	1671	1132	655	242
0.15	4983	4983	3994	2764	1995	1377	775	270
0.14	6291	6290	4955	3339	2362	1602	860	261
0.13	8000	7998	6208	4093	2881	1972	1034	281
0.12	10476	10472	7970	5118	3562	2414	1189	289
0.11	13813	13802	10267	6352	4305	2804	1284	264
0.10	18615	18594	13468	8068	5409	3395	1423	245
0.09	25729	25686	18035	10399	6920	4094	1587	203
0.08	36812	36714	24870	13681	9032	5008	1708	153
0.07	54793	54550	35441	18477	12147	6276	1803	95
0.06	85492	84873	52295	25595	16376	7568	1747	29

Table 4.4: Pattern orderings for Internet Usage data.

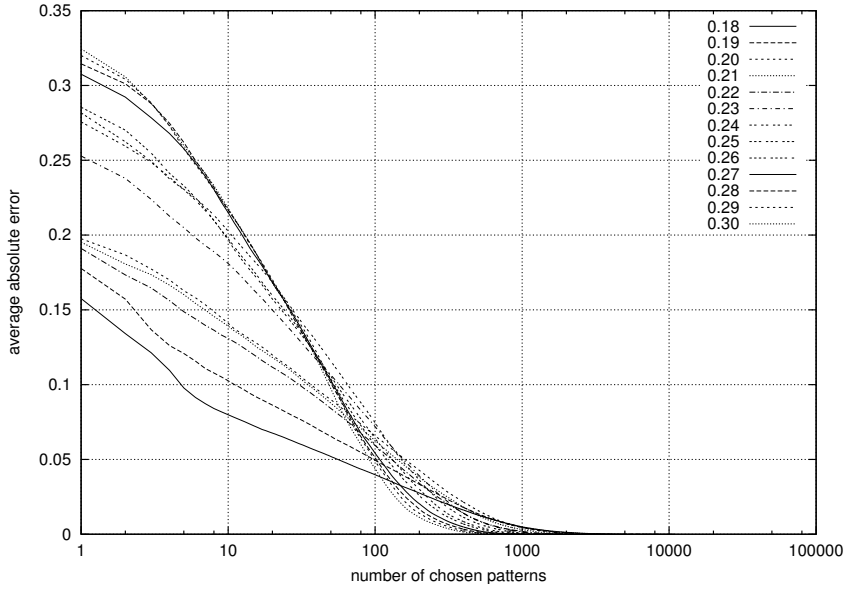

Figure 4.6: Pattern orderings for IPUMS Census data.

σ	$\lvert\mathcal{F}(\sigma,\mathcal{D})\rvert$	0	0.001	0.005	0.01	0.02	0.04	0.08
0.30	8205	1335	444	285	212	153	107	61
0.29	9641	1505	496	317	236	167	116	65
0.28	11443	1696	551	351	260	184	120	66
0.27	13843	1948	624	395	292	203	128	68
0.26	17503	2293	725	456	338	233	147	71
0.25	20023	2577	810	502	369	256	161	77
0.24	23903	3006	944	583	427	293	185	92
0.23	31791	3590	1093	661	477	328	196	85
0.22	53203	4271	1194	678	481	316	171	57
0.21	64731	5246	1454	813	573	372	189	62
0.20	86879	6689	1771	949	661	424	218	67
0.19	151909	8524	1974	953	628	363	151	27
0.18	250441	10899	2212	992	625	312	99	10

Table 4.5: Pattern orderings for IPUMS Census data.

the minimum frequency thresholds of the corresponding frequent itemset collections.

The tables can be interpreted as follows. The columns σ and $|\mathcal{F}(\sigma, \mathcal{D})|$ correspond to the minimum frequency threshold σ and the number of σ-frequent itemsets. The rest of the columns 0, 0.001, 0.005, 0.01, 0.02, 0.04 and 0.08 correspond to the number of itemsets in the shortest prefix with the loss at most 0, 0.001, 0.005, 0.01, 0.02, 0.04 and 0.08, respectively. (Note that the column 0 corresponds to the number of closed frequent itemsets by Theorem 4.3.)

The results show that already relatively short prefixes of the pattern orderings provide frequency estimates with high accuracy. The inversions of the orders of the error curves in Figure 4.5 and in Figure 4.6 are due to the used combination of the estimation method and the loss functions: On one hand the average absolute error is lower for frequent itemset collections with lower minimum frequency threshold for the frequency estimation without any data since initially all frequency estimates of the frequent itemsets are zero. On the other hand the frequencies can be estimated correctly from the closed frequent itemsets and the number of closed frequent itemsets is smaller for higher minimum frequency thresholds.

CHAPTER 5

Exploiting Partial Orders of Pattern Collections

Large collections of interesting patterns can be very difficult to understand and it can be too expensive even to manipulate all patterns. Because of these difficulties, recently a large portion of pattern discovery research has been focused on inventing condensed representations for pattern collections. (See Section 2.4 for more details.)

Most of the condensed representations are based on relatively local properties of the pattern collections: the patterns in the condensed representations are typically chosen solely based on small neighborhoods in the original pattern collection regardless of which of the patterns are deemed to be redundant and which are chosen to the condensed representation.

Two notable exceptions to this are the condensed frequent pattern bases [PDZH02] and non-derivable itemsets [CG02]. Still, even these condensed representations have certain drawbacks and limitations.

The construction of condensed frequent patterns bases is based on a greedy strategy: The patterns are pruned from minimal to maximal or vice versa. A pattern is deemed to be redundant (i.e., not being in the pattern base) if its frequency is close enough to the frequency of some already found irredundant pattern that is its super- or subpattern, depending on the processing direction of the pattern collection. Alas, also the condensed frequent pattern bases can be interpreted to be dependent only on the local neighborhoods

of the patterns, although the neighborhoods are determined by the frequencies rather than only by the structure of the underlying pattern class.

The non-derivable itemsets (Definition 2.12) take into account more global properties of the pattern collection. Namely, the irredundancy (i.e., non-derivability) of an itemset with respect to derivability depends on the frequencies of its all subitemsets. However, the irredundancy of the itemset in the case of non-derivable itemsets is determined using inclusion-exclusion truncations. Although the non-derivable itemsets can be superficially understood as the itemsets whose frequencies cannot be derived exactly from the frequencies of their subitemsets, it is not so easy to see immediately which aspects of the itemset collection and the frequencies of the itemsets one particular non-derivable itemset represents, i.e., to see the essence of the upper and the lower bounds of the itemsets for the underlying transaction database.

The pattern collections have also other structure than the quality values. (In fact, not all pattern collections have quality values at all. For example, the interesting patterns could be determined by an oracle that is not willing to say anything else than whether or not a pattern is fascinating.) In particular, virtually all pattern collections adhere some non-trivial partial order (Definition 2.5).

The goal of this chapter is to make pattern collections more understandable and concise by exploiting the partial orders of the collections. We use the partial orders to partition a given pattern collection to subcollections of (in)comparable patterns, i.e., to (anti)chains. In addition to clustering the patterns in the collection into smaller groups using the partial order of the pattern class, we show that the chaining of patterns can also condense the pattern collection: for many pattern classes each chain representing possibly several patterns can be represented as only a slightly more complex pattern than each of the patterns in the chain.

In this chapter, we propose the idea of (anti)chaining patterns, illustrate its usefulness and potential pitfalls, and discuss the computational aspects of the chaining massive pattern collections. Furthermore, we explain how, for some pattern classes, each chain can represented as one slightly more complex pattern than the patterns in the underlying pattern collection.

This chapter is based on the article “Chaining Patterns” [Mie03a].

5.1 Exploiting the Structure

The collections of interesting patterns (and the underlying pattern classes, too) have usually some structure.

Example 5.1 (structuring the collection of itemsets by frequencies). The collection $2^{\mathcal{I}}$ of all itemsets can be structured based on their frequencies: every subset of a frequent itemset is frequent and every superset of an infrequent itemset is infrequent. Thus, for each minimum frequency threshold $\sigma \in [0,1]$, a given transaction database $\mathcal{D}$ determines a partition

$$\left\langle \mathcal{F}(\sigma,\mathcal{D}), 2^{\mathcal{I}} \setminus \mathcal{F}(\sigma,\mathcal{D}) \right\rangle$$

of the itemset collection $2^{\mathcal{I}}$. □

The downward closed collections of frequent itemsets are examples of data-dependent structures of pattern collections. The pattern collections have also some data-independent structure. Maybe the most typical data-independent structure in a pattern collection is a partial order over the patterns.

Example 5.2 (set inclusion as a partial order over itemsets). Let the pattern class be again $2^{\mathcal{I}}$. A natural partial order for itemsets is the partial order determined by the set inclusion relation:

$$X \prec Y \iff X \subset Y$$

for all $X, Y \subseteq \mathcal{I}$. □

A partial order where no two patterns are comparable, i.e., an empty partial order, is called a *trivial* partial order. For example, any partial order restricted to maximal or minimal patterns is trivial. A trivial partial order is the least informative partial order in the sense that it does not relate the patterns to each other at all.

Besides of merely detecting the structure in the pattern collection, the found structure can sometimes be further exploited. For example, the frequent itemsets can be stored into an *itemset tree* by defining a total order for over $\mathcal{I}$. In an itemset tree, each itemset corresponds to a path from root to some node the labels of the edges being the items of the itemsets in ascending order. (Itemset trees are known also by several other names, see [AAP01, AMS+96, HPYM04, Zak00].)

Example 5.3 (an itemset tree). Let the itemset collection consist of itemsets $\emptyset$, $\{A\}$, $\{A, B, C\}$, $\{A, B, D\}$, $\{A, C\}$, $\{B\}$, $\{B, C, D\}$, and $\{B, D\}$. The itemset tree representing this itemset collection is shown as Figure 5.1.

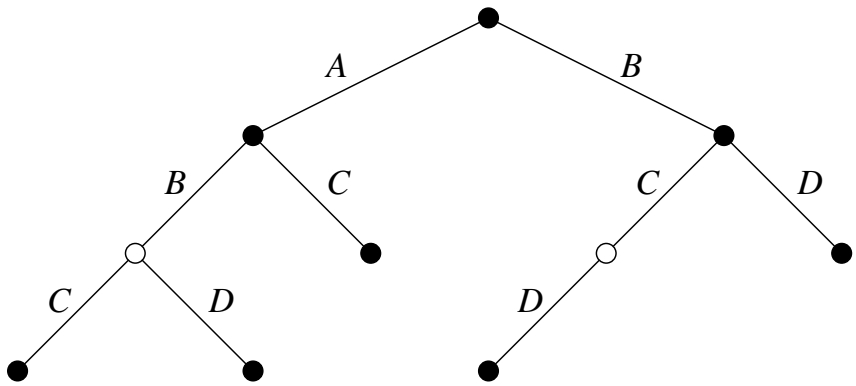

Figure 5.1: An itemset tree representing the itemset collection of Example 5.3. Each itemset can be seen in the tree as a path from the root to a solid node.

□

Representing an itemset collection as an itemset tree can save space and support efficient quality value queries. The quality value of an itemset X can be retrieved (or decided that it is not in the itemset tree) in time $\mathcal{O}(|X|)$. (Time and space complexities similar to itemset tries can be obtained also by refining the itemset trees to automata [Mie05a].) Unfortunately, the structure of itemset trees is strongly dependent on the ordering of the items: there are not always natural orderings for the items and an arbitrary ordering can induce artificial structure to the itemset tree that hides the essence of the pattern collection.

The exploitation of the partial order structure of a pattern collection somehow might still be beneficial although, for example, it is not clear whether the itemset tree makes a partial order of an itemset collection more understandable or even more obscure from the human point of view. A simple approach to reduce the obscurity of the itemset trees is to construct for each itemset collection a random forest of itemset trees where each itemset tree represents

the itemset collection using some random ordering of the items. (These random forests should not be confused with the random forests of Leo Breiman [Bre01].) Unfortunately, the ordering of the items is still present in each of the itemset trees. Fortunately, there are structures in partial orders that do not depend on anything else than the partial order. Two important examples of such structures are *chains* and *antichains*.

Definition 5.1 (chains and antichains). A subset $\mathcal{C}$ of a partially ordered set $\mathcal{P}$ is called a *chain* if and only if all elements in $\mathcal{C}$ are comparable with each other, i.e., $p \preceq p'$ or $p' \preceq p$ holds for all $p, p' \in \mathcal{C}$.

The *rank* of a pattern p in chain $\mathcal{C}$, denoted by $rank(p, \mathcal{C})$, is the number of elements that has to be removed from the chain before p is the minimal pattern in the chain.

A subset $\mathcal{A}$ of a partially ordered set $\mathcal{P}$ is called an *antichain* if and only if all elements in $\mathcal{C}$ are incomparable with each other, i.e., $p \prec p'$ holds for no $p, p' \in \mathcal{C}$.

Example 5.4 (chains and antichains). The itemset collection

$$\{\{A, B, C, D, E, F\}, \{A, C, E\}, \{C, E\}, \{C\}\}$$

is a chain with respect to the set inclusion relation since all itemsets in the collections are comparable with each other. Similarly, the itemset collection $\{\{A, B\}\{A, C\}\}$ is an antichain. The itemset collection $\{\{A, B\}, \{A\}, \{B\}\}$ is not a chain nor an antichain: the collection is not a chain since $\{A\}$ and $\{B\}$ are not comparable, and it is not an antichain since $\{A, B\}$ is comparable with $\{A\}$ and $\{B\}$. □

A chain or an antichain in a pattern collection can be understood more easily than the whole pattern collection, since each pattern either does or does not have relationship with each other pattern in the chain or the antichain, respectively. Thus, a natural approach to make a pattern collection more digestible using a partial order structure is to partition the pattern collection into chains or antichains.

Definition 5.2 (chain and antichain partitions). A *chain partition* (an *antichain partition*) of a partially ordered set $\mathcal{P}$ is a partition of the set $\mathcal{P}$ into disjoint chains $\mathcal{C}_1, \ldots, \mathcal{C}_m$ (disjoint antichains $\mathcal{A}_1, \ldots, \mathcal{A}_m$).

A chain partition (an antichain partition) of $\mathcal{P}$ is *minimum* if and only if there is no chain partition (no antichain partition) of $\mathcal{P}$ consisting of a smaller number of chains (antichains).

A chain partition (an antichain partition) of $\mathcal{P}$ is *minimal* if and only if there are no two chains $\mathcal{C}_i$ and $\mathcal{C}_j$ (antichains $\mathcal{A}_i$ and $\mathcal{A}_j$) in the chain partition (the antichain partition) such that their union is a chain (an antichain).

A chain or an antichain partition can be interpreted as a structural clustering of the patterns. Each chain represents a collection of comparable patterns and each antichain a collection of incomparable ones, i.e., a chain consists of structurally similar patterns whereas an antichain can be seen as a representative collection of patterns.

The minimum chain partition is not necessarily unique. The lack of uniqueness is not only a problem because of the exploratory nature of data mining. Different partitions highlight different aspects of the pattern collection which can clearly be beneficial when one is trying to understand the pattern collection (and the underlying data set).

The maximum number of chains (antichains) in a chain partition (an antichain partition) of a pattern collection $\mathcal{P}$ is $|\mathcal{P}|$ since each pattern $p \in \mathcal{P}$ as a singleton set $\{p\}$ is a chain and an antichain simultaneously. The minimum number of chains in a chain partition is at least the cardinality of the largest antichain in $\mathcal{P}$ since no two distinct patterns in the largest antichain can be in the same chain. This inequality can be shown to be actually an equality and the result is known as *Dilworth's Theorem*:

Theorem 5.1 (Dilworth's Theorem [Juk01]). *A partially ordered set $\mathcal{P}$ can be partitioned into m chains if and only if the largest antichain in $\mathcal{P}$ is of cardinality at most m.*

Example 5.5 (bounding the number of chains from below). The maximal patterns in a pattern collections form an antichain. Thus, the number of maximal patterns is a lower bound for the number of chains in the minimum chain partition. □

Similarly to bounding the minimum chain partitions by maximum antichains, it is possible to bound the minimum number of

antichains needed to cover all patterns in $\mathcal{P}$ by the cardinality of the maximum chain in $\mathcal{P}$:

Theorem 5.2 ([Sch03]). *The number of antichains in a minimum antichain partition of a partially ordered set $\mathcal{P}$ is equal to the cardinality of a maximum chain in $\mathcal{P}$.*

5.2 Extracting Chains and Antichains

The problem of finding the minimum chain partition for a partially ordered pattern collection $\mathcal{P}$ can be formulated as follows:

Problem 5.1 (minimum chain partition). Given a pattern collection $\mathcal{P}$ and a partial order $\prec$ over $\mathcal{P}$, find a partition of $\mathcal{P}$ into the minimum number of chains $\mathcal{C}_1, \ldots, \mathcal{C}_m$.

The minimum chain partition can be found efficiently by finding a maximum matching in a bipartite graph [LP86]. The maximum bipartite matching problem is the following [Sch03]:

Problem 5.2 (maximum bipartite matching). Given a bipartite graph $\langle U, V, E \rangle$ where U and V are sets of vertices and E is a set of edges between U and V, i.e., a set of pairs in $U \times V$, find a maximum bipartite matching $M \subseteq E$, i.e., find a largest subset M of E such that

$$deg(u, M) = |\{e \in M : (u, v) = e \text{ for some } v \in V\}| \leq 1$$

for all $u \in U$ and

$$deg(v, M) = |\{e \in M : (u, v) = e \text{ for some } u \in U\}| \leq 1$$

for all $v \in V$.

The matching is computed in a bipartite graph consisting two copies $\mathcal{P}$ and $\mathcal{P}'$ of the pattern collection $\mathcal{P}$ and the partial order $\prec'$ as edges between $\mathcal{P}$ and $\mathcal{P}'$ corresponding to the partial order $\prec$. Thus, the bipartite graph representation of the pattern collection $\mathcal{P}$ is a triplet $\langle \mathcal{P}, \mathcal{P}', \prec' \rangle$.

Proposition 5.1. *The a matching M in a bipartite graph $\langle \mathcal{P}, \mathcal{P}', \prec' \rangle$ determines a chain partition. The number of unmatched vertices in $\mathcal{P}$ (or equivalently in $\mathcal{P}'$) is equal to the number of chains.*

Proof. Let us consider the partially ordered set $\mathcal{P}$ as a graph $\langle \mathcal{P}, \prec \rangle$ A matching $M \subseteq \prec'$ ensures that the in-degree and out-degree of each $p \in \mathcal{P}$ is at most one. Thus, the set M partitions the graph $\langle \mathcal{P}, \prec \rangle$ to paths. By transitivity of partial orders, each path is a chain.

The number of unmatched patterns in $\mathcal{P}$ correspond to the minimal patterns of the chains. Each unmatched pattern $p \in \mathcal{P}$ is a minimal pattern in some chain and if a patterns p is minimal pattern in some chain then it is unmatched. As each chain contains exactly one minimal pattern, the number of unmatched patterns in $\mathcal{P}$ is equal to the number of chains in the chain partition corresponding to the matching M. □

Due to Proposition 5.1, the number of chains is minimized when the cardinality of the matching is maximized. The chain partition can be extracted from the matching M in time linear in the cardinality of $\mathcal{P}$. The partition of a partially ordered pattern collection into the minimum number of chains can be computed as described by Algorithm 5.1.

A maximum matching M in a bipartite graph $\langle U, V, E \rangle$ can be found in time $\mathcal{O}(\sqrt{\min\{|U|, |V|\}}\,|E|)$ [Gal86]. Thus, if the partial order $\prec$ is known explicitly, then the minimum chain partition ca be found in time $\mathcal{O}(\sqrt{|\mathcal{P}|}\,|\prec|)$ which can be bounded above by $\mathcal{O}(|\mathcal{P}|^{5/2})$ since the cardinality of $\prec$ is at most $|\mathcal{P}|\,(|\mathcal{P}|-1)/2$.

The idea of partitioning the graph $\langle \mathcal{P}, \prec \rangle$ into the minimum number of disjoint paths can be generalized to partitioning it into disjoint degree-constrained subgraphs with maximum number of matched edges by finding a maximum bipartite b-matching instead of a maximum (ordinary) bipartite matching matching. The maximum bipartite b-matching differs from the maximum bipartite matching (Problem 5.2) only by the degree constraints. Namely, each vertex in $v \in U \cup V$ has a positive integer $b(v)$ constraining the maximum degree of the vertex: the degree $deg(v, M)$ of v in the matching M can be at most $b(v)$. Thus the maximum bipartite matching is a special case of the maximum bipartite b-matching with $b(v) = 1$ for all $v \in U \cup V$.

If there is a weight function $w : \prec \to \mathbb{R}$, then the graph $\langle \mathcal{P}, \prec \rangle$ can be partitioned also into disjoint paths with maximum total weight. That is, the pattern can be partitioned into disjoint chains in such a

Algorithm 5.1 A minimum chain partition.

Input: A pattern collection $\mathcal{P}$ and a partial order $\prec$ over $\mathcal{P}$.

Output: Partition of $\mathcal{P}$ into the minimum number m of chains $\mathcal{C}_1, \ldots, \mathcal{C}_m$.

```
 1: function PARTITION-INTO-CHAINS(P, ≺)
 2:     M ← MAXIMUM-MATCHING(P, P, ≺)
 3:     m ← 0
 4:     for all p ∈ P do
 5:         prev [p] ← p
 6:         next [p] ← p
 7:     end for
 8:     for all ⟨p, p'⟩ ∈ M do
 9:         next [p] ← p'
10:         prev [p'] ← p
11:     end for
12:     for all p ∈ P, p = prev [p] do
13:         m ← m + 1
14:         C_m ← {p}
15:         while p ≠ next [p] do
16:             p ← next [p]
17:             C_m ← C_m ∪ {p}
18:         end while
19:     end for
20:     return ⟨C_1, ..., C_m⟩
21: end function
```

way that the sum of the weights of consecutive patterns in the chains is maximized. This can be done by finding a maximum weight bipartite matching that differs from the maximum bipartite matching (Problem 5.2) by the objective function: instead of maximizing the cardinality $|M|$ of the matching M, the weight $\sum_{e \in M} w(e)$ of the edges in the matching M is maximized.

However, there are two traits in partitioning the partially ordered pattern collections into chains: pattern collections are often enormously large and the partial order over the collection might be known only implicitly.

Due to the problem of pattern collections being massive, finding the maximum bipartite matching in time $\mathcal{O}(\sqrt{|\mathcal{P}|}\,|\prec|)$ can be

too slow. This problem can be overcome by finding a maximal matching instead of a maximum matching. A maximal matching in $\langle \mathcal{P}, \mathcal{P}', \prec' \rangle$ can be found in time $\mathcal{O}(|\mathcal{P}| + |\prec|)$ as shown by Algorithm 5.2.

Algorithm 5.2 A greedy algorithm for finding a maximal matching in a bipartite graph $\langle \mathcal{P}, \mathcal{P}', \prec' \rangle$.

Input: A bipartite graph $\langle \mathcal{P}, \mathcal{P}', \prec' \rangle$.
Output: A maximal matching M in the graph.

```
1: function MAXIMAL-MATCHING(P, P', ≺')
2:     for all p ∈ P do
3:         prev[p] ← p
4:         next[p] ← p
5:     end for
6:     for all ⟨p, p'⟩ ∈≺' do
7:         if p = next[p] and p' = prev[p'] then
8:             next[p] ← p'
9:             next[p'] ← p
10:        end if
11:    end for
12:    M ← ∅
13:    for all p ∈ P, p ≠ next[p] do
14:        M ← M ∪ {⟨p, next[p]⟩}
15:        next[p] ← p
16:    end for
17:    return M
18: end function
```

It is easy to see that the cardinality of a maximal matching is at least half of the cardinality of the maximum matching in the same graph. Unfortunately, this does not imply any non-trivial approximation quality guarantees for the corresponding chain partitions.

