Dipartimento di Informatica
Università di Pisa

Ph.D. Thesis: TD-01/06

Abduction and Anonymity in Data Mining

Maurizio Atzori

Supervisor	Supervisor
Paolo Mancarella	Franco Turini

L.go B. Pontecorvo 3, I-56127 Pisa - Italy
atzori@di.unipi.it

“Although nature commences with reason and ends with experience
it is necessary for us to do the opposite, that is to commence with experience and
from this to proceed to investigate the reason.”
Leonardo da Vinci

Abstract

This thesis investigates two new research problems that arise in modern data mining: reasoning on data mining results, and privacy implication of data mining results.

Most of the data mining algorithms rely on inductive techniques, trying to infer information that is generalized from the input data. But very often this inductive step on raw data is not enough to answer the user questions, and there is the need to process data again using other inference methods. In order to answer high level user needs such as explanation of results, we describe an environment able to perform *abductive* (hypothetical) reasoning, since often the solutions of such queries can be seen as the set of hypothesis that satisfy some requirements. By using cost-based abduction, we show how classification algorithms can be boosted by performing abductive reasoning over the data mining results, improving the quality of the output.

Another growing research area in data mining is the one of privacy-preserving data mining. Due to the availability of large amounts of data, easily collected and stored via computer systems, new applications are emerging, but unfortunately privacy concerns make data mining unsuitable. We study the privacy implications of data mining in a mathematical and logical context, focusing on the anonymity of people whose data are analyzed. A formal theory on *anonymity-preserving data mining* is given, together with a number of anonymity-preserving algorithms for pattern mining.

The post-processing improvement on data mining results (w.r.t. utility and privacy) is the central focus of the problems we investigated in this thesis.

Acknowledgments

My wonderful advisors Paolo Mancarella and Franco Turini for believing in me, helping me and being my role models and mentors. All the people of the KDD-Lab in Pisa and particularly Fosca Giannotti and Dino Pedreschi for supporting me, guiding me and sharing with me exciting moments of research life (of course I will never forget PKDD/ECML 2004 “back of the stage”!) Francesco Bonchi for his friendship, advice, and good work as a coauthor of common papers. Christopher W. Clifton and Antonis C. Kakas for reviewing my work and giving very valuable comments and suggestions. Ivan Lanese and Roberto Zunino for their enthusiasm and love for science. Miriam Baglioni and Andrea Bracciali for bearing me in our shared office. All the professors, friends, people I am not explicitely mentioning that took part of my life during my PhD program.

Dulcis in fundo, my parents Chiara and Efisio, my brother Stefano and my best friend Sary for encouraging me during these years.

Thank you all.

Contents

Introduction

Due to the availability of large amounts of data, easily collected and stored via computer systems, the field of data mining is gaining momentum. Data Mining, one of the most important steps in the process of Knowledge Discovery, has received enormous attentions by the research community. Several important results have been obtained in the context of specific algorithms, in applying the techniques in several application fields, and in designing suitable environments in which the data mining step can be embedded. Such environments support the phases that come before (e.g., cleaning the data) and the ones that come after (e.g., visualization of results), and attempt also at providing a context in which one can process the results of the data mining step in order to answer higher level questions than the ones directly provided by the computed data mining model. For example, extracting association rules from a set of supermarket transactions is very useful, and can answer basic questions like "which are the items that induce a buying attitude toward other items?", but it would be even more interesting answering the question "did the new layout of the store influence the buying attitude?". Answering the last question requires taking the association rules computed with respect to the old layout and comparing them with the ones computed according to the new ones, possibly taking into account the rules underneath the new design.

We believe that the results of data mining algorithms may be the input to suitable environments that can elaborate and exploit them in order to satisfy the initial needs of the user in a fully automatic way. This thesis focuses on the output of the data mining results. We investigate on two new research problems which arise in modern Data Mining: improving and reasoning on data mining results (by using abduction), and privacy implication of data mining results (by anonymization).

I.1 Abduction in Data Mining

Induction means generalization. Most data mining algorithms rely on inductive techniques, trying to infer information that is generalized from the input data. But very often this inductive step on raw data is not enough to answer the user questions, and there is the need to process data again using deductive techniques. In order to answer high level questions, we need an environment able to perform abductive reasoning, since often the solutions of such queries can be seen as the set of hypothesis that satisfy some requirements [Men96, KM98]. Our claim is that abductive reasoning environments can be shown to be useful in data mining to represent high level knowledge and questions. In this thesis (Chapter 4) we show that abduction can be profitably used for data mining purposes such as classification and explanation of the data mining results. We develop a general framework for abductive reasoning that exploits classification rules induced from the data and information coming from domain experts. Then we exploit also association rules, together with their support and confidence values, improving the quality of the results and making the process almost automatic, without the need of having prior knowledge about the domain.

I.2 Anonymity in Data Mining

When personal data is analyzed, one important question to take into account is whether the analysis violates the privacy of individuals whose data is referred to. The importance of privacy is growing constantly, and several application of data mining to real data have been stopped because of the privacy implication [DAR03, Fun04, Com03].

In very recent years, the field of privacy-preserving data mining was born and traced some interesting research directions, with the aim of making data mining aware of the privacy implication and enabling new real-world applications.

In this thesis (Chapter 5) we concentrate on individual privacy, in the strict sense of *non-identifiability*, as prescribed by the European Union regulations on privacy, as well as US rules on protected health information (HIPAA rules). Privacy is regulated at the European level by Directive 95/46/EC (Oct. 24, 1995) and Regulation (EC) No 45/2001 (December 18, 2000). In such documents, general statements about identifiability of an individual are given, such as:

> *"To determine whether a person is identifiable, account should be taken of all the means likely to be reasonably used either by the controller or by any person to identify the said person. The principles of protection should not apply to data rendered anonymous in such a way that the data subject is no longer identifiable."*

According to this perspective, we focus on *anonymity* of individuals, and ask ourselves whether the disclosure of patterns extracted by data mining techniques may open up the risk of privacy breaches that may reveal individual identity. We show in this thesis that the answer to this question is, unfortunately, positive: data mining results can indeed violate anonymity. This observation originates the following research problem: *is it possible to devise well-founded definitions