Example 5.6 (minimum and minimal chain partitions by maximum and maximal matchings). Let us consider the pattern collection $\{1, 2, \ldots, 2n\}$ with partial order

$$\prec = \{\langle i, j \rangle : i < j\}.$$

The maximum matching

$$\{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \ldots, \langle 2n-1, 2n \rangle\}$$

determines only one chain

$$\mathcal{C} = \{1, 2, \ldots, 2n\}$$

whereas the worst maximal matching

$$\{\langle 1, 2n\rangle, \langle 2, 2n-1\rangle, \ldots, \langle n, n+1\rangle\}$$

determines n chains

$$\mathcal{C}_1 = \{1, 2n\}, \mathcal{C}_2 = \{2, 2n-1\}, \ldots, \mathcal{C}_n = \{n, n+1\}.$$

Thus, in the worst case the chain partition found by maximal matching is $|\mathcal{P}|/2$ times worse than the optimal chain partition found by maximum matching. □

The quality of the maximal matching, i.e., the quality of the minimal chain partition can be improved by finding a total order conforming the partial order. If the partial order is known explicitly, then a total order conforming it can be found by topological sorting in time $\mathcal{O}(|\prec| + |\mathcal{P}|)$. Sometimes there is a total order that can be computed without even knowing the partial order explicitly. For example, frequent itemsets can be sorted with respect to their cardinalities. This kind of ordering can reduce the number of chains in the chain partition found by maximal matchings considerably. The amount of the improvement depends on how well the total order is able to capture the essence of the partial order (whatever it might be).

Example 5.7 (improving maximal matching using a total order). Let us consider the pattern collection and the partial order of Example 5.6. If the patterns in the collection $\{1, 2, \ldots, 2n\}$ are ordered in ascending or in descending order, then the maximal matching agrees with the maximum matching, i.e., the minimal chain partition agrees with the minimum chain partition. □

If the partial order is given implicitly, as a function that can be evaluated for any pair of patterns $p, p' \in \mathcal{P}$, then the explicit construction of the partial order relation $\prec$ might itself be a major bottleneck of the chaining of the patterns. The brute force construction of the partial order $\prec$, i.e., testing of all pairs of patterns in $\mathcal{P}$ requires $\mathcal{O}(|\mathcal{P}|^2)$ comparisons. In the worst case this upper bound is tight.

Example 5.8 (the number of comparison in the worst case). Let the pattern collection $\mathcal{P}$ be an antichain with respect to a partial order $\prec$. Then all patterns in $\mathcal{P}$ must be compared with all other patterns in $\mathcal{P}$ in order to construct $\prec$ explicitly, i.e., to ensure that all patterns in $\mathcal{P}$ are incomparable with each other and thus that $\mathcal{P}$ indeed is an antichain. □

Fortunately, the partial order relations have two useful properties that can be exploited in the construction of $\prec$, namely transitivity and antisymmetry holding for any partial order relation $\preceq$. Due to transitivity, $p \preceq p'$ and $p' \preceq p''$ together imply $p \preceq p''$, and antisymmetry guarantees that the graph $\langle \mathcal{P}, \prec \rangle$ is acyclic. The partial order $\preceq$ can be computed also as a side product of the construction of a chain partition as shown for minimal chain partitions by Algorithm 5.3.

Although Algorithm 5.3 needs time $\mathcal{O}(|\mathcal{P}|^2)$ in the worst case, $|\mathcal{C}_i|$ comparisons are always sufficient to decide whether a pattern $p \in \mathcal{P}$ can be added to $\mathcal{C}_i$. Furthermore, the number of comparison can be reduced to $1 + \lfloor \log_2 |\mathcal{C}_i| \rfloor$ comparisons if $\mathcal{C}_i$ is represented as, e.g., a search tree instead of a linked list. The number of comparisons can be reduced also by reusing already evaluated comparisons and transitivity. Furthermore, there are several other strategies to construct the partial order relation $\prec$. The usefulness of different strategies depends on the cost of evaluating the comparisons and the actual partial order. Thus, it seems that choosing the best strategy for constructing the partial order has to be estimated experimentally in general.

Another partition of a pattern collection based on a partial order is an antichain partition. The problem of finding a minimum antichain partition of a partially ordered pattern collection $\mathcal{P}$ can be formulated as follows:

Problem 5.3 (minimum antichain partition). Given a pattern collection $\mathcal{P}$ and a partial order $\prec$ over $\mathcal{P}$, find a partition of $\mathcal{P}$ into the minimum number of antichains $\mathcal{A}_1, \ldots, \mathcal{A}_m$.

Solving Problem 5.3 is relatively easy based on Theorem 5.2. Algorithm 5.4 solves the problem in the case of arbitrary pattern collections.

In many cases the minimum antichain partition can be found even more easily. For example, the minimum antichain partition

Algorithm 5.3 An algorithm to find a minimal chain partition.

Input: A pattern collection $\mathcal{P}$ and a partial order $\prec$ over $\mathcal{P}$.

Output: A minimal chain partition $\mathcal{C}_1, \ldots, \mathcal{C}_m$ of the pattern collection $\mathcal{P}$.

```
1: function MINIMAL-PARTITION-INTO-CHAINS(P, ≺)
2:     m ← 0
3:     for all p ∈ P do
4:         prev [p] ← p
5:         next [p] ← p
6:         i ← 1
7:         while i ≤ m and p = prev [p] = next [p] do
8:             if p ≺ min C_i then
9:                 next [p] ← min C_i
10:                prev [min C_i] ← p
11:            else if max C_i ≺ p then
12:                prev [p] ← max C_i
13:                next [max C_i] ← p
14:            end if
15:            p' ← prev [max {p'' ∈ C_i : p ≺ p''}]
16:            if p = prev [p] = next [p] and p' ≺ p then
17:                next [p] ← next [p']
18:                prev [p] ← p'
19:                next [p'] ← p
20:                prev [next [p]] ← p
21:            end if
22:            if p ≠ prev [p] or p ≠ next [p] then
23:                C_i ← C_i ∪ {p}
24:            end if
25:            i ← i + 1
26:        end while
27:        if p = prev [p] = next [p] then
28:            m ← m + 1
29:            C_m ← {p}
30:        end if
31:    end for
32:    return ⟨C_1, ..., C_m⟩
33: end function
```

Algorithm 5.4 A minimum antichain partition.

Input: A pattern collection $\mathcal{P}$ and a partial order $\prec$ over $\mathcal{P}$.
Output: Partition of $\mathcal{P}$ into the minimum number m of antichains $\mathcal{A}_1, \ldots, \mathcal{A}_m$.

```
1: function PARTITION-INTO-ANTICHAINS(P, ≺)
2:     P' ← P
3:     m ← 0
4:     while P' ≠ ∅ do
5:         m ← m + 1
6:         A_m ← {p ∈ P' : p ⪯ p' ∈ P' ⇒ p = p'}
7:         P ← P' \ A_m
8:     end while
9:     return ⟨A_1, ..., A_m⟩
10: end function
```

of σ-frequent itemsets can be computed in time linear in the sum of cardinalities of the σ-frequent itemsets: The length m of the longest chain in $\mathcal{F}(\sigma, \mathcal{D})$ is one greater than the cardinality of the largest itemset in the collection. Thus, the collection $\mathcal{F}(\sigma, \mathcal{D})$ can be partitioned into m antichains $\mathcal{A}_1, \ldots, \mathcal{A}_m$ containing all σ-frequent itemsets of cardinalities $0, \ldots, m-1$, respectively. Clearly, this partition can be constructed in time linear in $\sum_{X \in \mathcal{F}(\sigma,\mathcal{D})} |X|$ by maintaining m lists of patterns.

5.3 Condensation by Chaining Patterns

A chain partition of a pattern collection can be more than a mere structural clustering if the collection has more structure than a partial order. One example of such a pattern collection is a transaction database without its transaction identifiers.

Example 5.9 (itemset chains). Let us consider the transaction database $\mathcal{D}$ shown as Table 5.1.

Note that the transaction database $\mathcal{D}$ could be represented also as a collection of weighted itemsets, i.e., as a collection

$$\{\{1\}, \{2\}, \{1,3\}, \{2,4\}, \{1,2,3\}, \{1,2,4\}\} = \{1, 2, 13, 23, 123, 124\}$$

Table 5.1: The transaction identifiers and the itemsets of the transactions in $\mathcal{D}$.

tid	X	*tid*	X
1	$\{1\}$	9	$\{1,3\}$
2	$\{2\}$	10	$\{2,4\}$
3	$\{2\}$	11	$\{2,4\}$
4	$\{2\}$	12	$\{2,4\}$
5	$\{2\}$	13	$\{2,4\}$
6	$\{2\}$	14	$\{1,2,3\}$
7	$\{1,3\}$	15	$\{1,2,3\}$
8	$\{1,3\}$	16	$\{1,2,4\}$

of itemsets together with a weight function

$$w = \{1 \mapsto 1, 2 \mapsto 5, 13 \mapsto 3, 24 \mapsto 4, 123 \mapsto 2, 124 \mapsto 1\}.$$

The collection of itemsets representing the transaction database $\mathcal{D}$ can be partitioned into two chains $\mathcal{C}_1 = \{1, 13, 123\}$ and $\mathcal{C}_2 = \{2, 24, 124\}$. □

Each chain $\mathcal{C}$ of itemsets can be written as a one itemset X by adding to each item in the itemsets of $\mathcal{C}$ the information about the minimum rank of the itemset in $\mathcal{C}$ containing that item. That is, a chain $\mathcal{C} = \{X_1, \ldots, X_n\}$ such that $X_1 \subset \ldots \subset X_n = A_1 \ldots A_m$ can be written as

$$\mathcal{C} = \left\{A_1^{rank(A_1,\mathcal{C})}, \ldots, A_m^{rank(A_m,\mathcal{C})}\right\} = A_1^{rank(A_1,\mathcal{C})} \ldots A_m^{rank(A_m,\mathcal{C})}$$

where

$$rank(A, \mathcal{C}) = \min_{A \in X \in \mathcal{C}} rank(X, \mathcal{C})$$

for any item A. (Note that the superscript corresponding to the ranks serve also as separators of the items, i.e., no other separators such as commas are needed.) Furthermore, if there are several items $A_i, i \in I = \left\{i_1, \ldots, i_{|I|}\right\}$, with the same rank $rank(A_i, \mathcal{C}) = k$, then we can write $\{A_i : i \in I\}^k = \left\{A_{i_1}, \ldots, A_{i_{|I|}}\right\}^k$ instead of $A_{i_1}^k \ldots A_{i_{|I|}}^k$. The ranks can even be omitted in that case if the items are ordered by their ranks.

The quality values of the itemsets in the chain $\mathcal{C}$ can be expressed as a vector of length $|\mathcal{C}|$ where ith position of the vector is quality

value of the itemset with rank $i-1$ in the chain. Also, if the interestingness measure is known to be strictly increasing or strictly decreasing with respect to the partial order, then the ranks can be replaced by the quality values of the itemsets.

Example 5.10 (representing the itemset chains). The itemset chains $\mathcal{C}_1$ and $\mathcal{C}_2$ of Example 5.9 can be written as

$$\mathcal{C}_1 = 1^0 2^2 3^1$$

and

$$\mathcal{C}_2 = 1^2 2^0 4^1$$

where the superscripts are the ranks. The whole transaction database (neglecting the actual transaction identifiers) is determined if also the weight vectors

$$w(\mathcal{C}_1) = \langle 1, 3, 2 \rangle$$

and

$$w(\mathcal{C}_2) = \langle 5, 4, 1 \rangle$$

associated to the chains $\mathcal{C}_1$ and $\mathcal{C}_2$ are given. □

From a chain represented as an itemset augmented with the ranks of items in the chain, it is possible to construct the original chain. Namely, a rank-k itemset of the chain

$$\mathcal{C} = A_1^{rank(A_1,\mathcal{C})} \dots A_m^{rank(A_m,\mathcal{C})}$$

is

$$\{A_i : rank(A_i, \mathcal{C}) \leq k, 1 \leq i \leq m\}.$$

This approach to represent pattern chains can be adapted to a wide variety of different pattern classes such sequences and graphs. Besides of making the pattern collection more compactly representable and hopefully more understandable, this approach can also compress the pattern collections.

Example 5.11 (condensation by itemset chains). Let an itemset collection consists of itemsets

$$\{0\}, \{0,1\}, \dots, \{0,\dots,n-1\}, \{1,\dots,n\}, \{2,\dots,n\}, \dots, \{n\}.$$

The collection can be partitioned to two chains

$$\mathcal{C}_1 = \{\{0\}, \{0,1\}, \ldots, \{0,\ldots,n-1\}\}$$

and

$$\mathcal{C}_2 = \{\{1,\ldots,n\}, \{2,\ldots,n\}, \{n\}\}.$$

The size of each chain is $\Theta(n^2)$ items if they are represented explicitly but only $\mathcal{O}(n \log n)$ items if represented as itemsets augmented with the item ranks, i.e., as

$$\mathcal{C}_1 = \left\{0^0, 1^1, \ldots, (n-1)^{n-1}\right\} = 0^0 1^1 \ldots (n-1)^{n-1}$$

and

$$\mathcal{C}_2 = \left\{1^{n-1}, 2^{n-2}, \ldots, n^0\right\} = 1^{n-1} 2^{n-2} \ldots n^0.$$

□

Example 5.12 (chain and antichain partitions in the course completion database). To illustrate chain and antichain partitions, let us again consider the course completion database (see Subsection 2.2.1) and especially the 0.20-frequent closed itemsets in it (see Example 2.9).

By Dilworth's Theorem (Theorem 5.1), each antichain in the collection gives a lower bound for the minimum number of chains in any chain partition of the collection. As itemsets of each cardinality form an antichain, we know (see Table 2.5) that there are at least 638 chains in any chain partition of the collection of 0.20-frequent closed itemsets in the course completion database.

The minimum number of chains in the collection is slightly higher, namely 735. (That can be computed by summing the values of the second column of Table 5.2 representing the numbers of chains of different lengths.) The mode and median lengths of the chains are both three.

Ten longest chains are shown in Table 5.3. (The eight chains of length five are chosen arbitrarily from the 36 chains of length five.) The columns of the table are follows. The column $|\mathcal{C}|$ corresponds to the lengths of the chains, the column $\mathcal{C}$ to the chains, and the column $supp(\mathcal{C}, \mathcal{D})$ to the vectors representing the supports of the itemsets in the chain.

The chains in Table 5.3 show one major problem of chaining by (unweighted) bipartite matching: the quality values can differ quite

Table 5.2: The number of chains of all non-zero lengths in the minimum chain partition of the 0.20-frequent closed itemsets in the course completion database.

the length of chain	the number of chains
1	63
2	189
3	277
4	168
5	36
6	2

Table 5.3: Ten longest chains in the minimum chain partition of the 0.20-frequent closed itemsets in the course completion database.

$\lvert\mathcal{C}\rvert$	$\mathcal{C}$	$supp(\mathcal{C}, \mathcal{D})$
6	$12^0 2^1 15^2 13^3 0^4 \{3,5\}^5$	$\langle 763, 739, 616, 558, 523, 520\rangle$
6	$7^0 10^1 3^2 5^3 13^4 2^5$	$\langle 1060, 570, 565, 559, 501, 496\rangle$
5	$\{6,13\}^0 15^1 2^2 12^3 \{3,5\}^4$	$\langle 625, 579, 528, 507, 504\rangle$
5	$\{0,12\}^0 6^1 13^2 7^3 \{2,3,5\}^4$	$\langle 690, 569, 500, 495, 481\rangle$
5	$15^0 0^1 1^2 5^3 3^4$	$\langle 748, 678, 539, 497, 491\rangle$
5	$\{2,5\}^0 15^1 0^2 7^3 \{6,12\}^4$	$\langle 992, 666, 608, 575, 499\rangle$
5	$0^0 10^1 2^2 1^3 3^4$	$\langle 2076, 788, 692, 526, 510\rangle$
5	$\{2,3\}^0 6^1 12^2 7^3 13^4$	$\langle 1098, 750, 601, 574, 515\rangle$
5	$1^0 9^1 5^2 0^3 2^4$	$\langle 1587, 684, 547, 489, 485\rangle$
5	$\{3,15\}^0 2^1 13^2 6^3 5^4$	$\langle 675, 668, 587, 527, 523\rangle$

much inside one chain. This problem can be slightly diminished by using weighted bipartite matching where the weight of the edge depends on how much the quality values of the corresponding itemsets differ from each other. This ensures only that the sum of the differences of the quality values of consecutive itemsets in the chains is minimized Thus, in long chains the minimum and the maximum quality values can still differ considerably. A more heuristic approach would be to further partition the obtained chains in such a way that the quality values of any two itemsets in the same chain do not differ too much from each other. Such partitions can be computed efficiently for several loss functions using the techniques described in Chapter 3. The minimality of the chain partition,

however, is sacrificed when the chains in the partition are further partitioned.

A simple minimum antichain partition of the collection of 0.20-frequent itemsets in the course completion database is the partition of the itemsets by their cardinalities (see Table 2.3). Especially, the 0.20-frequent items (Table 2.2) form an antichain in the collection of 0.20-frequent itemsets in the database. The frequent items can be considered as a simple summary of the collection of all frequent itemsets and the underlying transaction database, too.

Also the antichains can contain itemsets with very different quality values. Again, this problem can be diminished by further partitioning each antichain using the quality values of the patterns. □

We evaluated the condensation abilities of pattern chaining experimentally by chaining closed σ-frequent itemsets of the IPUMS Census and Internet Usage databases for several different minimum frequency thresholds $\sigma \in [0, 1]$. We chained the itemsets optimally by finding a maximum bipartite matching in the corresponding bipartite graph (Algorithm 5.1) and in a greedy manner (Algorithm 5.3) when the itemsets were ordered by their cardinalities.

As noticed in Example 5.5, the number of chains is bounded above by the cardinality of the pattern collection and below by the number of maximal patterns in the collections. In the case of closed σ-frequent itemsets this means that the number of chains is never greater than the number of closed σ-frequent itemsets and never smaller than the number of maximal σ-frequent itemsets. Furthermore, the lower bound given by the maximal itemsets might not be very tight:

Example 5.13 (slackness of lower bounds determined by maximal itemsets). If the collection of closed σ-frequent itemsets in $\mathcal{D}$ is

$$\mathcal{FC}(\sigma, \mathcal{D}) = 2^{\mathcal{I}} = \{X \subseteq \mathcal{I}\}$$

then the collection of maximal σ-frequent itemsets in $\mathcal{D}$ is

$$\mathcal{FM}(\sigma, \mathcal{D}) = \{\mathcal{I}\}$$

but largest antichain $\mathcal{A}$ in $\mathcal{FC}(\sigma, \mathcal{D})$ consists of all itemsets of car-

dinality $\lfloor |\mathcal{I}|/2 \rfloor$. Thus, the cardinality of $\mathcal{A}$ is

$$|\mathcal{A}| = \binom{|\mathcal{I}|}{\lfloor |\mathcal{I}|/2 \rfloor}.$$

□

The chaining of closed σ-frequent itemsets was computed for many different minimum frequency thresholds $\sigma \in [0, 1]$. The results are shown in Figure 5.2 and in Figure 5.3. The upper figures show the minimum frequency thresholds against the number of patterns. Each curve corresponds to some class of patterns expressed by the label of the curve. The lower figures show the minimum frequency thresholds against the relative number of closed frequent itemsets and chains with respect to the number of maximal frequent itemsets.

The number of chains in experiments were smaller than the number of closed frequent itemsets. Thus, the idea of finding a minimum chain partition seems to be useful for condensation. It is also worth to remember that the fundamental assumption in frequent itemset mining is that not very large itemsets are frequent since also all subitemsets of the frequent itemsets are frequent. This implies that the chains with respect to the partial order relation subset inclusion cannot be very long as the length of the longest chain in the frequent itemset collection is the cardinality of the largest frequent itemset. This observation makes the results even more satisfactory.

All the more interesting results were obtained when comparing the minimal and the minimum chain partitions: the greedy heuristic produced almost as small chain partitions as the computationally much more demanding approach based on maximum bipartite matchings. (Similar results were obtained also with all other transaction databases we experimented.) It is not clear, however, whether the quality of the maximal matchings is specific to closed frequent itemsets or if the results generalize to some other pattern collections as well.

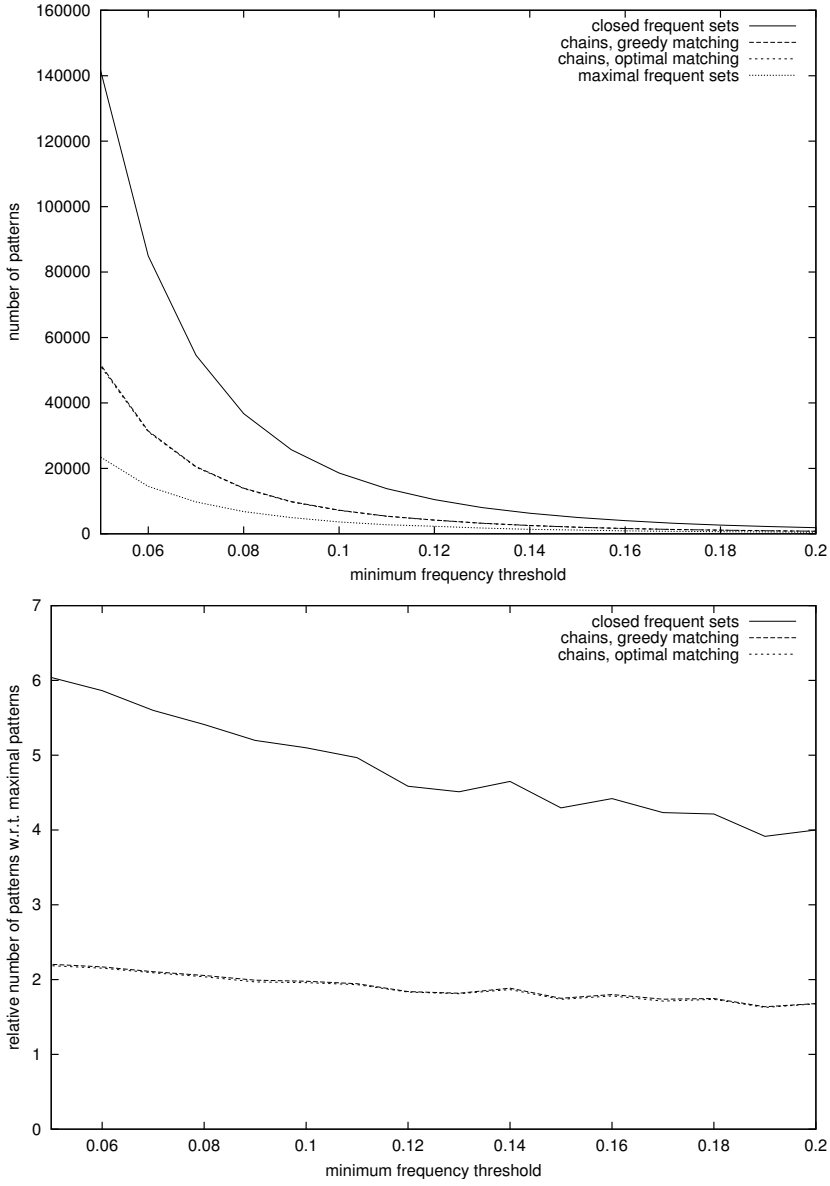

Figure 5.2: Pattern chains in Internet Usage data.

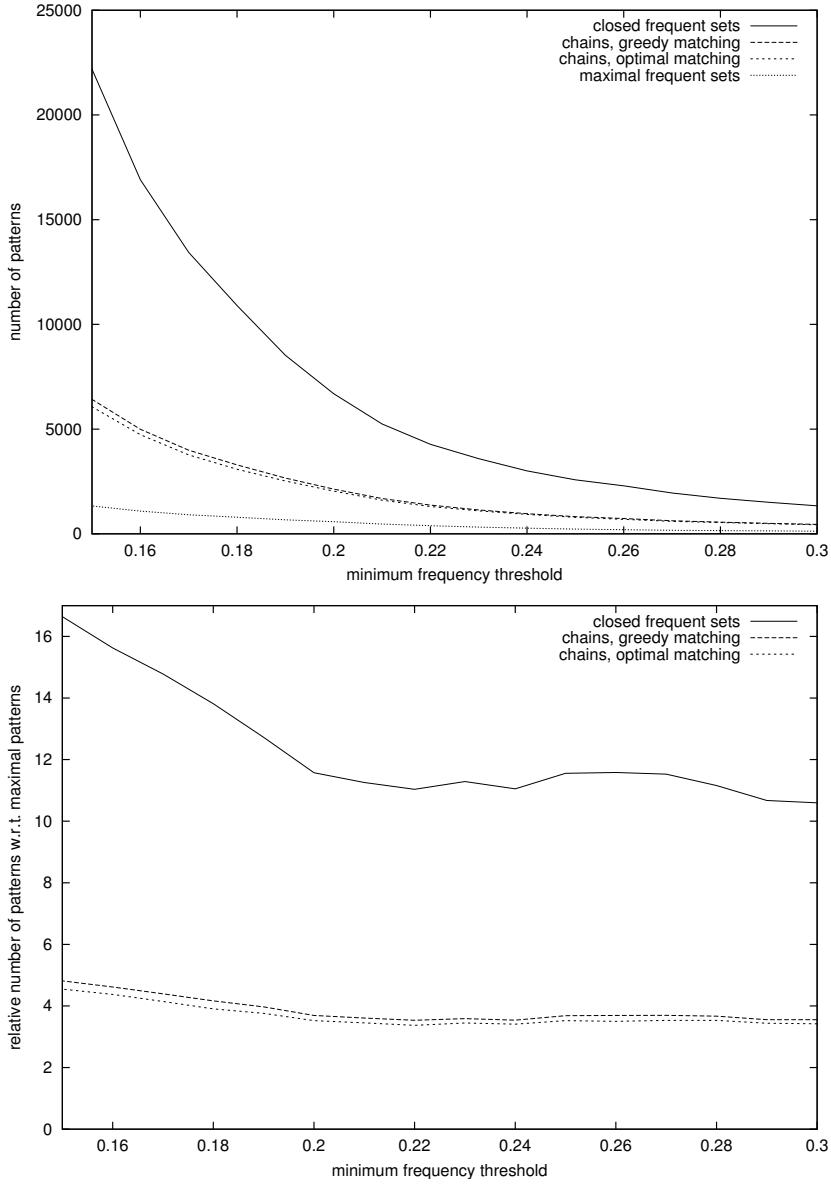

Figure 5.3: Pattern chains IPUMS Census data.

CHAPTER 6

Relating Patterns by Their Change Profiles

To make pattern collections more understandable, it would often be useful to relate the patterns to each other. In Chapter 5 the relationships between patterns were determined by a partial order over the pattern collection. The patterns can be related to each other also by their quality values. For example, absolute or relative differences between the quality values of the patterns could be used to measure their (dis)similarity. It is not immediate, however, whether comparing the quality values of two patterns actually tells much about their similarity.

An alternative approach is to measure the similarity between two patterns based on how they relate to other patterns. That is, the patterns are considered similar if they are related to other patterns similarly. This approach depends strongly on what it means to be related to other patterns. A simple solution is to consider how the quality value of the pattern has to be modified in order to obtain the quality values of its super- and subpatterns.

Example 6.1 (modifying quality values). Two simplest examples of modifications of $\phi(p)$ to $\phi(p')$ are multiplying the quality value $\phi(p)$ by the value $\phi(p')/\phi(p)$, and adding to the quality value $\phi(p)$ the value $\phi(p') - \phi(p)$.

In this chapter we restrict the modifications to the first case, i.e., modifying the quality value $\phi(p')$ of a pattern $p' \in \mathcal{P}$ from the quality value $\phi(p)$ of a pattern $p \in \mathcal{P}$ by multiplying $\phi(p)$ by

$\phi(p')/\phi(p)$. □

These modifications for one pattern can be combined as a mapping from the patterns to modifications. This mapping for a pattern p is called a *change profile* ch^p of the pattern p and each value $ch^p(p')$ is called the *change* of p with respect to $p' \in \mathcal{P}$. To simplify the considerations, the change profile ch^p is divided into two parts (adapting the terminology of [Mit82, MT97]): the *specializing* change profile ch^p_s describes the changes to the superpatterns and the *generalizing* change profile ch^p_s describes the changes to the subpatterns. When the type of the change profile is not of importance, a change profile of X is denoted by ch^X.

Example 6.2 (specializing change profiles for itemsets). Let us consider the transaction database $\mathcal{D}$ shown as Table 6.1.

Table 6.1: The transaction identifiers and the itemsets of the transactions in $\mathcal{D}$.

tid	X
1	$\{A\}$
2	$\{A, C\}$
3	$\{A, B, C\}$
4	$\{B, C\}$

The collection of 1/4-frequent itemsets in $\mathcal{D}$ and their frequencies are shown in Table 6.2.

Table 6.2: The frequent itemsets and their frequencies in $\mathcal{D}$.

X	$fr(X, \mathcal{D})$
$\emptyset$	1
A	3/4
B	1/2
C	3/4
AB	1/4
AC	1/2
ABC	1/4

For the itemsets and the frequencies, the changes in the special-

izing change profiles are of form

$$ch_s^X(Y) = \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})}.$$

Thus, the specializing change profiles of the singleton itemsets A, B and C of the 1/4-frequent itemsets in $\mathcal{D}$ are determined by the changes

$$\begin{aligned}
ch_s^A &= \left\{B \mapsto \frac{1}{3}, C \mapsto \frac{2}{3}, BC \mapsto \frac{1}{3}\right\} = \left\{B, BC \mapsto \frac{1}{3}, C \mapsto \frac{2}{3}\right\}, \\
ch_s^B &= \left\{A \mapsto \frac{1}{2}, C \mapsto 1, AC \mapsto \frac{1}{2}\right\} = \left\{A, AC \mapsto \frac{1}{2}, C \mapsto 1\right\} \text{ and} \\
ch_s^C &= \left\{A \mapsto \frac{2}{3}, B \mapsto \frac{2}{3}, AB \mapsto \frac{1}{3}\right\} = \left\{A, B \mapsto \frac{2}{3}, AB \mapsto \frac{1}{3}\right\}.
\end{aligned}$$

□

The change profiles attempt to reach from a local description of data, i.e., a pattern collection, to more global view, i.e., to relationships between the patterns in the collection. The change profiles can be used to define similarity measures between the patterns, to score the patterns and also in the condensed representations of pattern collections.

In this chapter, we introduce the concept of change profiles, a new representation of pattern collections that pursues to bridge the gap between local and global descriptions of data. We describe several variants of change profiles and study their properties. We consider different approaches to cluster change profiles and show that they are NP-hard and inapproximable for a wide variety of dissimilarity functions for change profiles, but that in practice change profiles can be used to provide reasonable clusterings. Furthermore, we suggest representing a pattern collection using approximate change profiles and propose algorithms to estimate the quality values from the approximate change profiles.

This chapter is based on the article "Change Profiles" [Mie03b]. In the remaining of the chapter we shall focus on frequent itemsets; change profiles can readily be generalized to arbitrary pattern collections with a partial order.

6.1 From Association Rules to Change Profiles

The frequency $fr(X, \mathcal{D})$ of a frequent itemset X in a transaction database $\mathcal{D}$ can be interpreted as the probability $\mathbb{P}(X)$ of the event "a transaction drawn randomly from the transaction database $\mathcal{D}$ contains itemset X" and the accuracy $acc(X \Rightarrow Y, \mathcal{D})$ of an association rule $X \Rightarrow Y$ as the conditional probability $\mathbb{P}(Y|X)$. Thus, each association rule $X \Rightarrow Y$ describes one relationship of the itemset X to other itemsets. (Empirical conditional probabilities of also different kinds of events have been studied in data mining under the name of *cubegrades* [IKA02].)

A more global view of the relationships between the frequent itemset X and other frequent itemsets can be obtained by combining the association rules $X \Rightarrow Y$ with common body into a mapping from the frequent itemsets to the interval $[0, 1]$. This mapping is called a *specializing change profile*:

Definition 6.1 (specializing change profiles). A specializing change profile of a σ-frequent itemset X in $\mathcal{D}$ is a mapping

$$ch_s^X : \{Y \subseteq \mathcal{I} : X \cup Y \in \mathcal{F}(\sigma, \mathcal{D})\} \to [0, 1]$$

consisting the accuracies of the σ-frequent rules $X \Rightarrow Y$ in $\mathcal{D}$, i.e.,

$$ch_s^X(Y) = \frac{fr(X \cup Y, \mathcal{D})}{fr(X, \mathcal{D})}$$

where $X \cup Y \in \mathcal{F}(\sigma, \mathcal{D})$.

A specializing change profile ch_s^X can be interpreted as the conditional probability $\mathbb{P}(\mathbf{Y}|X)$ where $\mathbf{Y}$ is a random variable.

Example 6.3 (specializing change profiles). Let us consider the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in $\mathcal{D}$ with $\sigma = 1/4$ where $\mathcal{D}$ is as shown in Table 6.1, and the σ-frequent itemsets and their frequencies as shown in Table 6.2. Then the specializing change profiles of $\mathcal{F}(\sigma, \mathcal{D})$ are

$$\begin{aligned} ch_s^{\emptyset} &= \left\{\emptyset \mapsto 1, A, C \mapsto \frac{3}{4}, B, AC, BC \mapsto \frac{1}{2}, AB, ABC \mapsto \frac{1}{4}\right\}, \\ ch_s^{A} &= \left\{\emptyset, A \mapsto 1, C, AC \mapsto \frac{2}{3}, B, AB, BC, ABC \mapsto \frac{1}{3}\right\}, \end{aligned}$$

$$
\begin{aligned}
ch_s^B &= \left\{\emptyset, B, C, BC \mapsto 1, A, AB, AC, ABC \mapsto \frac{1}{2}\right\}, \\
ch_s^C &= \left\{\emptyset, C \mapsto 1, A, B, AC, BC \mapsto \frac{2}{3}, AB, ABC \mapsto \frac{1}{3}\right\}, \\
ch_s^{AB} &= \{\emptyset, A, B, C, AB, AC, BC, ABC \mapsto 1\}, \\
ch_s^{AC} &= \left\{\emptyset, A, C, AC \mapsto 1, B, AB, BC, ABC \mapsto \frac{1}{2}\right\}, \\
ch_s^{BC} &= \left\{\emptyset, B, C, BC \mapsto 1, A, AB, AC, ABC \mapsto \frac{1}{2}\right\} \text{ and} \\
ch_s^{ABC} &= \{\emptyset, A, B, C, AB, AC, BC, ABC \mapsto 1\}.
\end{aligned}
$$

□

Similarly to the specializing change profiles, we can define a change profile to describe how the frequency of a σ-frequent itemset X changes when some items are removed from it. A change profile of this kind is called a *generalizing change profile*:

Definition 6.2 (generalizing change profiles). A generalizing change profile of a σ-frequent itemset X in $\mathcal{D}$ is a mapping

$$ch_g^X : \mathcal{F}(\sigma, \mathcal{D}) \to \left[1, \frac{1}{\sigma}\right]$$

consisting of the inverse accuracies of the frequent rules $X \setminus Y \Rightarrow X$, i.e.,

$$ch_g^X(Y) = \frac{fr(X \setminus Y, \mathcal{D})}{fr(X, \mathcal{D})}.$$

where $Y \subseteq \mathcal{I}$.

The generalizing change profile ch_g^X corresponds to the mapping $1/\mathbb{P}(X|X \setminus \mathbf{Y})$ where $\mathbf{Y}$ is a random variable.

Example 6.4 (generalizing change profiles). Let us consider the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in $\mathcal{D}$ with $\sigma = 1/4$ where $\mathcal{D}$ is as shown in Table 6.1, and the σ-frequent itemsets and their frequencies as shown in Table 6.2. Then the generalizing change profiles in $\mathcal{F}(\sigma, \mathcal{D})$ are

$$ch_g^{\emptyset} = \{\emptyset, A, B, C, AB, AC, BC, ABC \mapsto 1\},$$

$$
\begin{aligned}
ch_g^A &= \left\{\emptyset, B, C, BC \mapsto 1, A, AB, AC, ABC \mapsto \frac{4}{3}\right\}, \\
ch_g^B &= \{\emptyset, A, C, AC \mapsto 1, B, AB, BC, ABC \mapsto 2\}, \\
ch_g^C &= \left\{\emptyset, A, B, AB \mapsto 1, C, AC, BC, ABC \mapsto \frac{4}{3}\right\}, \\
ch_g^{AB} &= \{\emptyset, C \mapsto 1, A, AC \mapsto 2, B, BC \mapsto 3, AB, ABC \mapsto 4\}, \\
ch_g^{AC} &= \left\{\emptyset, B \mapsto 1, A, C, AB, BC \mapsto \frac{3}{2}, AC, ABC \mapsto 2\right\}, \\
ch_g^{BC} &= \left\{\emptyset, A, C, AC \mapsto 1, B, AB \mapsto \frac{3}{2}, BC, ABC \mapsto 2\right\} \text{ and} \\
ch_g^{ABC} &= \{\emptyset, C \mapsto 1, A, B, AC \mapsto 2, BC \mapsto 3, AB, ABC \mapsto 4\}.
\end{aligned}
$$

□

Each specializing and generalizing change profile ch_s^X and ch_g^X describe upper and lower neighborhoods

$$N_s(X) = \{X \cup Y \in \mathcal{F}(\sigma, \mathcal{D}) : Y \subseteq \mathcal{I}\}$$

and

$$N_g(X) = \{X \setminus Y \in \mathcal{F}(\sigma, \mathcal{D}) : Y \subseteq \mathcal{I}\}$$

of the frequent itemset X in the collection $\mathcal{F}(\sigma, \mathcal{D})$, respectively. The neighborhood

$$N(X) = N_s(X) \cup N_g(X)$$

of X consists of the frequent itemsets $Y \in N_s(X)$ that contain the frequent itemset X and the frequent itemsets $X \setminus Y \in N_g(X)$ that are contained in X, i.e., the frequent itemsets that are comparable with X.

As seen in Example 6.3 and Example 6.4, the change profiles (Definition 6.1 and Definition 6.2) are often highly redundant. This is due to the following properties of itemsets:

Observation 6.1. *Let $X, Y \subseteq \mathcal{I}$. Then*

$$X \cup Y = X \cup (Y \setminus X)$$

and

$$X \setminus Y = X \setminus (Y \cap X).$$

The number of defined values of the change profile is reduced (without losing any information) considerably by exploiting Observation 6.1.

Example 6.5 (redundancy in change profiles). Let X be a frequent itemset with only one frequent superitemset $X \cup \{A\}$ where $A \notin X$. There are $2^{|X|+1}$ subitemsets of $X \cup \{A\}$. The first equation in Observation 6.1 implies that frequency of $X \cup Y$ is equal to the frequency of X if $Y \subseteq X$. Thus, the specializing changes $ch_s^X(Y) = 1$ for all $Y \subseteq X$ can be neglected. Furthermore, the specializing changes $ch_s^X(Y \cup \{A\})$ are equal for all $Y \subseteq X$ and it is sufficient to store just the specializing change $ch_s^X(\{A\})$. This reduces the size of the specializing change profile of X by factor $2^{|X|+1}$.

Let X be an arbitrary frequent itemset and let $\mathcal{F}(\sigma, \mathcal{D})$. Based on the second equation of Observation 6.1, there is no need to store changes $ch_g^X(Y) = 1$ for $Y \subseteq \mathcal{I}$ such that $Y \not\subseteq X$. This reduces the number of of changes in the generalizing change profile of X by factor $2^{|\mathcal{I}|-|X|} = 2^{|\mathcal{I} \setminus X|}$. □

The change profiles with redundancy reduced as in Example 6.5 are called *concise change profiles*:

Definition 6.3 (concise specializing change profiles). A concise specializing change profile cch_s^X is a restriction of a specializing change profile ch_s^X to itemsets Y such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{F}(\sigma, \mathcal{D})$.

Definition 6.4 (concise generalizing change profiles). A concise generalizing change profile cch_g^X is a restriction of a generalizing change profile ch_g^X to itemsets Y such that $Y \subseteq X$.

Example 6.6 (concise change profiles). Let us consider the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in $\mathcal{D}$ with $\sigma = 1/4$ where $\mathcal{D}$ is as shown in Table 6.1, and the σ-frequent itemsets and their frequencies as shown in Table 6.2. The concise specializing change profiles of $\mathcal{F}(\sigma, \mathcal{D})$ are

$$cch_s^{\emptyset} = \left\{\emptyset \mapsto 1, A, C \mapsto \frac{3}{4}, B, AC, BC \mapsto \frac{1}{2}, AB, ABC \mapsto \frac{1}{4}\right\},$$

$$cch_s^{A} = \left\{\emptyset \mapsto 1, C \mapsto \frac{2}{3}, B, BC \mapsto \frac{1}{3}\right\},$$

$$
\begin{aligned}
cch_s^B &= \left\{\emptyset, C \mapsto 1, A, AC \mapsto \frac{1}{2}\right\}, \\
cch_s^C &= \left\{\emptyset \mapsto 1, A, B \mapsto \frac{2}{3}, AB \mapsto \frac{1}{3}\right\}, \\
cch_s^{AB} &= \{\emptyset, C \mapsto 1\}, \\
cch_s^{AC} &= \left\{\emptyset \mapsto 1, B \mapsto \frac{1}{2}\right\}, \\
cch_s^{BC} &= \left\{\emptyset \mapsto 1, A \mapsto \frac{1}{2}\right\} \text{ and} \\
cch_s^{ABC} &= \{\emptyset \mapsto 1\}
\end{aligned}
$$

and the concise generalizing change profiles of $\mathcal{F}(\sigma, \mathcal{D})$ are

$$
\begin{aligned}
cch_g^{\emptyset} &= \{\emptyset \mapsto 1\}, \\
cch_g^{A} &= \left\{\emptyset \mapsto 1, A \mapsto \frac{4}{3}\right\}, \\
cch_g^{B} &= \{\emptyset \mapsto 1, B \mapsto 2\}, \\
cch_g^{C} &= \left\{\emptyset \mapsto 1, C \mapsto \frac{4}{3}\right\}, \\
cch_g^{AB} &= \{\emptyset \mapsto 1, A \mapsto 2, B \mapsto 3, AB \mapsto 4\}, \\
cch_g^{AC} &= \left\{\emptyset \mapsto 1, A, C \mapsto \frac{3}{2}, AC \mapsto 2\right\}, \\
cch_g^{BC} &= \left\{\emptyset, C \mapsto 1, B \mapsto \frac{3}{2}, BC \mapsto 2\right\} \text{ and} \\
cch_g^{ABC} &= \{\emptyset, C \mapsto 1, A, B, AC \mapsto 2, BC \mapsto 3, AB, ABC \mapsto 4\}.
\end{aligned}
$$

□

The concise change profiles can be interpreted as affine axis-parallel subspaces of $\mathbb{R}^{|\mathcal{F}(\sigma,\mathcal{D})|}$ (i.e., affine hyperplanes in $\mathbb{R}^{|\mathcal{F}(\sigma,\mathcal{D})|}$) that are indexed

- by itemsets Y such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{F}(\sigma, \mathcal{D})$ in the specializing case, and
- by itemsets Y such that $Y \subseteq X$ in the generalizing case.

The concise change profiles for a frequent itemset collection $\mathcal{F}(\sigma, \mathcal{D})$ can be computed efficiently by Algorithm 6.1.

Algorithm 6.1 Generation of concise change profiles.

Input: The collection $\mathcal{F}(\sigma, \mathcal{D})$ of σ-frequent itemsets in a transaction database $\mathcal{D}$ and their frequencies.

Output: The concise specializing change profiles and the concise generalizing change profiles of $\mathcal{F}(\sigma, \mathcal{D})$.

```
1: function CHANGE-PROFILES(F(σ, D), fr)
2:     for all X ∈ F(σ, D) do
3:         for all Y ⊆ X do
4:             cch_s^{X\Y}(Y) ← fr(X, D)/fr(X \ Y, D)
5:             cch_g^X(Y) ← fr(X \ Y, D)/fr(X, D)
6:         end for
7:     end for
8:     return ⟨cch_s, cch_g⟩
9: end function
```

As shown in Example 6.7, the neighborhoods of even the concise change profiles can be too large.

Example 6.7 (redundancy in concise change profiles). Let X be an itemset in the collection $\mathcal{F}(\sigma, \mathcal{D})$. Then $\left|cch_s^{\emptyset}\right| \geq 2^{|X|}$ and $\left|cch_g^X\right| \geq 2^{|X|}$ in $\mathcal{F}(\sigma, \mathcal{D})$. □

Thus, the following definitions of association rules, we define *simple specializing change profiles* and *simple generalizing change profiles*:

Definition 6.5 (simple change profiles). A simple specializing (generalizing) change profile sch_s^X (sch_g^X) is restriction of cch_s^X (cch_g^X) to singleton itemsets Y.

Example 6.8 (simple change profiles). Let us consider the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in $\mathcal{D}$ with $\sigma = 1/4$ where $\mathcal{D}$ is as shown in Table 6.1, and the σ-frequent itemsets and their frequencies as shown in Table 6.2. The simple specializing change profiles of $\mathcal{F}(\sigma, \mathcal{D})$ are

$$\begin{aligned} sch_s^{\emptyset} &= \left\{A, C \mapsto \frac{3}{4}, B \mapsto \frac{1}{2}\right\}, \\ sch_s^{A} &= \left\{C \mapsto \frac{2}{3}, B \mapsto \frac{1}{3}\right\}, \end{aligned}$$

$$
\begin{aligned}
sch_s^B &= \left\{C \mapsto 1, A \mapsto \frac{1}{2}\right\}, \\
sch_s^C &= \left\{A, B \mapsto \frac{2}{3}\right\}, \\
sch_s^{AB} &= \{C \mapsto 1\}, \\
sch_s^{AC} &= \left\{B \mapsto \frac{1}{2}\right\}, \\
sch_s^{BC} &= \left\{A \mapsto \frac{1}{2}\right\} \text{ and} \\
sch_s^{ABC} &= \{\}
\end{aligned}
$$

and the simple generalizing change profiles of $\mathcal{F}(\sigma, \mathcal{D})$ are

$$
\begin{aligned}
sch_g^{\emptyset} &= \{\}, \\
sch_g^A &= \left\{A \mapsto \frac{4}{3}\right\}, \\
sch_g^B &= \{B \mapsto 2\}, \\
sch_g^C &= \left\{C \mapsto \frac{4}{3}\right\}, \\
sch_g^{AB} &= \{A \mapsto 2, B \mapsto 3\}, \\
sch_g^{AC} &= \left\{A, C \mapsto \frac{3}{2}\right\}, \\
sch_g^{BC} &= \left\{C \mapsto 1, B \mapsto \frac{3}{2}\right\} \text{ and} \\
sch_g^{ABC} &= \{C \mapsto 1, A, B \mapsto 2\}.
\end{aligned}
$$

□

The number of bits needed for representing a simple change profile is at most $|\mathcal{I}| \log |\mathcal{D}|$: Each change profile can be described as a length-$|\mathcal{I}|$ vector of changes as the number of singleton subsets of the set $\mathcal{I}$ of items is $|\mathcal{I}|$. Each change can be described using at most $\log |\mathcal{D}|$ bits since there are at most as many different possible changes from a given itemset to any other itemset as there are are transactions in $\mathcal{D}$. This upper bound can sometimes be quite loose as shown by Example 6.9.

Example 6.9 (loose upper bounds for simple generalizing change profiles). Let the set $\mathcal{I}$ of items be large. Then the above

upper bound is often very loose: The number of itemsets in the collection $\mathcal{F}(\sigma, \mathcal{D})$ of σ-frequent itemsets in $\mathcal{D}$ is exponential in the cardinality of the largest itemset in $\mathcal{F}(\sigma, \mathcal{D})$. Thus, the largest itemset X in $\mathcal{F}(\sigma, \mathcal{D})$ has to be moderately small in order to be able to represent the collection $\mathcal{F}(\sigma, \mathcal{D})$ in a reasonable space. Thus, in this case, the upper bound for binary description of a simple generalizing change profile of an itemset $X \in \mathcal{F}(\sigma, \mathcal{D})$ should rather be $|X|\,(\log|\mathcal{I}|)\,(\log|\mathcal{D}|)$. □

6.2 Clustering the Change Profiles

In order to be able to find groups of similar change profiles, it would be useful to be able to somehow measure the (similarity or) dissimilarity between change profiles ch^X and ch^Y.

The dissimilarity between the change profiles ch^X and ch^Y can be defined to be their distance in their common domain $Dom(ch^X) \cap Dom(ch^Y)$ with respect to some distance function d. A complementary approach would be to focus on the differences in the structure of the pattern collection, e.g., to measure the difference between two change profiles by computing the symmetric difference of their domains. This kind of dissimilarity function concentrates solely on the structure of the pattern collection and thus neglects the frequencies. A sophisticated dissimilarity should probably consist of both points of view.

We shall focus on the first one. The only requirements we have for a distance function are given by Definition 6.6.

Definition 6.6 (a distance function). A function d is a distance function if

$$d(ch^X, ch^Y) = 0 \iff ch^X(Z) = ch^Y(Z)$$

holds for all $Z \in Dom(ch^X) \cap Dom(ch^Y)$.

There are several ways to define what is a good clustering and each approach has its own strengths and weaknesses [EC02, Kle02]. A simple way to group the change profiles based on a dissimilarity function defined in their (pairwise) common domains is to allow two change profiles ch^X and ch^Y to be in the same group only if $d(ch^X, ch^Y) = 0$. Thus, the problem can be formulated as follows.

Problem 6.1 (change profile packing). Given a collection $\mathcal{C}h$ of change profiles and a dissimilarity function d, find a partition of $\mathcal{C}h$ into groups $\mathcal{C}h_1, \ldots, \mathcal{C}h_k$ with the smallest possible k such that $d(ch^X, ch^Y) = 0$ holds for all $ch^X, ch^Y \in \mathcal{C}h_i$ with $1 \leq i \leq k$.

Unfortunately, the problem seems to be very difficult. Namely, it can be shown to be at least as difficult as the minimum graph coloring problem:

Problem 6.2 (minimum graph coloring [ACK+99]). Given a graph $G = \langle V, E \rangle$, find a labeling $label : V \to \mathbb{N}$ of the vertices with smallest number $|label(V)|$ of different labels such that if $u, v \in V$ are adjacent then $label(u) \neq label(v)$.

Theorem 6.1. *The change profile packing problem is at least as hard as the minimum graph coloring problem.*

Proof. Let $G = \langle V, E \rangle$ be an instance of the minimum graph coloring problem where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and $E = \{e_1, \ldots, e_m\}$ is the set of edges.

We reduce the minimum graph coloring problem (Problem 6.2) to the change profile packing problem (Problem 6.1) by first constructing an instance $\langle \sigma, \mathcal{D} \rangle$ of the frequent itemset mining problem and then showing that the collection $\mathcal{C}h$ of specializing change profiles computed from the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in $\mathcal{D}$ and their frequencies can be partitioned into $k + 2$ subcollections $\mathcal{C}h_1, \ldots, \mathcal{C}h_{k+2}$ if and only if the graph G is k-colorable. To simplify the description, we shall consider, without loss of generality, simple change profiles instead of change profiles in general.

The set $\mathcal{I}$ of items consists of elements in $V \cup E$. For each vertex $v_i \in V$ there are $3n$ transactions with transaction identifiers $\langle i, 1 \rangle, \ldots, \langle i, 3n \rangle$. Thus, in total there are $3n^2$ transactions in $\mathcal{D}$.

Each transaction $\langle \langle i, j \rangle, X \rangle$ contains the vertex $v_i \in V$. Transactions $\langle \langle i, 3(j-1)+1 \rangle, X \rangle$ and $\langle \langle i, 3(j-1)+2 \rangle, X \rangle$ contain an edge $\{v_i, v_j\} \in E$ if and only if $i < j$. The transaction $\langle \langle i, 3j \rangle, X \rangle$ contains an edge $\{v_i, v_j\} \in E$ if $i > j$.

Let the minimum frequency threshold σ be $1/\left(3n^2\right)$. Then the collection $\mathcal{F}(\sigma, \mathcal{D})$ consists of the empty itemset $\emptyset$ the singleton itemsets $\{v_1\}, \ldots, \{v_n\}, \{e_1\}, \ldots, \{e_m\}$ and 2-itemsets $\{v_i, e\}$ where $v_i \in e \in E$. Thus, the cardinality of $\mathcal{F}(\sigma, \mathcal{D})$ is polynomial in

the number of vertices of G. The simple change profiles of $\mathcal{F}(\sigma, \mathcal{D})$ are the following ones:

$$\begin{aligned} sch_s^{\emptyset}(x) &= \begin{cases} 1/n & \text{if } x \in V \\ 1/n^2 & \text{if } x \in E \end{cases} \\ sch_s^{e}(v) &= 1 \qquad\qquad \text{if } v \in e \\ sch_s^{v_i}(\{v_i, v_j\}) &= \begin{cases} 2/(3n) & \text{if } i < j, \{v_i, v_j\} \notin E \\ 1/(3n) & \text{if } i > j, \{v_i, v_j\} \notin E \end{cases} \end{aligned}$$

Clearly,

- $d(sch_s^{\emptyset}, sch_s^{v}) > 0$ for $sch_s^{\emptyset}$ and all sch_s^{v} where $v \in V$,
- $d(sch_s^{\emptyset}, sch_s^{e}) > 0$ for $sch_s^{\emptyset}$ and all sch_s^{e} where $e \in E$, and
- $d(sch_s^{v}, sch_s^{e}) > 0$ for all sch_s^{v} and sch_s^{e} where $v \in V$ and $e \in E$.

On one hand, no two of $ch_s^{\emptyset}$, ch_s^{v} and sch_s^{e} can be in the same group for any $v \in V, e \in E$. On the other hand, all sch_s^{e} can be packed into one set $\mathcal{C}h_{k+1}$ and $sch_s^{\emptyset}$ always needs its own set $\mathcal{C}h_{k+1}$.

Hence, it is sufficient to show that the simple specializing change profiles sch_s^{v} can be partitioned into k sets $\mathcal{C}h_1, \ldots, \mathcal{C}h_k$ without any error if and only if the graph G is k-colorable. No two simple specializing change profiles $sch_s^{v_i}$ and $sch_s^{v_j}$ with $\{v_i, v_j\} \in E$ can be in the same group since $sch_s^{v_i}(\{v_i, v_j\}) \neq sch_s^{v_j}(\{v_i, v_j\})$. If $\{v_i, v_j\} \notin E$ then $Dom(sch_s^{v_i}) \cap Dom(sch_s^{v_j}) = \emptyset$, i.e., $sch_s^{v_i}$ and $sch_s^{v_j}$ can be in the same group.

As the minimum graph coloring problem can be mapped to the change profile packing for specializing change profiles in polynomial time, the latter is at least as hard as the minimum graph coloring problem. □

The minimum graph coloring problem is hard to approximate within $|V|^{1-\epsilon}$ for any $\epsilon > 0$ unless NP=ZPP [FK98]. (Recall that the complexity class ZPP consists of the decision problems that have randomized algorithms that always make the right decision and run in expected polynomial time [Pap95].) Assuming that the graph is connected we get from the above mapping from graphs to change profiles the following rough upper bound

$$|\mathcal{C}h| = 1 + |V| + |E| \leq 1 + |V| + \binom{|V|}{2} = \mathcal{O}(|V|^2).$$

Therefore, the change profile packing problem is hard to approximate within $\Omega(|\mathcal{C}h|^{(1/2)-\epsilon})$ for any $\epsilon > 0$ unless NP=ZPP.

Although the inapproximability results seem to be devastating, there are efficient heuristics, such as the first-fit and the best-fit heuristics [CJCG+02], that might be able to find sufficiently good partitions efficiently. However, the usefulness of such heuristics depends on the actual transaction databases inducing the collections of frequent itemsets.

The requirement that two change profiles ch^X and ch^Y can be in the same group $\mathcal{C}h_i$ only if $d(ch^X, ch^Y) = 0$ might be too strict. This restriction can be relaxed also by discretizing the frequencies of the frequent itemsets or the changes in the change profiles. (Recall that in Section 3.2 we have seen that discretizations minimizing several different loss functions can be found efficiently.)

Instead of minimizing the number of clusters, one could minimize the error for a fixed number of clusters. This kind of clustering is called a *k-clustering*. The problem of finding good k-clusterings is well-studied and good approximation algorithms are known if the dissimilarity function is a metric [Das02, dlVKKR03, FG88]. The problem of finding the k-clustering of change profiles that minimizes the sum of intracluster distances can be defined as follows:

Problem 6.3 (minimum sum of distances k-clustering of change profiles). Given a collection $\mathcal{C}h$ of change profiles, a distance function $d : \mathcal{C}h \times \mathcal{C}h \to \mathbb{R}$ and a positive integer k, find the partition of $\mathcal{C}h$ into k groups $\mathcal{C}h_1, \ldots, \mathcal{C}h_k$ such that

$$\sum_{i=1}^{k} \sum_{ch^X, ch^Y \in \mathcal{C}h_i} d(ch^X, ch^Y)$$

is minimized.

Unfortunately, it turn out that a dissimilarity function that is defined to consist of the dissimilarities between the change profiles in their common domains cannot be a metric since it cannot satisfy even the triangle inequality in general:

Proposition 6.1. *A function d that measures the distance between the change profiles ch^X and ch^Y in their common domain $Dom(ch^X) \cap Dom(ch^Y)$ is not a metric.*

Proof. Let ch^X, ch^Y and ch^Z be three change profiles such that

$$Dom(ch^X) \cap Dom(ch^Y) = \emptyset = Dom(ch^Y) \cap Dom(ch^Z)$$

but $d(ch^X, ch^Z) > 0$ (and thus $Dom(ch^X) \cap Dom(ch^Z)$). The distance between these change profiles do not satisfy triangle inequality since

$$d(ch^X, ch^Z) > 0 = d(ch^X, ch^Y) + d(ch^Y, ch^Z).$$

Thus, such a distance cannot be not a metric. □

It turns out that the minimum k-clustering of specializing change profiles is even worse than the change profile packing problem in the sense of approximability as combining Theorem 6.1 and Proposition 6.1 we get:

Theorem 6.2. *The minimum sum of distances k-clustering (Problem 6.3) of specializing change profiles cannot be approximated within any ratio.*

Proof. If we could approximate k-clustering of specializing change profiles, then we could, by Theorem 6.1 and Proposition 6.1, solve the minimum graph coloring problem exactly. Namely, if a graph is k-colorable, then the corresponding change profiles have k-clustering with the sum of intracluster distances being zero. Thus, an approximation algorithm with any approximation guarantees would find a solution with error zero if and only if the corresponding graph is k-colorable. □

A major goal in the clustering of the change profiles is to further understand the relationships between the frequent itemsets (and collection of interesting patterns in general). As the nature of pattern discovery is exploratory, defining a maximum number of clusters or a maximum dissimilarity threshold might be difficult and unnecessary. Fixing these parameters in advance can be avoided by searching for a hierarchical clustering, instead [HTF01].

A hierarchical clustering of $\mathcal{C}h$ is a recursive partition of the elements to $1, 2, \ldots, |\mathcal{C}h|$ clusters. It is most fortunate for the exploratory data analysis point of view that in the case of hierarchical clustering, the clusterings of all cardinalities can be visualized in the same time by a tree (often called a *dendrogram*).

There are two main types of hierarchical clustering: agglomerative and divisive (see also Example 3.10). The first begins with $|\mathcal{C}h|$ singleton clusters and recursively merges them and the latter recursively partitions the set $\mathcal{C}h$. Both are optimal in certain sense: each agglomerative (divisive) hierarchical clustering of $\mathcal{C}h$ into k groups is optimal with respect to the clustering into $k+1$ groups ($k-1$ groups) determined by the same agglomerative (divisive) hierarchical clustering.

Example 6.10 (a hierarchical clustering of change profiles). Let us consider subsets of the simple change profiles of Example 6.8. As the distance function between the change profiles, we use the sum of absolute distances in the common domain. (For brevity, we write the simple change profiles as 3-tuples. The positions denote the changes with respect to A, B and C, respectively, $*$ denoting undefined value.)

First, let us consider the simple specializing change profiles

$$\begin{aligned} sch_s^A &= \left\langle *, \frac{1}{3}, \frac{2}{3} \right\rangle, \\ sch_s^B &= \left\langle \frac{1}{2}, *, 1 \right\rangle \quad \text{and} \\ sch_s^C &= \left\langle \frac{2}{3}, \frac{2}{3}, * \right\rangle. \end{aligned}$$

The sums of the absolute differences in their common domains are

$$\begin{aligned} d(sch_s^A, sch_s^B) &= \left|sch_s^A(C) - sch_s^B(C)\right| = \left|\frac{2}{3} - 1\right| = \frac{1}{3}, \\ d(sch_s^A, sch_s^C) &= \left|sch_s^A(B) - sch_s^C(B)\right| = \left|\frac{1}{3} - \frac{2}{3}\right| = \frac{1}{3} \quad \text{and} \\ d(sch_s^B, sch_s^C) &= \left|sch_s^B(A) - sch_s^C(A)\right| = \left|\frac{1}{2} - \frac{2}{3}\right| = \frac{1}{6}. \end{aligned}$$

Agglomerative and divisive hierarchical clusterings suggest both that the clustering into two groups is $\left\{sch_s^A\right\}$ and $\left\{sch_s^B, sch_s^C\right\}$ with the sums of distances 0 and 1/6, respectively.

Second, let us consider the simple generalizing change profiles

$$sch_g^{AB} = \langle 2, 3, * \rangle,$$

$$sch_g^{AC} = \left\langle \frac{3}{2}, *, \frac{3}{2} \right\rangle \quad \text{and}$$
$$sch_g^{BC} = \left\langle *, \frac{3}{2}, 1 \right\rangle.$$

The sums of the absolute differences in their common domains are

$$d(sch_g^{AB}, sch_g^{AC}) = \left|sch_g^{AB}(A) - sch_s^{AC}(A)\right| = \left|2 - \frac{3}{2}\right| = \frac{1}{2},$$
$$d(sch_g^{AB}, sch_g^{BC}) = \left|sch_s^{AB}(B) - sch_s^{BC}(B)\right| = \left|3 - \frac{3}{2}\right| = \frac{3}{2} \quad \text{and}$$
$$d(sch_g^{AC}, sch_g^{BC}) = \left|sch_s^{AC}(C) - sch_s^{BC}(C)\right| = \left|\frac{3}{2} - 1\right| = \frac{1}{2}.$$

This time there are two equally good clusterings to two groups: the only requirement is that sch_g^{AB} and sch^{BC} are in different clusters. The sums of the distances for the singleton cluster and the cluster of two change profiles are 0 and 1/2, respectively.

The dendrogram visualizations of the hierarchical clusterings are shown in Figure 6.1. □

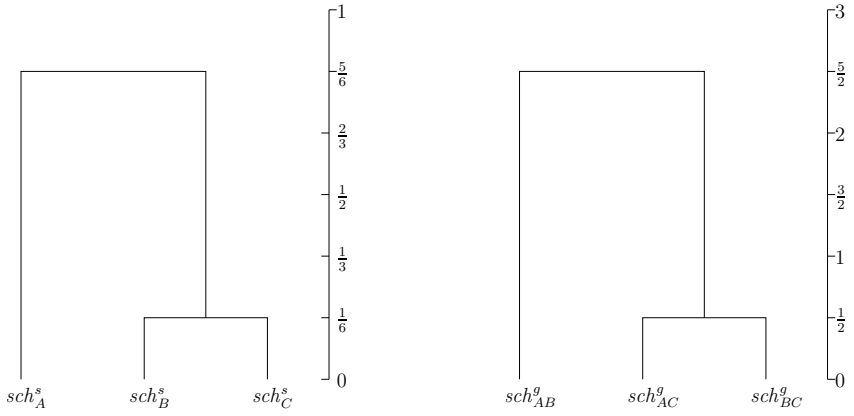

Figure 6.1: The dendrogram of the hierarchical clusterings of the simple specializing change profiles sch_A^s, sch_B^s and sch_C^s, and the simple generalizing change profiles sch_{AB}^g, sch_{AC}^g and sch_{BC}^g, respectively. The y-axis corresponds to the sum of absolute errors.

The divisive strategy seems to be more suitable for clustering the change profiles since the dissimilarity functions we consider are defined to be distances between the change profiles in their (pairwise)

common domains: The agglomerative clustering first puts more or less arbitrarily the change profiles with disjoint domains into the clusters. The choices made in the first few merges can cause major differences in the clusterings into smaller number of clusters, although the groups of change profiles with disjoint domains are probably quite unimportant for determining the complete hierarchical clustering. Contrary to the agglomerative clustering, the divisive clustering concentrates first on the nonzero distances and thus the change profiles with disjoint domains do not bias the whole hierarchical clustering.

Example 6.11 (hierachical clustering of the simple specializing change profiles of the 34 most frequent courses in the course completion database). To illustrate the hierarchical clustering of change profiles, let us consider the simple specializing change profiles of the 34 most frequent items (i.e., the courses shown in Table 2.2) in the collection consisting of all 1- and 2-subsets of the 34 most frequent items in the course completion database (see Subsection 2.2.1).

The agglomerative clustering of the simple change profiles using the average distances between the courses as the merging criterion (i.e., the average linkage hierarchical clustering) is shown in Figure 6.2.

The clustering of the specializing change profiles captures many important dependencies between the courses. For example, the courses 8 (English Oral Test) and 11 (Oral and Written Skills in Swedish) are close to each other. Also, the courses 16 (Approbatur in Mathematics I) and 32 (Approbatur in Mathematics II) are in the same branch although their ranking with respect to their frequencies differ quite much. Furthermore, the courses 18 (Discrete Mathematics I) and 27 (Logic I) are close to each other as their content overlap considerably and they form two thirds of an alternative for the courses 16 and 32 to obtain Approbatur in Mathematics.

The courses 14 (Scientific Writing) and 20 (Maturity Test in Finnish) are naturally close to each other since it is very customary to take the maturity test in the end of the Scientific Writing course. Also the course 31 (Software Engineering Project) is close to the course 14. The explanation for this is that both courses have almost the same prerequisites and both are needed for the Bachelor of

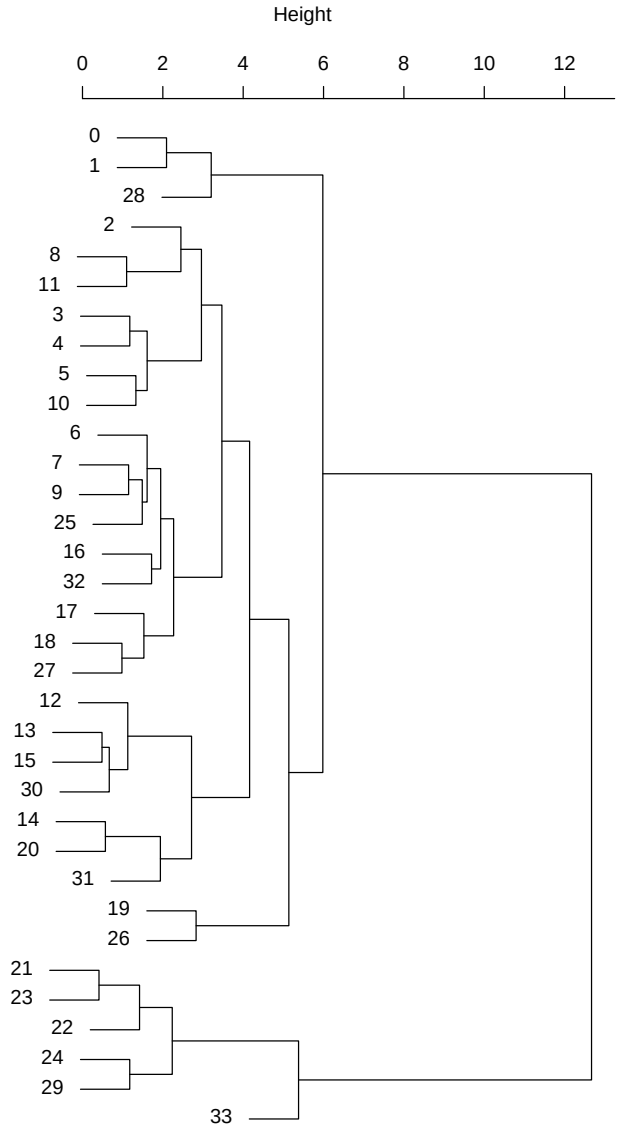

Figure 6.2: The hierarchical clustering of the 34 most frequent items based on their simple specializing change profiles.

Science degree with Computer Science as the major subject.

The courses 5 (Information Systems) and 10 (Programming in Pascal) are deprecated and they replaced in the current curriculum by the courses 21 (Introduction to Application Design), 23 (Introduction to Databases), 24 (Introduction to Programming) and 22 (Programming in Java). Similarly, the course 12 (Information Systems Project) has been replaced by the course 33 (Database Application Project). The courses close to the course 12, namely the courses 13 (Concurrent Systems), 15 (Databases Systems I) and 30 (Data Communications), are also deprecated versions although there are courses with the same names in the current curriculum.

As the simplest comparison, the clustering of items based on the absolute differences between their frequencies is shown in Figure 6.3. However, the clustering based on frequencies does not capture much of the relationships between the courses. This is not very surprising since the frequencies of the courses contain quite little information about the courses.

A more realistic comparison would be the average linkage hierarchical clustering based on the Hamming distances between the items. The Hamming distance between the two items in a transaction database is the number transactions in the database containing one of the items but not both of them, i.e., the Hamming distance between items A and B in a transaction database $\mathcal{D}$ is

$$\begin{aligned} d_H(A, B, \mathcal{D}) &= |cover(A, \mathcal{D}) \setminus cover(B, \mathcal{D})| + \\ &\quad |cover(B, \mathcal{D}) \setminus cover(A, \mathcal{D})| . \end{aligned}$$

Such a clustering is shown in Figure 6.4.

The results obtained using Hamming distance are quite similar to the results obtained using the change profiles. There are slight differences, however. For example, the courses 0, 1 and 28 that are close to each other in Figure 6.2, are quite far from each other in in Figure 6.4. The courses 17 and 31 are close to each other in Figure 6.4, whereas the course 31 is in the same cluster with the courses 14 and 20 in Figure 6.2.

The courses 18, 26 and 27 form a cluster in Figure 6.4 forming an alternative Approbatur in Mathematics but in Figure 6.2 the course 26 is together with the course 19 which is mathematically demanding for many students. (In Figure 6.4 the course 19 is in the same cluster with the course 33.)

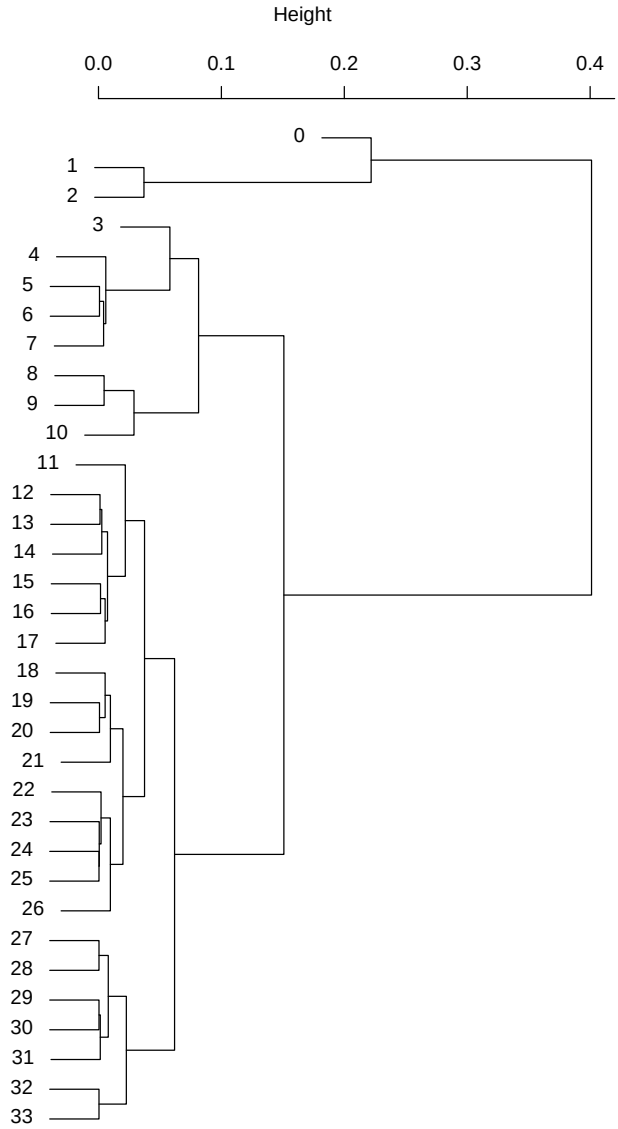

Figure 6.3: The hierarchical clustering of the 34 most frequent items based on the absolute differences between their frequencies.

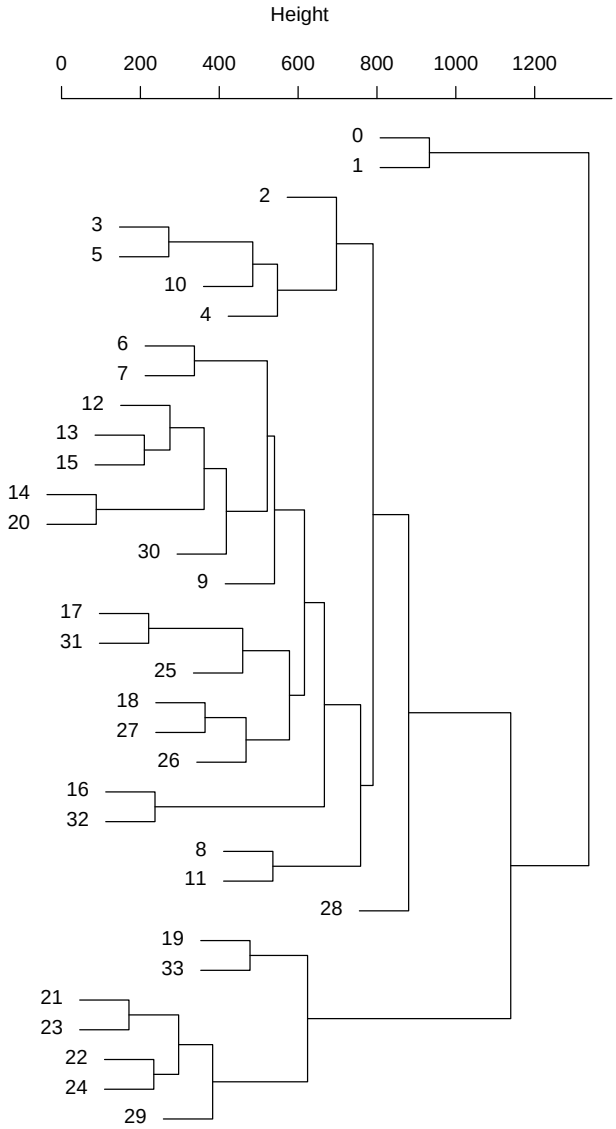

Figure 6.4: The hierarchical clustering of the 34 most frequent items based on their Hamming distances.

In general, the hierarchical clustering with respect to Hamming distances seems to capture courses forming entities (for example, pairs of courses that earlier formed one course), whereas the hierarchical clustering of change profiles seems to be related more closely to the essence of the courses in a broader way. This is in line with the fact that the Hamming distances between the items compare the co-occurrences of the items directly, whereas the distances between the change profiles measure the similarity of the behavior of the items with respect to their whole neighborhoods except each other. Note that the change profiles do not depend on the actual frequencies of the items but the Hamming distances are strongly affected by the frequencies.

The Hamming distance is symmetric with respect to whether the item is contained in the transaction. As the transaction databases often correspond to sparse binary matrices, this assumption about the symmetry of presence and absence is not always justified. The similarity between two items could be measured by the number of transactions containing them both instead of counting the number of transactions containing either both or neither of them. This is also equal to the scalar product between the binary vectors representing the covers of the items. To transform similarity to dissimilarity, we subtract the similarity value from the cardinality of the database. Thus, the dissimilarity is

$$|\mathcal{D}| - |cover(A, \mathcal{D}) \cap cover(B, \mathcal{D})| = supp(\emptyset, \mathcal{D}) - supp(AB, \mathcal{D}).$$

The hierarchical clustering for this dissimilarity is shown in Figure 6.5. The results are unfortunately similar to the clustering based on the frequencies (Figure 6.3) although also some related courses, such as the courses 16 and 32, are close to each other in the dendrogram regardless of their dissimilar frequencies.

The change profiles used in the clustering in Figure 6.2 can be computed from the frequencies of the 34 most frequent items (Figure 6.3) and the frequencies of the 2-itemsets formed from the 34 most frequent itemsets (Figure 6.5). Thus, in this particular case, the specializing simple change profiles can be considered as normalizations of the frequencies of the 2-itemsets by the frequencies of the items. Another approach to normalize the frequencies of the 2-itemsets by the frequencies of the items is as follows. The supports of the 2-itemsets can be considered the scalar products between

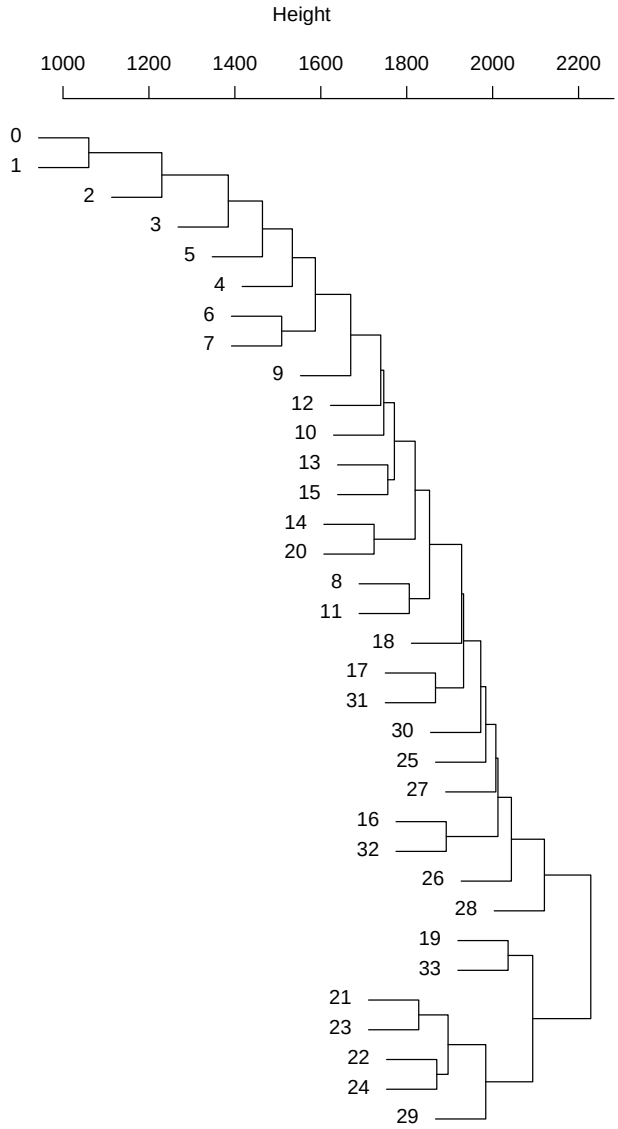

Figure 6.5: The hierarchical clustering of the 34 most frequent items based on the their scalar products.

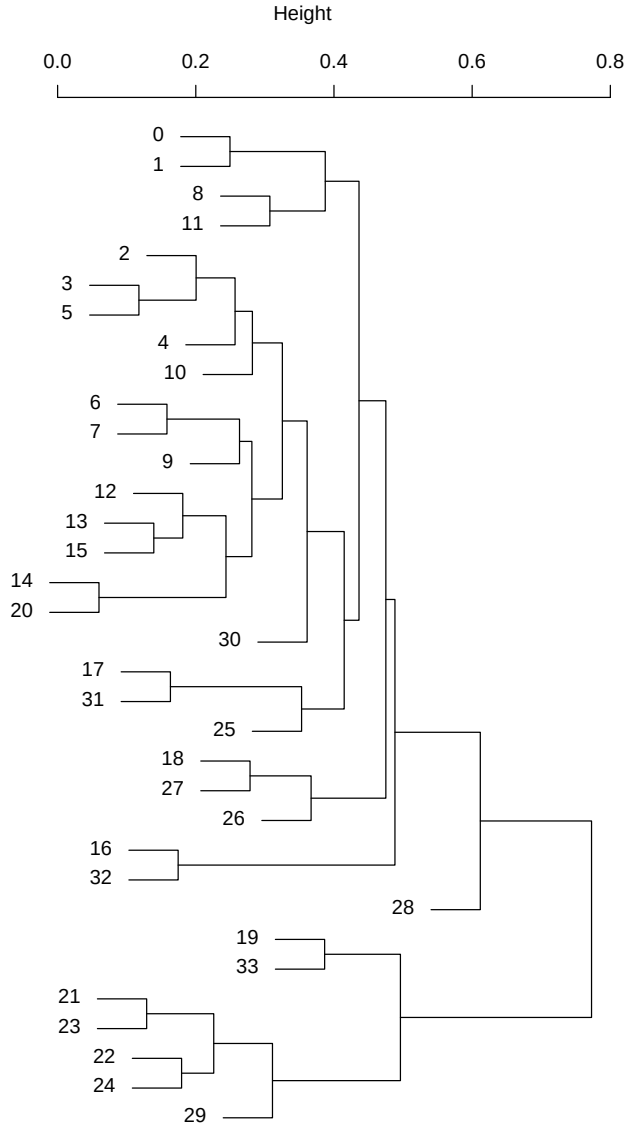

Figure 6.6: The hierarchical clustering of the 34 most frequent items based on their cosine distances.

the the items. By normalizing the scalar product by the euclidean lengths of the vectors corresponding to the covers of the items, we get the cosine of the angle between the vector. The cosine of the angle between two (non-zero) binary vectors is always in the interval $[0, 1]$. Thus, the cosine distance between two items A and B in a transaction database $\mathcal{D}$ is

$$\begin{aligned} d_{\cos}(A, B, \mathcal{D}) &= 1 - \frac{|cover(A, \mathcal{D}) \cap cover(B, \mathcal{D})|}{\sqrt{|cover(A, \mathcal{D})|}\sqrt{|cover(B, \mathcal{D})|}} \\ &= 1 - \frac{supp(AB, \mathcal{D})}{\sqrt{supp(A, \mathcal{D})}\sqrt{supp(B, \mathcal{D})}} \\ &= 1 - \frac{supp(AB, \mathcal{D})/supp(\emptyset, \mathcal{D})}{\sqrt{supp(A, \mathcal{D})}\sqrt{supp(B, \mathcal{D})}/supp(\emptyset, \mathcal{D})} \\ &= 1 - \frac{fr(AB, \mathcal{D})}{\sqrt{fr(A, \mathcal{D})fr(B, \mathcal{D})}}. \end{aligned}$$

The hierarchical clustering of the 34 most frequent items based cosine distances is shown in Figure 6.6. The clustering is very close to the one shown in Figure 6.4. The main difference between these two clusterings is that in the clustering shown in Figure 6.4 the courses 0 and 1 are very different to everything (including each other), whereas the clustering shown in Figure 6.6 grasps the similarity between the courses 0, 1, 8 and 11. □

6.3 Estimating Frequencies from Change Profiles

The change profiles can be used as a basis for condensed representations of frequent itemsets. Furthermore, several known condensed representations can be adapted to change profiles. One interesting approach to condense the change profiles (and thus the underlying pattern collections, too) is to choose a small set of representative change profiles (using, e.g., hierarchical clustering) and replace the original change profiles by the chosen representatives. Then the frequencies of the frequent itemsets can be estimated from the approximate change profiles.

Representing the frequencies of the frequent itemsets by approximate change profiles can be seen as a condensed representation of

the collection of frequent itemsets as the approximate change profiles can (potentially) fit into smaller space than the exact change profiles or even the frequent itemsets. Also, the condensed representations can be applied to further condense the approximate change profiles.

In addition to the fact that the frequencies can be estimated from the approximate change profiles, the change profiles themselves can benefit from the frequency estimation. Namely, the quality of the approximate change profiles can be assessed by evaluating how well the frequencies can be approximated from them.

For the rest of the section we consider only the case where no change profile is missing but the changes are not exact. The methods described in this section can be generalized to handle missing change profiles and missing changes.

Given the approximations of the change profiles for the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets in $\mathcal{D}$, it is possible to estimate the frequencies of the itemsets in $\mathcal{F}(\sigma, \mathcal{D})$ from the approximate change profiles. The estimation can be done in many ways and the quality of each estimation method depends on how the approximations of the change profiles are obtained.

Next we describe an approach based on the estimates given by different paths (in the graph determined by the changes of the change profiles) from the empty itemset $\emptyset$ to the itemset X whose frequency is under estimation. Especially, we concentrate on computing the average frequencies given by the paths from $\emptyset$ to X. The methods are described using simple specializing change profiles, but their generalization to other kinds of change profiles is straightforward.

Without loss of generality, let $X = \{1, \ldots, k\}$. In principle, we could compute the frequency estimate $fr(X)$ of the itemset X in $\mathcal{F}(\sigma, \mathcal{D})$, the average of the frequencies suggested by paths from $\emptyset$ to X. Let Π_k be the collection of all permutations of $\{1, \ldots, k\}$. Then the frequency estimate can be written as

$$fr(X) = \frac{1}{|X|!} \sum_{\pi \in \Pi_k} sch_s^{\emptyset}(\pi(1)) \prod_{i=2}^{k} sch_s^{\pi(i-1)}(\pi(i)). \tag{6.1}$$

The main practical difficulty of this formula is the number of paths: The number of paths from $\emptyset$ to X is equal to the number of

permutations of items in X, i.e., the number of paths from $\emptyset$ to X is $|X|!$. This can be superpolynomial in $|\mathcal{F}(\sigma, \mathcal{D})|$.

Example 6.12 (the number of paths given by simple change profiles is superpolynomial). Let $\mathcal{F}(\sigma, \mathcal{D})$ consist of an itemset X and all of its subitemsets. Then

$$|\mathcal{F}(\sigma, \mathcal{D})| = 2^{|X|}$$

and

$$|X|! = \sqrt{2\pi\,|X|}\left(\frac{|X|}{e}\right)^{|X|}\left(1 + \Theta(|X|^{-1})\right).$$

Hence,

$$\frac{|X|!}{|\mathcal{F}(\sigma, \mathcal{D})|} = \sqrt{2\pi\,|X|}\left(\frac{|X|}{2e}\right)^{|X|}\left(1 + \Theta(|X|^{-1})\right)$$

which is clearly exponential in $|X|$. □

The frequency estimate $fr(X)$ of X as the average over all paths from $\emptyset$ to X can be computed much faster by observing that the frequency of X is the average of the frequencies of the itemsets $X \setminus \{A\}, A \in X$, scaled by the changes $sch_s^{X\setminus\{A\}}(\{A\})$, i.e.,

$$fr(X) = \frac{1}{|X|}\sum_{Y\subset X, |Y|=|X|-1} fr(Y)\, sch_s^Y(X \setminus Y).$$

This observation readily gives a dynamic programming solution described as Algorithm 6.2.

As the frequency estimate has to be computed also for all subsets of X and the frequency estimate of Y can be computed from the frequency estimates of the subsets of Y in time $\mathcal{O}(|Y|)$ the time complexity of Algorithm 6.2 is

$$\mathcal{O}(|X|\,2^{|X|}) = \mathcal{O}(|\mathcal{F}(\sigma, \mathcal{D})| \log |\mathcal{F}(\sigma, \mathcal{D})|).$$

Even this can be too much for a restive data analyst. The estimation can be further speeded up by sampling uniformly from the paths from $\emptyset$ to X as described by Algorithm 6.3.

The time complexity of Algorithm 6.3 is $\mathcal{O}(k\,|X|)$ where k is the number of randomly chosen paths in the estimate. Note that the algorithm can be easily modified to be an any-time algorithm.

Algorithm 6.2 A dynamic programing solution for frequency estimation from (inexact) change profiles.

Input: An itemset X and the simple specializing change profiles (at least) for X and all of its subsets.

Output: The frequency estimate $fr(X)$ of X as described by Equation 6.1.

```
1:  function DP-FROM-SCHS(X, sch_s)
2:      fr(∅) ← 1
3:      for i = 1, ..., |X| do
4:          for all Y ⊆ X, |Y| = i do
5:              fr(Y) = 0
6:              for all Z ⊂ Y, |Z| = |Y| − 1 do
7:                  fr(Y) ← fr(Y) + fr(Z) sch_s^Z(Y \ Z)
8:              end for
9:              fr(Y) ← fr(Y)/|Y|
10:         end for
11:     end for
12: end function
```

Algorithm 6.3 A randomized algorithm for frequency estimation from (inexact) change profiles.

Input: An itemset X, the simple specializing change profiles (at least) for X and all of its subsets, and a positive integer k.

Output: An estimate of the frequency estimate $fr(X)$ of X as described by Equation 6.1.

```
1:  function SAMPLE-FROM-SCHS(X, sch_s, k)
2:      fr(∅) ← 1
3:      fr(X) ← 0
4:      for j = 1, ..., k do
5:          Y ← ∅
6:          for i = 1, ..., |X| − 1 do
7:              A ← RANDOM-ELEMENT(X \ Y)
8:              fr(Y ∪ {A}) ← fr(Y) sch_s^Y({A})
9:              Y ← Y ∪ {A}
10:         end for
11:         fr(X) ← fr(X) + fr(Y) sch_s^Y(X \ Y)
12:     end for
13:     fr(X) ← fr(X)/k
14: end function
```

This would sometimes be useful in interactive data mining and for resource bounded data mining in general.

Algorithm 6.2 and Algorithm 6.3 can be adapted to other kinds of estimates, too. Especially, if upper and lower bounds for the changes $sch_s^Y(A)$ are given for all $Y \subseteq X$ such that $A \in X \setminus Y$, then it is possible to compute the upper and lower bounds for the frequency of X for all itemsets X reachable from $\emptyset$ by changes of the change profiles. Namely, the frequency of the itemset X is at most the minimum of the upper bound estimates and at least the maximum of the lower bound estimates determined by the change paths from $\emptyset$ to X.

6.4 Condensation by Change Profiles

The usefulness of approximate change profiles, the stability of the frequency estimation algorithms proposed in Section 6.3 and the accuracy of the path sampling estimates were evaluated by estimating frequencies from noisified simple specializing change profiles in the transaction databases Internet Usage and IPUMS Census (see Subsection 2.2.1).

In order to study how the estimation methods (i.e., Algorithm 6.2 and Algorithm 6.3) tolerate different kinds of noise, the simple specializing change profiles were noisified in three different ways:

- randomly perturbing the changes of the change profiles by $\pm\epsilon$,
- adding uniform noise from the interval $[-\epsilon, \epsilon]$ to the changes of the change profiles, and
- adding Gaussian noise with zero mean and standard deviation ϵ to the changes of the change profiles.

The changes of the noisified change profiles were truncated to the interval $[0, 1]$ since, by the definition of specializing change profiles (Definition 6.1), the changes in the specializing change profiles must be in the interval $[0, 1]$.

We tested the dependency of the approximation on the number of sample paths by evaluating the absolute difference between the correct and the estimated frequencies for the dynamic programming solution corresponding to the average frequency estimate over all

paths, and the sample solution corresponding to the average frequency estimate over the sampled paths. The experiments were repeated with different number of randomly chosen paths, minimum frequency thresholds and noise levels ϵ.

The results for Internet Usage data with minimum frequency threshold 0.20 are shown in Figures 6.7, 6.9 and 6.11, and for IPUMS Census data with minimum frequency threshold 0.30 are shown in Figures 6.8, 6.10 and 6.12, with noise level $\epsilon = 0.01$. The each of the curves are averages of 1000 random experiments. The results were similar with the other minimum frequency thresholds, too.

The results show that already a quite small number of random paths suffices to give frequency approximations closed to the dynamic programming solution. Furthermore, the average absolute errors achieved by dynamic programming were relatively small, especially as the errors in the changes cumulate multiplicatively as the frequencies are estimated as the paths.

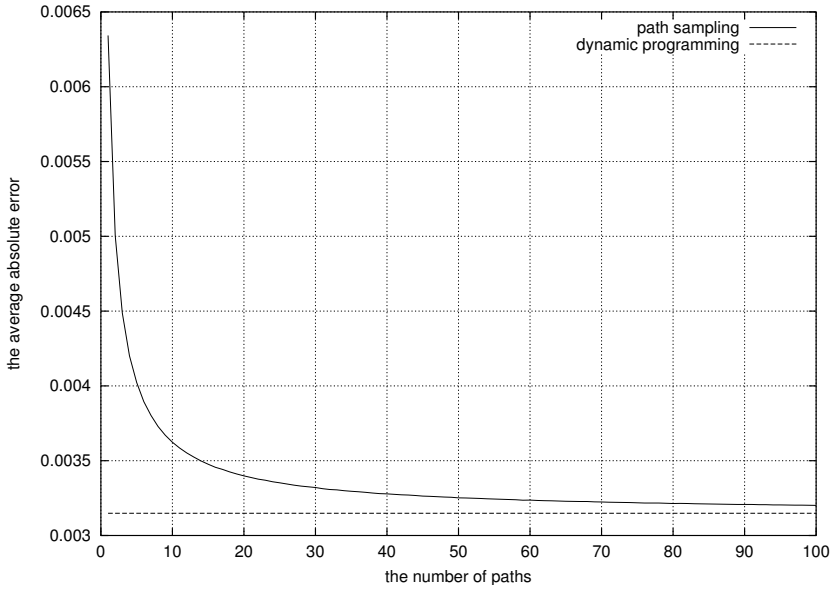

Figure 6.7: Internet Usage data, Gaussian noise.

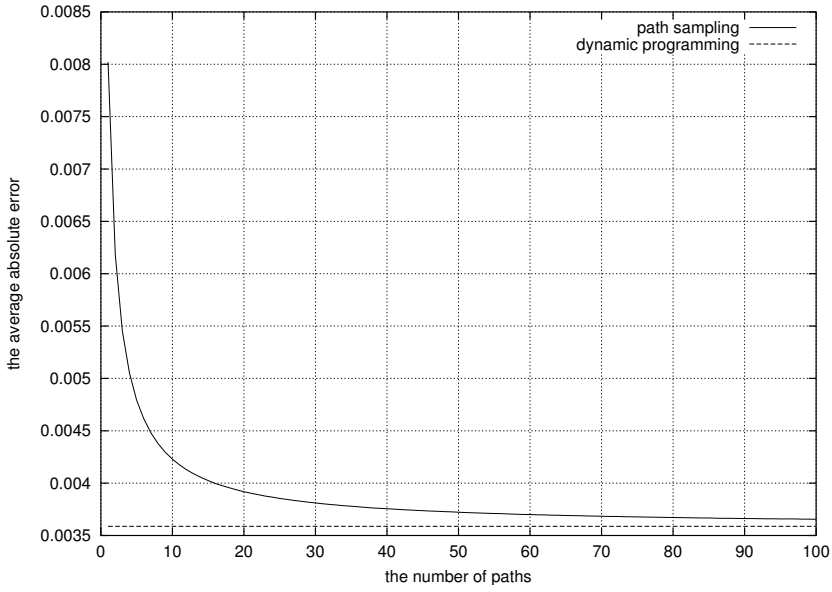

Figure 6.8: IPUMS Census data, Gaussian noise.

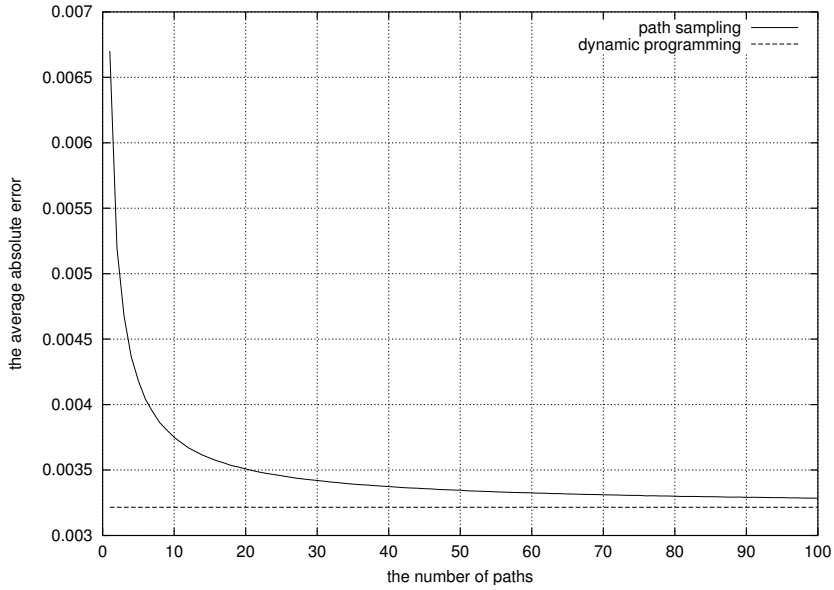

Figure 6.9: Internet Usage data, perturbation.

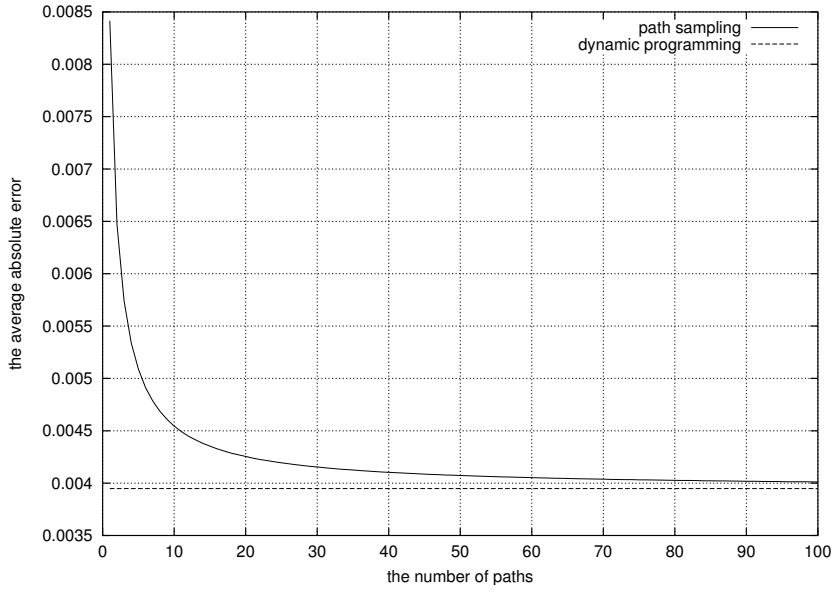

Figure 6.10: IPUMS Census data, perturbation.

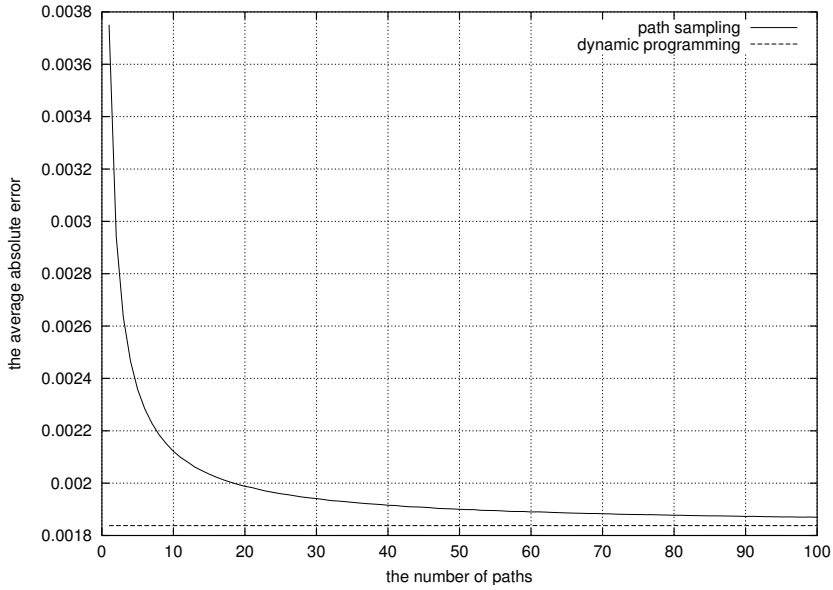

Figure 6.11: Internet Usage data, uniform noise.

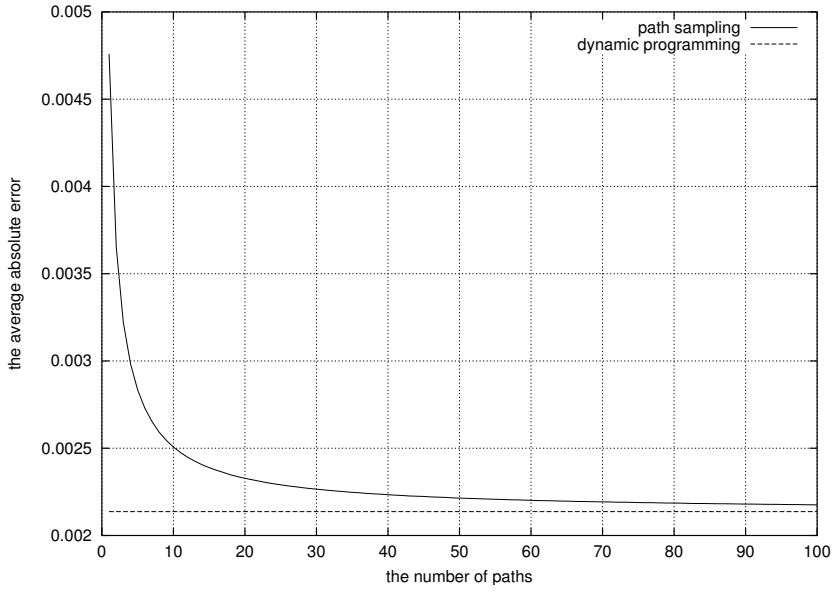

Figure 6.12: IPUMS Census data, uniform noise.

CHAPTER 7

Inverse Pattern Discovery

The problem of discovering interesting patterns from data has been studied very actively in data mining. (See, e.g., Chapter 2 for more details.) For such a well-studied problem, it is natural to study also the inverse version of the problem, i.e., the problem of the inverse pattern discovery. That is, to study the problem of finding a database compatible with a given collection of interesting patterns.

In addition of being an important class of tasks for data mining as a scientific discipline, inverse pattern discovery might also have some practical relevancy.

First, the existence of databases compatible with the given collection of patterns is usually highly desirable since the collection of interesting patterns is often assumed to be a summary of some database. Deciding whether there exists a database compatible with the given collection of interesting patterns (and their quality values) can be considered as a very harsh quality control. An efficient method for answering to that question could have also some practical implications to pattern discovery since several instances are not willing to share their data but might sell some patterns claiming they are interesting in their database. (If interaction with the pattern provider would be allowed, then also, e.g., zero knowledge proofs deciding whether they are interesting or not could be considered [Gol02].)

Second, if the number of compatible databases could be counted, then the pattern provider could evaluate how well the pattern user could detect the correct database from the patterns: without any background information, a randomly chosen database compatible

with the patterns would be the original one with probability $1/k$ where k is the number of databases compatible with the patterns. If there is more background information, however, then the probability of finding the original database can sometimes made higher but still the number of compatible databases is likely to tell about the difficulty of finding the original database based on the patterns. Furthermore, the number of compatible databases can be used as a measure of how well the pattern collection characterizes the database.

Third, the pattern collection can be considered as a collection of queries that should be answered correctly. Thus, the patterns can be used to optimize the database with respect to, e.g., query efficiency or space consumption. The optimization task could be expressed as follows: given the pattern collection, find the smallest database that gives the correct quality values for the patterns.

In this chapter, the computational complexity of inverse pattern discovery is studied in the special case of frequent itemsets. Deciding whether there is a database compatible with the frequent itemsets and their frequencies is shown to be NP-complete although some special cases of the problem can be solved in polynomial time. Furthermore, finding the smallest compatible transaction database for an itemset collection consisting only of two disjoint maximal itemsets and all their subitemsets is shown to be NP-hard.

Obviously inverse frequent itemset mining is just one example of inverse pattern discovery. For example, let us assume that the data is a d-dimensional matrix (i.e., a d-dimensional data cube [GBLP96]) and the summary of the data consists of the sums over each coordinate (i.e., all $d-1$-dimensional sub-cubes). The computational complexity of the inverse pattern discovery for such data and patterns, i.e., the computational complexity of the problem of reconstructing a multidimensional table compatible with the sums, has been studied in the field of discrete tomography [HK99]: the problem is solvable in polynomial time if the matrix is two-dimensional binary matrix [Kub89] and NP-hard otherwise [CD01, GDVW00, IJ94]. In this chapter, however, we shall focus on inverting frequent itemset mining.

This chapter is based on the article "On Inverse Frequent Set Mining" [Mie03e]. Some similar results were independently shown by Toon Calders [Cal04a]. Recently, a heuristic method for the in-

verse frequent itemset mining problem has been proposed [WWWL05].

7.1 Inverting Frequent Itemset Mining

The problem of deducing a transaction database compatible with a given collection frequent itemsets are their supports can be formulated as follows.

Problem 7.1 (inverse frequent itemset mining). Given a downward closed collection $\mathcal{F}$ of itemsets and the support $supp(X)$ for each itemset $X \in \mathcal{F}$, find a transaction database $\mathcal{D}$ compatible with the supports of the collection, i.e., a transaction database $\mathcal{D}$ such that $supp(X, \mathcal{D}) = supp(X, \mathcal{F})$ for all $X \in \mathcal{F}$.

Example 7.1 (inverse frequent itemset mining). The collection $\mathcal{F} = \{\emptyset, A, B, C, AB, BC\}$ with supports

$$\begin{aligned} supp(\emptyset, \mathcal{F}) &= 6, \\ supp(A, \mathcal{F}) &= 4, \\ supp(B, \mathcal{F}) &= 4, \\ supp(C, \mathcal{F}) &= 4, \\ supp(AB, \mathcal{F}) &= 3 \quad \text{and} \\ supp(BC, \mathcal{F}) &= 3 \end{aligned}$$

restrict the collection of transaction databases compatible with these constraints. For example, the following constraints can be deduced from the support constraints:

- The support of the empty itemset tells that the number of transactions in any compatible databases is six.
- There are exactly one transaction with A and without B ($supp(A, \mathcal{F}) - supp(AB, \mathcal{F}) = 1$), one with B and without C ($supp(B, \mathcal{F}) - supp(BC, \mathcal{F}) = 1$), and vice versa ($supp(C, \mathcal{F}) - supp(BC, \mathcal{F}) = 1$).
- The support of AC is at least two since there are at most two transactions that contain B but not A and C.

One transaction database compatible with these constraints is

$$\mathcal{D} = \{\langle 1, ABC\rangle, \langle 2, ABC\rangle, \langle 3, AB\rangle, \langle 4, BC\rangle, \langle 5, A\rangle, \langle 6, C\rangle\}$$

since the supports

$$\begin{aligned} supp(\emptyset, \mathcal{D}) &= 6, \\ supp(A, \mathcal{D}) &= 4, \\ supp(B, \mathcal{D}) &= 4, \\ supp(C, \mathcal{D}) &= 4, \\ supp(AB, \mathcal{D}) &= 3, \\ supp(AC, \mathcal{D}) &= 2, \\ supp(BC, \mathcal{D}) &= 3 \quad \text{and} \\ supp(BC, \mathcal{D}) &= 3 \end{aligned}$$

determined by $\mathcal{D}$ agree with the given supports.

There are also other databases compatible with the constraints. For example,

$$\mathcal{D} = \{\langle 1, ABC\rangle, \langle 2, ABC\rangle, \langle 3, ABC\rangle, \langle 4, A\rangle, \langle 5, B\rangle, \langle 6, C\rangle\}$$

is one such database. □

The reason why we use supports instead of frequencies in the inverse frequent itemset mining problem (Problem 7.1) is that supports are slightly more informative. On one hand, the frequencies of the frequent itemsets $X \in \mathcal{F}(\sigma, \mathcal{D})$ can be computed from their supports since $fr(X, \mathcal{D}) = supp(X, \mathcal{D})/supp(\emptyset, \mathcal{D})$. On the other hand, the supports cannot be computed from the frequencies: the number of transactions in the database, i.e., $supp(\emptyset, \mathcal{D})$ is not revealed by the frequencies of the itemsets.

7.2 Frequent Itemsets and Projections

To determine the complexity of the inverse frequent itemset mining problem (Problem 7.1), let us consider an intermediate representation between the frequent itemsets and the transaction database.

Definition 7.1 (projections of transaction databases). The *projection* of the transaction database $\mathcal{D}$ onto itemset X is a restriction

$$pr(X, \mathcal{D}) = \{\langle i, X \cap Y\rangle : \langle i, Y\rangle \in \mathcal{D}\}$$

of the database $\mathcal{D}$. The collection of projections $pr(X, \mathcal{D})$ onto itemsets $X \in \mathcal{F}$ is denoted by

$$pr(\mathcal{F}, \mathcal{D}) = \{pr(X, \mathcal{D}) : X \in \mathcal{F}\}.$$

Two projections $pr(X, \mathcal{D})$ and $pr(X, \mathcal{D}')$ are considered to be equivalent if and only if $|\mathcal{D}| = |\mathcal{D}'|$ and there is a bijective mapping π from $tid(\mathcal{D})$ to $tid(\mathcal{D}')$ such that for each $\langle i, Y\rangle \in \mathcal{D}$ there is $\langle \pi(i), Y'\rangle \in \mathcal{D}'$ with $X \cap Y = X \cap Y'$. (That is, the mapping π is a permutation since we can assume that $tid(\mathcal{D}) = tid(\mathcal{D}') = \{1, \ldots, |\mathcal{D}|\}$; see Definition 2.2.)

The projections of transaction databases have many desirable similarities to itemsets. For example, neglecting the transaction identifiers, the projections of the database $\mathcal{D}$ onto maximal σ-frequent itemsets contain the same information than the σ-frequent itemsets and their supports.

Theorem 7.1. *The frequent itemsets in $\mathcal{F}(\sigma, \mathcal{D})$ and their supports in $\mathcal{D}$ can be computed from the projections $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$ and the projections equivalent to $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$ can be computed from the frequent itemsets in $\mathcal{F}(\sigma, \mathcal{D})$ and their supports in $\mathcal{D}$.*

Proof. For each $X \in \mathcal{F}(\sigma, \mathcal{D})$ and each $Y \supseteq X$ we have

$$\begin{aligned} supp(X, \mathcal{D}) &= |\{\langle i, Z\rangle \in \mathcal{D} : X \subseteq Z\}| \\ &= |\langle i, Y \cap Z\rangle \in \mathcal{D} : X \subseteq (Y \cap Z)| \\ &= supp(X, pr(Y, \mathcal{D})). \end{aligned}$$

By definition, each σ-frequent itemset $X \in \mathcal{F}(\sigma, \mathcal{D})$ is contained in some maximal σ-frequent itemset $Y \in \mathcal{FM}(\sigma, \mathcal{D})$. Furthermore, no σ-infrequent itemset is contained in any of the maximal σ-frequent itemsets in $\mathcal{D}$. Thus, the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets and their supports in $\mathcal{D}$ can be computed from the collection $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$ of projections of the transaction database $\mathcal{D}$ onto the maximal σ-frequent itemsets in $\mathcal{D}$.

The projections equivalent to $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$ can be computed from the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets and their supports in $\mathcal{D}$ by Algorithm 7.1. The running time of the algorithm is polynomial in $|\mathcal{F}(\sigma, \mathcal{D})|$, $|\mathcal{I}|$ and $|\mathcal{D}| = supp(\emptyset, \mathcal{D})$.

The running time can be further improved if the transaction database $\mathcal{D}'$ has a primitive for inserting k transactions consisting of an itemset X into $\mathcal{D}'$ in time polynomial in $|\mathcal{D}'|$ and in $|X|$ but not in k at all. Namely, then the running time of Algorithm 7.1 can be expressed as a polynomial of $|\mathcal{F}(\sigma, \mathcal{D})|$ and $|\mathcal{I}|$, i.e., not depending on the actual number of transactions in the transaction database $\mathcal{D}$.

The efficient insertion of k transactions with the itemset X into $\mathcal{D}$ can be implemented, e.g., by "run-length encoding" the database, i.e., by describing the transactions $\langle i, X\rangle, \ldots, \langle i + k - 1, X\rangle$ by the triple $\langle i, k, X\rangle$. Then the insertion of k transactions with the itemset X to $\mathcal{D}'$ can be implemented by inserting the tuple $\langle |\mathcal{D}| + 1, k, X\rangle$ to $\mathcal{D}'$. □

Theorem 7.1 also implies at if $\mathcal{F}(\sigma, \mathcal{D}) = 2^{\mathcal{I}}$, then the whole transaction database $\mathcal{D}$ (although without the correct transaction identifiers) can be reconstructed from the collection $\mathcal{F}(\sigma, \mathcal{D})$ and their supports in $\mathcal{D}$ in time polynomial in $|\mathcal{F}(\sigma, \mathcal{D})|$ and $|\mathcal{D}|$ since the collection $\mathcal{FM}(\sigma, \mathcal{D})$ consists only of the itemset $\mathcal{I}$. Furthermore, the supports of frequent itemsets can determine (implicitly) also supports of some infrequent itemsets [Cal04b, CG02].

Let us denote the projections determined by the downward closed itemset collection $\mathcal{F}$ by $pr(\mathcal{M}, \mathcal{F})$. The number of different itemsets in the transactions of $pr(\mathcal{M}, \mathcal{F})$ can be considerably smaller than the number itemsets in $\mathcal{F}$. Thus, each projection $pr(X, \mathcal{D})$ of $\mathcal{D}$ onto $X \in \mathcal{FM}(\sigma, \mathcal{D})$ represented as a list of tuples $\langle count(Y, \mathcal{D}), Y\rangle, Y \subseteq X$, can be used as a condensed representation of the collection $\mathcal{F}(\sigma, \mathcal{D})$ and their supports. Such projections provide sometimes very small representations compared to $\mathcal{F}(\sigma, \mathcal{D})$ [Mie03c].

As projections constructed from the collection $\mathcal{F}(\sigma, \mathcal{D})$ of the σ-frequent itemsets and their supports in $\mathcal{D}$ are (at least seemingly) closer to the original transaction database than the collection $\mathcal{F}(\sigma, \mathcal{D})$ and their supports, the projections could be useful to make the inverse frequent itemset mining problem more comprehensible by an equivalent formulation of the problem.

Algorithm 7.1 An algorithm to compute projections equivalent to $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$ from $\mathcal{F}(\sigma, \mathcal{D})$ and their supports.

Input: The collection $\mathcal{F}(\sigma, \mathcal{D})$ σ-frequent itemsets in a transaction database $\mathcal{D}$ and their supports.

Output: The projections $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{F}(\sigma, \mathcal{D}))$ equivalent to projections $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$.

1: **function** TO-PROJECTIONS($\mathcal{F}(\sigma, \mathcal{D}), supp$)
2: $\quad\mathcal{FM}(\sigma, \mathcal{D}) = \{X \in \mathcal{F}(\sigma, \mathcal{D}) : Y \supset X \Rightarrow Y \notin \mathcal{F}(\sigma, \mathcal{D})\}$
3: $\quad$**for all** $X \in \mathcal{FM}(\sigma, \mathcal{D})$ **do**
4: $\quad\quad\mathcal{D}' \leftarrow \emptyset$
5: $\quad\quad\mathcal{M} \leftarrow \{X\}$
6: $\quad\quad\mathcal{F} \leftarrow \{Y \in \mathcal{F}(\sigma, \mathcal{D}) : Y \subseteq X\}$
7: $\quad\quad$**for all** $Y \in \mathcal{F}$ **do**
8: $\quad\quad\quad supp(Y, \mathcal{F}) \leftarrow supp(Y, \mathcal{D})$
9: $\quad\quad$**end for**
10: $\quad\quad$**while** $\mathcal{M} \neq \emptyset$ **do**
11: $\quad\quad\quad\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{M}$
12: $\quad\quad\quad$**for all** $Y \in \mathcal{M}$ **do**
13: $\quad\quad\quad\quad\mathcal{D}' \leftarrow \mathcal{D}' \cup \{\langle |\mathcal{D}'| + 1, Y\rangle, \ldots, \langle |\mathcal{D}'| + supp(Y, \mathcal{F}), Y\rangle\}$
14: $\quad\quad\quad\quad$**for all** $Z \in \mathcal{F}, Z \subset Y$ **do**
15: $\quad\quad\quad\quad\quad supp(Z, \mathcal{F}) \leftarrow supp(Z, \mathcal{F}) - supp(Y, \mathcal{F})$
16: $\quad\quad\quad\quad\quad$**if** $supp(Z, \mathcal{F}) = 0$ **then**
17: $\quad\quad\quad\quad\quad\quad\mathcal{F} \leftarrow \mathcal{F} \setminus \{Z\}$
18: $\quad\quad\quad\quad\quad$**end if**
19: $\quad\quad\quad\quad$**end for**
20: $\quad\quad\quad$**end for**
21: $\quad\quad\quad\mathcal{M} \leftarrow \{Y \in \mathcal{F} : Y \subset Z \Rightarrow Z \notin \mathcal{F}\}$
22: $\quad\quad$**end while**
23: $\quad\quad pr(X, \mathcal{F}(\sigma, \mathcal{D})) \leftarrow pr(X, \mathcal{D}')$
24: $\quad$**end for**
25: $\quad$**return** $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{F}(\sigma, \mathcal{D}))$
26: **end function**

Problem 7.2 (database reconstruction from projections). Given a collection $pr(\mathcal{M}, \mathcal{F})$ of projections onto maximal itemsets, find a transaction database $\mathcal{D}$ such that $pr(\mathcal{M}, \mathcal{F}) = pr(\mathcal{M}, \mathcal{D})$.

There are, however, collections of projections that cannot be realized as downward closed itemset collections. We should be able to ensure in time polynomial in the sum of the cardinalities of transactions in the projections that the collection of projections can be realized as a downward closed itemset collection with some supports. Fortunately, there are simple conditions that are necessary and sufficient to ensure that there is a downward closed itemset collection compatible with a given collection of projections.

Theorem 7.2. *The projections* $pr(X_1, \mathcal{F}_1), \ldots, pr(X_m, \mathcal{F}_m)$ *have the compatible collection* $\mathcal{F}$ *of itemsets, i.e., a collection* $\mathcal{F}$ *such that*

$$supp(Y, pr(X_i, \mathcal{F}_i)) = supp(Y, \mathcal{F})$$

for all $Y \subseteq X_i, 1 \leq i \leq m$*, if and only if*

$$pr(X_i \cap X_j, \mathcal{F}_i) = pr(X_i \cap X_j, \mathcal{F}_j)$$

for all $1 \leq i, j \leq m$*.*

Proof. If there is a downward closed itemset collection $\mathcal{F}$ such that

$$supp(Y, pr(X_i, \mathcal{F}_i)) = supp(Y, \mathcal{F})$$

for all $Y \subseteq X_i, 1 \leq i \leq m$, then

$$pr(X_i \cap X_j, \mathcal{F}_i) = pr(X_i \cap X_j, \mathcal{F}_j)$$

for all $1 \leq i, j \leq m$. Otherwise $pr(X_i \cap X_j, \mathcal{F}_i)$ and $pr(X_i \cap X_j, \mathcal{F}_j)$ would determine different supports for some itemset $Y \subseteq X_i \cap X_j$ where $1 \leq i, j \leq m$.

If

$$pr(X_i \cap X_j, \mathcal{F}_i) = pr(X_i \cap X_j, \mathcal{F}_j)$$

for all $1 \leq i, j \leq m$, then

$$supp(Y, pr(X_i \cap X_j, \mathcal{F}_i)) = supp(Y, pr(X_i \cap X_j, \mathcal{F}_j))$$

for all itemsets $Y \subseteq X_i \cap X_j$ where $1 \leq i, j \leq m$. □

The number of transactions in the transaction database $\mathcal{D}$ can be exponential in the number of frequent itemsets (and thus also in the sum of the cardinalities of the frequent itemsets).

Example 7.2 (a transaction database being exponentially larger than the frequent itemset collection). Let the itemset collection consist of just one itemset $\emptyset$ with support exponential in $|\mathcal{I}|$. Then the number of transactions in $\mathcal{D}$ is exponential in $|\mathcal{I}|$. □

This fact does not have to be considered as a drawback since most of the results shown in this chapter are hardness results. Furthermore, it is reasonable to assume that if one is trying to reconstruct a transaction database then the number of transaction in the database is not considered to be unfeasibly large.

7.3 The Computational Complexity of the Problem

In this section we show that Problem 7.2 is difficult in general but some of its special cases can be solved in polynomial time and even in logarithmic space. Our first hardness result shows that Problem 7.2 is NP-hard in general. The hardness is shown by a reduction from the graph 3-colorability problem:

Problem 7.3 (graph 3-colorability [GJ79]). Given a graph $G = \langle V, E \rangle$, decide whether there is a good 3-coloring, i.e., a labeling $label : V \to \{r, g, b\}$ such that $label(u) \neq label(v)$ for all $\{u, v\} \in E$.

Theorem 7.3. *The problem of deciding whether there is a transaction database $\mathcal{D}$ compatible with the projections $pr(\mathcal{M}, \mathcal{F})$ (i.e., the decision version of Problem 7.2) is* NP*-complete even when the compatible transaction databases consist of only six transactions.*

Proof. The problem is clearly in NP since it can be verified in time polynomial in the sizes of $pr(\mathcal{M}, \mathcal{F})$ and $\mathcal{D}$ whether a certain transaction database $\mathcal{D}$ is compatible with projections $pr(\mathcal{M}, \mathcal{F})$ simply by computing the projections $pr(\mathcal{M}, \mathcal{D})$.

We show the NP-hardness of Problem 7.2 by a reduction from an instance $G = \langle V, E \rangle$ of the graph 3-colorability problem (Problem 7.3) to projections $pr(\mathcal{M}, \mathcal{F})$ in $pr(\mathcal{M}, \mathcal{F})$ are compatible with

the projections $pr(\mathcal{FM}(\sigma, \mathcal{D}), \mathcal{D})$ of some transaction database $\mathcal{D}$ if and only if G is 3-colorable.

Let the set $\mathcal{I}$ of items be $\{r_v, g_v, b_v : v \in V\}$. The projections are constructed as follows. For each edge $\{u, v\} \in E$ we define a projection

$$\begin{aligned} pr(\{r_u, g_u, b_u, r_v, g_v, b_v\}, \mathcal{F}) \quad = \quad & \{\langle 1, \{r_u, g_v\}\rangle, \langle 2, \{r_u, b_v\}\rangle, \\ & \langle 3, \{g_u, r_v\}\rangle, \langle 4, \{g_u, b_v\}\rangle, \\ & \langle 5, \{b_u, r_v\}\rangle, \langle 6, \{b_u, g_v\}\rangle\}. \end{aligned}$$

If the graph $G = \langle V, E\rangle$ is not 3-colorable then there is no transaction database $\mathcal{D}$ compatible with the projections: for every 3-coloring of G, there is an edge $\{u, v\} \in E$ with $label(u) = label(v)$ but none of the pairs $\{r_u, r_v\}$, $\{g_u, g_v\}$, and $\{b_u, b_v\}$ appear in the projection $pr(\{r_u, g_u, b_u, r_v, g_v, b_v\}, \mathcal{F})$. Thus there is not even a partial solution of one transaction compatible the projections.

If the graph G is 3-colorable then there is a transaction database $\mathcal{D}$ that is compatible with the projections: the six transactions in the database $\mathcal{D}$ are the six permutations of a 3-coloring *label* such that $label(u) \neq label(v)$ for all $\{u, v\} \in E$. □

As mentioned in the beginning of the chapter, it would be desirable to be able to estimate how many compatible databases there exist. The proof of Theorem 7.3 can also be adapted to give the hardness result for the counting version of Problem 7.2. (See [Pap95] for more details on counting complexity.)

Theorem 7.4. *The problem of counting the number of transaction databases $\mathcal{D}$ compatible with the projections $pr(\mathcal{M}, \mathcal{F})$ is #P-complete.*

Proof. The problem is in $\#P$ since its decision version is in NP. Using the reduction described in the proof of Theorem 7.3, the number of good 3-colorings could be counted: the number of good 3-colorings is $1/6! = 1/720$ times the number of transaction databases compatible with the projections corresponding to the given graph G. As counting the number of good 3-colorings is $\#P$-hard [GJ79], so is counting the number of compatible databases. □

Although the database reconstruction problem is NP-complete in general, there are some special cases that can be solved in poly-

nomial time. In one of the most simplest such cases the instance consists of only two projections (with arbitrary number of items).

Theorem 7.5. *It can be decided in polynomial time whether there is a transaction database $\mathcal{D}$ that is compatible with given projections $pr(X_1, \mathcal{F}_1)$ and $pr(X_2, \mathcal{F}_2)$. Furthermore, the number of compatible transaction databases $\mathcal{D}$ can be computed in polynomial time.*

Proof. By definition, the projection $pr(X_1, \mathcal{F}_1)$ is compatible with a transaction database $\mathcal{D}$ if and only if $pr(X_1, \mathcal{F}_1) = pr(X_1, \mathcal{D})$ and the projection $pr(X_2, \mathcal{F}_2)$ is compatible with $\mathcal{D}$ if and only if $pr(X_2, \mathcal{F}_2) = pr(X_1, \mathcal{D})$. The database $\mathcal{D}$ compatible with both projections if and only if

$$pr(X_1 \cap X_2, \mathcal{F}_1) = pr(X_1 \cap X_2, \mathcal{D}) = pr(X_1 \cap X_2, \mathcal{F}_2),$$

$$pr(X_1 \setminus X_2, \mathcal{F}_1) = pr(X_1 \setminus X_2, \mathcal{D})$$

and

$$pr(X_2 \setminus X_1, \mathcal{F}_2) = pr(X_2 \setminus X_1, \mathcal{D}).$$

A transaction database $\mathcal{D}$ compatible with the two projections $pr(X_1, \mathcal{F}_1)$ and $pr(X_2, \mathcal{F}_2)$ can be found by sorting the transactions in the projections $pr(X_1, \mathcal{F}_1)$ and $pr(X_2, \mathcal{F}_2)$ with respect to the itemsets in $pr(X_1 \cap X_2, \mathcal{F}_1)$ and $pr(X_1 \cap X_2, \mathcal{F}_2)$, respectively. This can be implemented to run in time $\mathcal{O}(|X_1 \cap X_2|\,|\mathcal{D}|)$ [Knu98]. This method for constructing the compatible database is shown as Algorithm 7.2. The running time of the algorithm is linear in the size of the input, i.e., in the sum of the cardinalities of the transactions in the projections.

The number of transaction databases compatible with the projections $pr(X_1, \mathcal{D})$ and $pr(X_2, \mathcal{D})$ of a given transaction database $\mathcal{D}$ can be computed from the counts $count(X, pr(X_1 \cap X_2, \mathcal{D}))$, $count(Y_1, pr(X_1, \mathcal{D}))$ and $count(Y_2, pr(X_2, \mathcal{D}))$ for all X, Y_1 and Y_2 such that $X = Y_1 \cap X_2 = Y_2 \cap X_1$, $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$, $count(Y_1, pr(X_1, \mathcal{D})) > 0$ and $count(Y_2, pr(X_2, \mathcal{D})) > 0$.

The collection

$$\mathcal{S} = \{X \subseteq X_1 \cap X_2 : count(X, pr(X_1 \cap X_2, \mathcal{D})) > 0\}$$

partitions the transactions in $pr(X_1, \mathcal{D})$ and $pr(X_2, \mathcal{D})$ into equivalence classes of transactions with the same projections to $X_1 \cap X_2$.

Algorithm 7.2 An algorithm for constructing a transaction database $\mathcal{D}$ compatible with projections $pr(X_1, \mathcal{F}_1)$ and $pr(X_2, \mathcal{F}_2)$.

Input: Projections $pr(X_1, \mathcal{F}_1)$ and $pr(X_2, \mathcal{F}_2)$.

Output: A transaction database $\mathcal{D}$ compatible with $pr(X_1, \mathcal{F}_1)$ and $pr(X_2, \mathcal{F}_2)$, or $\emptyset$ if such a database does not exist.

```
1: function FROM-TWO-TO-ONE(pr(X_1, F_1), pr(X_2, F_2))
2:     P_1 ← ∅
3:     for all ⟨i, Y⟩ ∈ pr(X_1, F_1) do
4:         X ← Y ∩ X_2
5:         P_1 ← P_1 ∪ {X}
6:         S^1_X ← S^1_X ∪ {⟨i, Y⟩}
7:     end for
8:     P_2 ← ∅
9:     for all ⟨j, Z⟩ ∈ pr(X_2, F_2) do
10:        X ← Z ∩ X_1
11:        P_2 ← P_2 ∪ {X}
12:        S^2_Y ← S^2_X ∪ {⟨j, Z⟩}
13:    end for
14:    if P_1 ≠ P_2 then
15:        return ∅
16:    end if
17:    D ← ∅
18:    for all X ∈ P_1 do
19:        if |S^1_X| ≠ |S^2_X| then
20:            return ∅
21:        end if
22:        while S^1_X ≠ ∅ do
23:            Choose ⟨i, Y⟩ ∈ S^1_X and ⟨j, Z⟩ ∈ S^2_X arbitrarily.
24:            D ← D ∪ ⟨|D| + 1, Y ∪ Z⟩
25:            S^1_X ← S^1_X \ {⟨i, Y⟩}
26:            S^2_X ← S^2_X \ {⟨j, Z⟩}
27:        end while
28:    end for
29:    return D
30: end function
```

The partition can be further refined by the collections

$$\mathcal{S}_X^1 = \{Y_1 \subseteq X_1 : Y_1 \cap X_2 = X, count(Y_1, pr(X_1, \mathcal{D})) > 0\}$$

and

$$\mathcal{S}_X^2 = \{Y_2 \subseteq X_2 : Y_2 \cap X_2 = X, count(Y_2, pr(X_2, \mathcal{D})) > 0\}.$$

Using these collections, the number of compatible databases can be computed as follows. The transaction identifiers can be partitioned to classes $X \in \mathcal{S}$ in

$$c = \frac{|\mathcal{D}|!}{\prod_{X \in \mathcal{S}} count(X, pr(X_1 \cap X_2, \mathcal{D}))!}$$

ways. In each class $X \in \mathcal{S}$, the transaction identifiers can be further partitioned into classes $Y \in \mathcal{S}_X^1$ in

$$a_X = \frac{count(X, pr(X_1 \cap X_2, \mathcal{D}))!}{\prod_{Y_1 \in \mathcal{S}_X^1} count(Y_1, pr(X_1, \mathcal{D}))!}$$

ways. Now we have counted the number of different projections $pr(X_1, \mathcal{D})$. The number of different databases that can be obtained by merging the transactions in $pr(X_2, \mathcal{D})$ to the transactions of $pr(X_1, \mathcal{D})$ using the transaction identifiers of $pr(X_1, \mathcal{D})$ is

$$b_X = \frac{count(X, pr(X_1 \cap X_2, \mathcal{D}))!}{\prod_{Y_2 \in \mathcal{S}_X^2} count(Y_2, pr(X_2, \mathcal{D}))!}.$$

Thus, the total number of transaction databases compatible with $pr(X_1, \mathcal{D})$ and $pr(X_2, \mathcal{D})$ is $c \prod_{X \in \mathcal{S}} a_X b_X$. □

The practical relevancy of this positive result (Theorem 7.5) depends on how much the domains X_1 and X_2 overlap. If $|X_1 \cap X_2|$ is very small but $|X_1 \cup X_2|$ is large then there is a great danger that there are several compatible transaction databases. Fortunately, in the case of two projections we are able to efficiently count the number of compatible databases and thus to evaluate the usefulness of the found database.

In the simplest case of the database reconstruction problem all projections $pr(X_1, \mathcal{F}_1), \ldots, pr(X_m, \mathcal{F}_m)$ are disjoint since in that case any database with projections $pr(X_1, \mathcal{F}_1), \ldots, pr(X_m, \mathcal{F}_m)$ is

compatible one. Unfortunately this also means that the number compatible databases is very large. Thus, one should probably require something more than mere compatibility.

One natural restriction, applying the Occam's razor, is to search for the compatible database with the smallest number of transactions with different itemsets. This kind of database is (in some sense) the simplest hypothesis based on the downward closed itemset collection. This can be beneficial for both analyzing the data and actioning using the database.

Unfortunately, it can be shown that finding the transaction database with the smallest number of different transactions is NP-hard for already two disjoint projections. We show the NP-hardness by a reduction from 3-partition problem:

Problem 7.4 (3-partition [GJ79]). Given a set A of $3l$ elements, a bound $B \in \mathbb{N}$, and a size $s(a) \in \mathbb{N}$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = lB$, decide whether or not A can be partitioned into l disjoint sets $A_1, \ldots, A_l$ such that for each $\sum_{a \in A_i} = B$ for all $1 \leq i \leq l$.

Theorem 7.6. *It is* NP*-hard to find a transaction database consisting of the smallest number of different transactions and being compatible with the projections $pr(X_1, \mathcal{F})$ and $pr(X_2, \mathcal{F})$ such that $X_1 \cap X_2 = \emptyset$.*

Proof. We show the NP-hardness of the problem by reduction from the 3-partition problem (Problem 7.4).

As Problem 7.4 is known to be strongly NP-complete, we can assume that the sizes $s(a)$ of all elements $a \in A$ are bounded above by polynomial in l.

The instance $\langle A, B, s \rangle$ of 3-partition can be encoded as two projections as follows. Without loss of generality, let the elements of A be $1, \ldots, 3l$. Then

$$X_1 = \{1, \ldots, \lceil 3 \log l \rceil\}$$

and

$$X_2 = \{\lceil \log 3l \rceil + 1, \ldots, \lceil \log 3l \rceil + \lceil \log l \rceil\}.$$

Again, let us denote the binary coding of $x \in \mathbb{N}$ as a set consisting the positions of ones in the binary code by $bin(x)$. Then projection

$pr(X_1, \mathcal{F})$ consists of $s(a)$ transactions consisting of the itemset $bin(a) \subseteq X_1, a \in A$. Projection $pr(X_2, \mathcal{F})$ consists of B transactions consisting of the itemset $bin(b) + \lceil \log 3l \rceil \subseteq X_2, b \in \{1, \ldots, l\}$.

Clearly there is a 3-partition for $\langle A, B, s \rangle$ if and only if there is a database $\mathcal{D}$ with $3l$ different transactions that is compatible with projections $pr(X_1, \mathcal{F})$ and $pr(X_2, \mathcal{F})$. □

Finally, let us note that if the number of items is fixed, then a compatible transaction database can be found in time polynomial in the number of transactions in the projections: Finding a transaction database compatible with the projections can be formulated as a linear integer programming task where the variables are the possible different itemsets in the transactions. The number of possible different itemsets is $2^{|\mathcal{I}|}$. The linear integer programming tasks with a fixed number of variables can be solved in time polynomial in the size of the linear equations [LJ83].

CHAPTER 8

Conclusions

Pattern discovery is an important subfield of data mining that attempts to discover interesting (or high-quality) patterns from data. There are several efficient techniques to discover such patterns with respect to different interestingness measures. Merely discovering the patterns efficiently is rarely the ultimate goal, but the patterns are discovered for some purpose. One important use of patterns is to summarize data, since the pattern collections together with the quality values of the patterns can be considered a summaries of the data.

In this dissertation we have studied how the pattern collections could be summarized. Our approach has been five-fold.

First, we studied how to cast views to pattern collections by simplifying the quality values of the patterns. In particular, we gave efficient algorithms for optimally discretizing the quality values. Furthermore, we described how the discretizations can be used in conjunction with pruning of redundant patterns to simplify the pattern collections.

Second, continuing with the theme of simplifying pattern collections, we considered the trade-offs between the understandability and the accuracy of the pattern collections and their quality values. As a solution that supports exploratory data analysis, we proposed the pattern orderings. A pattern ordering of a pattern collection lists the patterns in such an order that each pattern improves our estimate about the whole pattern collection as much as possible (with respect to given loss function and estimation method). Furthermore, we showed that under certain reasonable assumptions each

length-k prefix of the pattern ordering provides a k-subcollection of patterns that is almost as good description of the whole pattern collection as the best k-subcollection. We illustrated the applicability of pattern orderings in approximating pattern collections and data.

Third, we examined how the structural properties (especially partial orders) of the pattern collections can be exploited to obtain clusterings of the patterns and more concise descriptions of the pattern collections. The same techniques can be used to simplify also transaction databases.

Fourth, we proposed a generalization of association rules: change profiles. A change profile of a pattern describes how the quality value of the pattern has to be changed to obtain the quality values of neighboring patterns. The change profiles can be used to compare patterns with each other: patterns can be considered similar, if their change profiles are similar. We studied the computational complexity of clustering patterns based on their change profiles. The problem turned out to be quite difficult if some approximation quality requirements are given. This does not rule out the use of heuristic clustering methods or hierarchical clustering. We illustrated the hierarchical clusterings of change profiles using real data. In addition to clustering change profiles, we considered frequency estimation from approximate change profiles that could be used as building blocks of condensed representations of pattern collections. We provided efficient algorithms for the frequency estimation from the change profiles and evaluated empirically the noise tolerance of the methods.

Fifth, we studied the problem of inverse pattern discovery, i.e., the problem of constructing data sets that could have induced the given patterns and their quality values. More specifically, we studied the computational complexity of inverse frequent itemset mining. We showed that the problem of finding a transaction database compatible with a given collection of frequent itemsets and their supports is NP-hard in general, but some of its special cases are solvable in polynomial time.

Although the problems studied in this dissertation are different, they have also many similarities. Frequency simplifications, pattern orderings, pattern chains and change profiles are all techniques for summarizing pattern collections. Frequency simplifications and

pattern orderings provide primarily approximations of the pattern collections, whereas pattern chains and change profiles describe the pattern collection by slightly more complex patterns obtained by combining the patterns of the underlying pattern collection.

There are also many other ways to group the techniques. For example, the following similarities and dissimilarities can be observed:

- Pattern orderings, pattern chains and change profiles make use of the relationships between the patterns directly, whereas frequency simplifications do not depend on the actual patterns.
- Frequency simplifications, pattern orderings and change profiles can be used to obtain an approximate description of the pattern collection, whereas pattern chains provide an exact description.
- Frequency simplifications, pattern orderings and pattern chains describe the quality values of the patterns, whereas change profiles describe the changes in the quality values.
- Frequency simplifications, pattern chains and change profiles can be used to cluster the patterns, whereas the interpretation of pattern orderings as clusterings is not so straightforward.

Also inverse pattern discovery has similarities with the other problems, as all the problems are related to the problem of evaluating the quality of the pattern collection. Furthermore, all problems are closely related to the two high-level themes of the dissertation, namely post-processing and condensed representations of pattern collections.

As future work, exploring the possibilities and limitations of condensed representations of pattern collections is likely to be continued. One especially interesting question is how the pattern collections should actually be represented. Some suggestions are provided in [Mie04c, Mie05a, Mie05b]. Also, measuring the complexity of the data and its relationships to condensed representations seems to be an important and promising research topic.

As data mining is inherently exploratory process involving often huge data sets, a proper data management infrastructure seems to be necessary. A promising model for that, and for data mining as

whole, is offered by inductive databases [MLK04]. There are many interesting questions related to inductive databases. For example, it is not completely clear what inductive databases are or what they should be [Mie04a].

Recently also the privacy issues of data mining have been recognized to be of high importance [Pin02, VBF+04]. There are two very important topics in privacy preserving data mining. First, sometimes no one has access to the whole data but still the data owners are interested in mining the data. There has been already many proposals for secure computation of many data mining results, for example frequent itemsets [ESAG02, FNP04, GLLM05, VC02]. Second, in addition to computing the data mining results securely, it is often very important that the data mining results themselves are secure, i.e., that they do not leak any sensitive information about the data [FJ02, Mie04b, OZS04, SVC01, VEEB+04].

Another important problem related to inductive databases is finding the underlying general principles of pattern discovery [MT97]. There are many pattern discovery algorithms, but it is still largely open what are the essential differences between the methods and how to choose the technique for some particular pattern discovery task. Some preliminary evaluation of the techniques in the case of frequent itemset mining has recently been done [GZ03], but the issues of a general theory of pattern discovery are still largely open.

References

[AAP01] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing*, 61:350–371, 2001.

[ACK+99] G. Ausiello, P. Crescenzi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag, 1999.

[AGM04] Foto N. Afrati, Aristides Gionis, and Heikki Mannila. Approximating a collection of frequent sets. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 12–19. ACM, 2004.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo.

Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT Press, 1996.

[AS04] Setsuo Arikawa and Einoshin Suzuki, editors. *Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2–5, 2004, Proceedings*, volume 3245 of *Lecture Notes in Computer Science*. Springer, 2004.

[BB00] Jean-François Boulicaut and Arthur Bykowski. Frequent closures as a concise representation for binary data mining. In Takao Terano, Huan Liu, and Arbee L. P. Chen, editors, *Knowledge Discovery and Data Mining, Current Issues and New Applications, 4th Pacific-Asia Conference, PAKDD 2000, Kyoto, Japan, April 18-20, 2000, Proceedings*, volume 1805 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2000.

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In Lucian Popa, editor, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 1–16. ACM, 2002.

[BBR03] Jean-François Boulicaut, Arthur Bykowski, and Christophe Rigotti. Free-sets: a condensed representation of Boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003.

[BCG01] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In Alex Buchmann and Dimitrios Georgakopoulos, editors, *Proceedings*

of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany, pages 443–452. IEEE Computer Society, 2001.

[BD03] Jean-François Boulicaut and Sašo Džeroski, editors. *2nd International Workshop on Knowledge Discovery in Inductive Databases*, 2003.

[BGKM02] Endre Boros, Vladimir Gurvich, Leonid Khachiyan, and Kazuhisa Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 133–141. Springer, 2002.

[BGMP03] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. ExAnte: Anticipated data reduction in constrained pattern mining. In Lavrač et al. [LGBT03], pages 59–70.

[BHPI02] Mihai Bădoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 250–257. ACM, 2002.

[BJ98] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 85–93. ACM Press, 1998.

[BJAG00] Roberto J. Bayardo Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4(2/3):217–240, 2000.

[Bou04] Jean-François Boulicaut. Inductive databases and multiple uses of frequent itemsets: The cInQ approach. In Meo et al. [MLK04], pages 1–23.

[BR01] Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *Proceedings of the Twenteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*. ACM, 2001.

[BRB04] Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Constraint-based mining of formal concepts in transactional data. In Dai et al. [DSZ04], pages 615–624.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[BSH02] Artur Bykowski, Jouni K. Seppänen, and Jaakko Holmén. Model-independent bounding of the supports of Boolean formulae in binary data. In Mika Klemettinen and Rosa Meo, editors, *KDID*. Helsinki University Printing House, Helsinki, 2002.

[BTP+00] Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, and Lotfi Lakhai. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, 2000.

[Cal02] Christian S. Calude. *Algorithmic Information Theory: An Algorithmic Perspective*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2nd edition, 2002.

[Cal04a] Toon Calders. Computational complexity of itemset frequency satisfiability. In *Proceedings of the Twenty-Third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-18, 2004, Maison de la Chimie, Paris, France*. ACM, 2004.

[Cal04b] Toon Calders. Deducing bounds on the supports of itemsets. In Meo et al. [MLK04], pages 214–233.

[CD01] Marek Chrobak and Christoph Dürr. Reconstructing polyatomic structures from discrete X-rays: NP-completeness proof for three atoms. *Theoretical Computer Science*, 259(1–2):81–98, 2001.

[CG02] Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*, volume 2431 of *Lecture Notes in Artificial Intelligence*, pages 74–865. Springer, 2002.

[CG03a] Toon Calders and Bart Goethals. Minimal k-free representations of frequent sets. In Lavrač et al. [LGBT03], pages 71–82.

[CG03b] Gemma Casas-Garriga. Discovering unbounded episodes in sequential data. In Lavrač et al. [LGBT03], pages 83–94.

[CJCG+02] E. G. Coffman Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, and M. Yannakakis. Perfect packing theorems and the average-case behaviour of optimal and online bin packing. *SIAM Review*, 44(1):95–108, 2002.

[CLW01] Nick Cercone, Tsau Young Lin, and Xindong Wu, editors. *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*. IEEE Computer Society, 2001.

[Das02] Sanjoy Dasgupta. Performance guarantees for hierarchical clustering. In Jyrki Kivinen and Robert H. Sloan, editors, *Computational Learning Theory, 15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 8-10, 2002, Proceedings*, volume 2375 of *Lecture Notes in Artificial Intelligence*, pages 351–363. Springer, 2002.

[dlVKKR03] W. Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*. ACM, 2003.

[DR03] Luc De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2003.

[DRJLM02] Luc De Raedt, Manfred Jaeger, Sau Dan Lee, and Heikki Mannila. A theory of inductive query answering. In Kumar and Tsumoto [KT02], pages 123–130.

[DSZ04] Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors. *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, volume 3056 of *Lecture Notes in Artificial Intelligence*. Springer, 2004.

[DT01] Luc Dehaspe and Hannu T.T. Toivonen. Discovery of relational association rules. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 189–212. Springer, 2001.

[EC02] Vladimir Estivill-Castro. Why so many clustering algorithms – a position paper. *SIGKDD Explorations*, 4(1):65–75, 2002.

[ER01] Tapio Elomaa and Juho Rousu. On the computational complexity of optimal multisplitting. *Fundamenta Informaticae*, 47(1–2):35–52, 2001.

[ESAG02] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In Hand et al. [HKN02], pages 217–228.

[Fay01] Usama Fayyad. The digital physics of data mining. *Communications of the ACM*, 44(3):62–65, 2001.

[FCFM00] Martin Farach-Colton, Paolo Ferragina, and S. Muthukrishnan. On the sorting-complexity

of suffix tree construction. *Journal of the Association for Computing Machinery*, 47(6):987–1011, 2000.

[Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the Association for Computing Machinery*, 45(4):634 – 652, 1998.

[FG88] Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM Symposium on Theory of Computing, Chicago, Illinois, May 2–4, 1988*, pages 434–444. ACM, 1988.

[Fis58] Walter D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.

[FJ02] Csilla Farkas and Sushil Jajodia. The inference problem: A survey. *SIGKDD Explorations*, 4(2):6–11, 2002.

[FK98] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and Systems Science*, 57(2):187–199, 1998.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[FU02] Usama Fayyad and Ramasamy Uthurusamy. Evolving data mining into solutions for insights. *Communications of the ACM*, 45(8):28–31, 2002.

[Gal86] Zvi Galil. Efficient algorithms for finding maximum matchings in graphs. *ACM Computing Surveys*, 18(1):23–38, 1986.

[GAS03] Robert Gwadera, Mikhail Atallah, and Wojciech Szpankowski. Reliable detection of episodes in event sequences. In Wu et al. [WTS03], pages 67–74.

[GBLP96] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and subtotals. In Stanley Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 152–159. IEEE Computer Society, 1996.

[GDVW00] Peter Gritzmann, Sven De Vries, and Markus Wiegelmann. Approximating binary images from discrete X-rays. *SIAM Journal on Optimization*, 11(2):522–546, 2000.

[GGM03] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. What you store is what you get (extended abstract). In Boulicaut and Džeroski [BD03].

[GGM04] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In Arikawa and Suzuki [AS04], pages 278–289.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[GK97] R. Giegerich and S. Kurtz. From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19:331–353, 1997.

[GK99] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228 – 248, 1999.

[GKM+03] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharma. Discovering all most specific sentences.

ACM Transactions on Database Systems, 28(2):140–174, 2003.

[GLLM05] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In Choonsik Park and Seongtaek Chee, editors, *Proceedings of the 7th International Conference on Information Security and Cryptology – ICISC 2004 , Seoul, Korea, December 2–3, 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 104–120, 2005.

[Gol02] Oded Goldreich. Zero-knowledge twenty years after its invention. Technical Report 63, Electronic Colloquium on Computation Complexity, 2002.

[GS05] Bart Goethals and Arno Siebes, editors. *KDID 2004, Knowledge Discovery in Inductive Databases, Proceedings of the Third International Workshop on Knowledge Discovery in Inductive Databases, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, volume 3377 of *Lecture Notes in Computer Science*. Springer, 2005.

[GSDF03] Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*. ACM, 2003.

[GTY03] Gunter Grieser, Yuzuru Tanaka, and Akihiro Yamamoto, editors. *Discovery Science, 6th International Conference, DS 2003, Sapporo, Japan, October 17–19, 2003, Proceedings*, volume 2843 of *Lecture Notes in Computer Science*. Springer, 2003.

[GVdB00] Bart Goethals and Jan Van den Bussche. On supporting interactive association rule mining. In Yahiko Kambayashi, Mukesh K. Mohania, and A. Min Tjoa, editors, *DaWaK*, volume 1874 of *Lecture Notes in Computer Science*, pages 307–316. Springer, 2000.

[GVdB02] Bart Goethals and Jan Van den Bussche. Relational association rules: Getting WARMeR. In Hand et al. [HAB02], pages 125–139.

[GW99] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.

[GZ01] Karam Gouda and Mohammed J. Zaki. Efficiently mining maximal frequent itemsets. In Cercone et al. [CLW01], pages 163–170.

[GZ03] Bart Goethals and Mohammed J. Zaki, editors. *Proceedings of the Workshop on Frequent Itemset Mining Implementations (FIMI-03), Melbourne Florida, USA, November 19, 2003*, volume 90 of *CEUR Workshop Proceedings*, 2003. `http://CEUR-WS.org/Vol-90/`.

[HAB02] David J. Hand, Niall M. Adams, and Richard J. Bolton, editors. *Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK, September 16-19, 2002, Proceedings*, volume 2447 of *Lecture Notes in Computer Science*. Springer, 2002.

[HAK+02] Jiawei Han, Russ B. Altman, Vipin Kumar, Heikki Mannila, and Daryl Pregibon. Emerging scientific applications in data mining. *Communications of the ACM*, 45(8):54–58, 2002.

[Han02] David J. Hand. Pattern detection and discovery. In Hand et al. [HAB02], pages 1–12.

[HK99] Gabor T. Herman and Attila Kuba, editors. *Discrete Tomography: Foundations, Algorithms, and Applications*. Applied and Numerical Harmonic Analysis. Birkhäuser, 1999.

[HK01] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.

[HKN02] D. Hand, D. Keim, and R. Ng, editors. *Proceedings of the Eighth ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada. ACM, 2002.

[HMS93] Paul Helman, Bernard M. E. Moret, and Henry D. Shapiro. An exact characterization of greedy structures. *SIAM Journal on Discrete Mathematics*, 6(2):274 – 283, 1993.

[HMS01] David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, 2001.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Auotmata Theory, Languages and Computation*. Addison-Wesley, 2nd edition, 2001.

[HPYM04] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer-Verlag, 2001.

[IJ94] Robert W. Irving and Mark R. Jerrum. Three-dimensional statistical data security problems. *SIAM Journal on Computing*, 23(1):170–184, 1994.

[IKA02] Thomasz Imieliński, Leonid Khachiyan, and Amin Abdulghani. Cubegrades: Generalizing association rules. *Data Mining and Knowledge Discovery*, 6(3):219–257, 2002.

[IM96] Thomas Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Communications of The ACM*, 39(11):58–64, 1996.

[IWM03] Akihiro Inokuchi, Takshi Washio, and Hiroshi Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50:321–354, 2003.

[JKM+98] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. Optimal histograms with quality guarantees. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 275–286. Morgan Kaufmann, 1998.

[Juk01] Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2001.

[KGBW03] Daniel Kifer, Johannes Gehrke, Cristian Bucila, and Walker White. How to quickly find a witness. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 272–283. ACM, 2003.

[KK01] Michihiro Kurakochi and George Karypis. Frequent subgraph discovery. In Cercone et al. [CLW01], pages 313–320.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In Getoor et al. [GSDF03].

[Kle02] Jon Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, 2002.

[KMN+04] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry: Theory and Applications*, 28:89–112, 2004.

[Knu98] Donald E. Knuth. *Sorting and Seaching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, second edition, 1998.

[KRS02] Ron Kohavi, Neal J. Rothleder, and Evangelos Simoudis. Emerging trends in business analytics. *Communications of the ACM*, 45(8):45–48, 2002.

[Kry01] Marzena Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In Cercone et al. [CLW01], pages 305–312.

[KS02] David Kessler and Jeremy Schiff. Inclusion-exclusion redux. *Electronic Communications in Probability*, 7:85 – 96, 2002.

[KS03] Juha Kärkkäinen and Peter Sanders. Simple linear work suffix array construction. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*. Springer, 2003.

[KSS04] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 454–462. IEEE Computer Society, 2004.

[KT02] Vipin Kumar and Shusaku Tsumoto, editors. *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*. IEEE Computer Society, 2002.

[Kub89] Attila Kuba. Determination of the structure of the class $\mathcal{A}(r, s)$ of $(0, 1)$-matrices. *Acta Cybernetica*, 9(2):121–132, 1989.

[KVV04] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[LGBT03] Nada Lavrač, Dragan Gamberger, Hendrik Blockeel, and Ljupco Todorovski, editors. *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, volume 2838 of *Lecture Notes in Artificial Intelligence*. Springer, 2003.

[LJ83] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[LLN03] Laks V.S. Lakshmanan, Carson Kai-Sang Leung, and Raymond T. Ng. Efficient dynamic mining of constrained frequent sets. *ACM Transactions on Database systems*, 28(4):337–389, 2003.

[LP86] L. Lovász and M.D. Plummer. *Matching Theory*, volume 121 of *Annals of Discrete Mathematics*. North-Holland, 1986.

[LV97] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer-Verlag, 3rd edition, 1997.

[LV03] Peter Lyman and Hal R. Varian. How much information, 2003. `http://www.sims.berkeley.edu/how-much-info-2003`.

[Man97] Heikki Mannila. Inductive databases and condensed representations for data mining. In Jan Maluszynski, editor, *Logic Programming, Proceedings of the 1997 International Symposium, Port Jefferson, Long Island, N.Y., October 13-16, 1997*, pages 21–30. MIT Press, 1997.

[Man02] Heikki Mannila. Local and global methods in data mining: Basic techniques and open problems. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP*

2002, Malaga, Spain, July 8-13, 2002, Proceedings, volume 2380 of *Lecture Notes in Computer Science*, pages 57–68. Springer, 2002.

[Mie03a] Taneli Mielikäinen. Chaining patterns. In Grieser et al. [GTY03], pages 232–243.

[Mie03b] Taneli Mielikäinen. Change profiles. In Wu et al. [WTS03], pages 219–226.

[Mie03c] Taneli Mielikäinen. Finding all occurring sets of interest. In Boulicaut and Džeroski [BD03], pages 97–106.

[Mie03d] Taneli Mielikäinen. Frequency-based views to pattern collections. In Peter L. Hammer, editor, *Proceedings of the IFIP/SIAM Workshop on Discrete Mathematics and Data Mining, SIAM International Conference on Data Mining (2003), May 1-3, 2003, San Francisco, CA, USA*. SIAM, 2003.

[Mie03e] Taneli Mielikäinen. On inverse frequent set mining. In Wenliang Du and Christopher W. Clifton, editors, *Proceedings of the 2nd Workshop on Privacy Preserving Data Mining (PPDM), November 19, 2003, Melbourne, Florida, USA*, pages 18–23. IEEE Computer Society, 2003.

[Mie04a] Taneli Mielikäinen. Inductive databases as ranking. In Yahiko Kambayashi, Mukesh K. Mohania, and Wolfram Wöß, editors, *Data Warehousing and Knowledge Discovery, 5th International Conference, DaWaK 2004, Zaragoza, Spain, September 1-3, 2003, Proceedings*, Lecture Notes in Computer Science. Springer, 2004.

[Mie04b] Taneli Mielikäinen. Privacy problems with anonymized transaction databases. In Arikawa and Suzuki [AS04], pages 219–229.

[Mie04c] Taneli Mielikäinen. Separating structure from interestingness. In Dai et al. [DSZ04], pages 476–485.

[Mie05a] Taneli Mielikäinen. An automata approach to pattern collections. In Goethals and Siebes [GS05].

[Mie05b] Taneli Mielikäinen. Implicit enumeration of patterns. In Goethals and Siebes [GS05].

[Mit82] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[MLK04] Rosa Meo, Pier Luca Lanzi, and Mika Klemettinen, editors. *Database Support for Data Mining Applications: Discovering Knowledge with Inductive Queries*, volume 2682 of *Lecture Notes in Computer Science*. Springer, 2004.

[MM03] Taneli Mielikäinen and Heikki Mannila. The pattern ordering problem. In Lavrač et al. [LGBT03], pages 327–338.

[MS03] Jérôme Maloberti and Einoshin Suzuki. Improving efficiency of frequent query discovery by eliminating non-relevant candidates. In Grieser et al. [GTY03], pages 220–232.

[MT96] Heikki Mannila and Hannu Toivonen. Multiple uses of frequent sets and condensed representations. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 189–194. AAAI Press, 1996.

[MT97] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[MTV97] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[OZS04] Stanley R. M. Oliveira, Osmar R. Zaïane, and Yücel Saygin. Secure association rule sharing. In Dai et al. [DSZ04], pages 74–85.

[Pap95] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1995.

[PB02] Carlotta Piscopo and Mauro Birattari. Invention vs. discovery. In Steffen Lange, Ken Satoh, and Carl H. Smith, editors, *Discovery Science, 5th International Conference, DS 2002, Lübeck, Germany, November 24–26, 2002, Proceedings*, volume 2534 of *Lecture Notes in Computer Science*, pages 457–462. Springer, 2002.

[PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In Catriel Beeri and Peter Buneman, editors, *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings*, volume 1540 of *Lecture Notes in Computer Science*, pages 398–416. Springer, 1999.

[PCT+03] Feng Pan, Gao Cong, Anthony K. H. Tung, Jiong Yang, and Mohammed J. Zaki. CARPENTER: Finding closed patterns in long biological datasets. In Getoor et al. [GSDF03].

[PDZH02] Jian Pei, Guozhu Dong, Wei Zou, and Jiawei Han. On computing condensed pattern bases. In Kumar and Tsumoto [KT02], pages 378–385.

[Pee03] René Peeters. The maximum edge biclique is NP-complete. *Discrete Applied Mathematics*, 131:651–654, 2003.

[Pin02] Benny Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explorations*, 4(2):12–19, 2002.

[PMS03] Dmitry Pavlov, Heikki Mannila, and Padhraic Smyth. Beyond independence: probabilistic methods for

query approximation on binary transaction data. *IEEE Transactions on Data and Knowledge Engineering*, 15(6):1409–1421, 2003.

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[Rie03] Erik Riedel. Storage systems: Not just a bunch of disks anymore. *ACM Queue*, 1(4):32–41, 2003.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In F. Tom Leighton and Peter Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 475–484. ACM, 1997.

[Sch03] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, 2003.

[SPF02] Padhraic Smyth, Daryl Pregibon, and Christos Faloutsos. Data-driven evolution of data mining algorithms. *Communications of the ACM*, 45(8):33–37, 2002.

[SU03] Ken Satoh and Takeaki Uno. Enumerating maximal frequent sets using irredundant dualization. In Grieser et al. [GTY03], pages 256–268.

[SVA97] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, and Daryl Pregibon, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, pages 67–73. AAAI Press, 1997.

[SVC01] Yücel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.

[TKR+95] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen, and Heikki Mannila. Pruning and grouping discovered association rules. In *ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases, Heraklion, Greece, April 1995*, pages 47–52, 1995.

[TKS02] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In Hand et al. [HKN02].

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 134–145. Morgan Kaufmann, 1996.

[UAUA04] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In Arikawa and Suzuki [AS04], pages 16–31.

[VBF+04] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yücel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[VC02] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In Hand et al. [HKN02], pages 639–644.

[VEEB+04] Vassilios S. Verykios, Ahmed K. Elmagarmid, Fellow Elisa Bertino, Yücel Saygin, and Elena Dasseni. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):434–447, 2004.

[WH04] Jianyong Wang and Jiawei Han. BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering (ICDE 2004)*. IEEE Computer Society, 2004.

[WHP03] Jianyong Wang, Jiawei Han, and Jian Pei. CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In Getoor et al. [GSDF03].

[WK04] Jianyong Wang and George Karypis. Summary: Efficiently summarizing transactions for clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 241–248. IEEE Computer Society, 2004.

[WTS03] Xindong Wu, Alex Tuzhilin, and Jude Shavlik, editors. *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*. IEEE Computer Society, 2003.

[WWS+02] Xiong Wang, Jason T.L. Wang, Dennis Shasha, Bruce A. Shapiro, Isidore Rigoutsos, and Kaizhong Zhang. Finding patterns in three-dimensional graphs: Algorithms and applications to scientific data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):731–749, 2002.

[WWWL05] Xintao Wu, Ying Wu, Yogge Wang, and Yingjiu Li. Privacy-aware market basket data set generation: A feasible approach for inverse frequent set mining. In *Proceedings of the Fifth SIAM International Conference on Data Mining*. SIAM, 2005.

[XYLD03] Yongqiao Xiao, Jenq-Foung Yao, Zhigang Li, and Margaret H. Dunham. Efficient data mining for maximal frequent subtrees. In Wu et al. [WTS03], pages 379–386.

[Yen04] John Yen. Emerging technologies for homeland security. *Communications of the ACM*, 47(3):33–35, 2004.

[YH02] Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In Kumar and Tsumoto [KT02], pages 721–724.

[YH03] Xifeng Yan and Jiawei Han. CloseGraph: mining closed frequent graph patterns. In Getoor et al. [GSDF03], pages 286–295.

[YHA03] Xifeng Yan, Jiawei Han, and Ramin Afshar. CloSpan: Mining closed sequential patterns in large data sets. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*. SIAM, 2003.

[Zak00] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[Zak01] Mohammed J. Zaki. SPADE: An efficient algoritm for mining frequent sequences. *Machine Learning*, 42:31–60, 2001.

[Zak02] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In Hand et al. [HKN02].

[ZH02] Mohammed J. Zaki and Ching-Jui Hsiao. CHARM: An efficient algorithms for closed itemset mining. In Robert L. Grossman, Jiawei Han, Vipin Kumar, Heikki Mannila, and Rajeev Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11-13, 2002*. SIAM, 2002.

[ZO98] Mohammed Javeed Zaki and Mitsunori Ogihara. Theoretical foundations of association rules. In *SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1998.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-1996-1 R. Kaivola: Equivalences, preorders and compositional verification for linear time temporal logic and concurrent systems. 185 pp. (Ph.D. thesis).

A-1996-2 T. Elomaa: Tools and techniques for decision tree learning. 140 pp. (Ph.D. thesis).

A-1996-3 J. Tarhio & M. Tienari (eds.): Computer Science at the University of Helsinki 1996. 89 pp.

A-1996-4 H. Ahonen: Generating grammars for structured documents using grammatical inference methods. 107 pp. (Ph.D. thesis).

A-1996-5 H. Toivonen: Discovery of frequent patterns in large data collections. 116 pp. (Ph.D. thesis).

A-1997-1 H. Tirri: Plausible prediction by Bayesian inference. 158 pp. (Ph.D. thesis).

A-1997-2 G. Lindén: Structured document transformations. 122 pp. (Ph.D. thesis).

A-1997-3 M. Nykänen: Querying string databases with modal logic. 150 pp. (Ph.D. thesis).

A-1997-4 E. Sutinen, J. Tarhio, S.-P. Lahtinen, A.-P. Tuovinen, E. Rautama & V. Meisalo: Eliot – an algorithm animation environment. 49 pp.

A-1998-1 G. Lindén & M. Tienari (eds.): Computer Science at the University of Helsinki 1998. 112 pp.

A-1998-2 L. Kutvonen: Trading services in open distributed environments. 231 + 6 pp. (Ph.D. thesis).

A-1998-3 E. Sutinen: Approximate pattern matching with the q-gram family. 116 pp. (Ph.D. thesis).

A-1999-1 M. Klemettinen: A knowledge discovery methodology for telecommunication network alarm databases. 137 pp. (Ph.D. thesis).

A-1999-2 J. Puustjärvi: Transactional workflows. 104 pp. (Ph.D. thesis).

A-1999-3 G. Lindén & E. Ukkonen (eds.): Department of Computer Science: annual report 1998. 55 pp.

A-1999-4 J. Kärkkäinen: Repetition-based text indexes. 106 pp. (Ph.D. thesis).

A-2000-1 P. Moen: Attribute, event sequence, and event type similarity notions for data mining. 190+9 pp. (Ph.D. thesis).

A-2000-2 B. Heikkinen: Generalization of document structures and document assembly. 179 pp. (Ph.D. thesis).

A-2000-3 P. Kähkipuro: Performance modeling framework for CORBA based distributed systems. 151+15 pp. (Ph.D. thesis).

A-2000-4 K. Lemström: String matching techniques for music retrieval. 56+56 pp. (Ph.D.Thesis).

A-2000-5 T. Karvi: Partially defined Lotos specifications and their refinement relations. 157 pp. (Ph.D.Thesis).

A-2001-1 J. Rousu: Efficient range partitioning in classification learning. 68+74 pp. (Ph.D. thesis)

A-2001-2 M. Salmenkivi: Computational methods for intensity models. 145 pp. (Ph.D. thesis)

A-2001-3 K. Fredriksson: Rotation invariant template matching. 138 pp. (Ph.D. thesis)

A-2002-1 A.-P. Tuovinen: Object-oriented engineering of visual languages. 185 pp. (Ph.D. thesis)

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. thesis)

A-2004-8 T. Kivioja: Computational Tools for a Novel Transcriptional Profiling Method. 98 pp. (Ph.D. thesis)

A-2004-9 H. Tamm: On Minimality and Size Reduction of One-Tape and Multitape Finite Automata. 80 pp. (Ph.D. thesis)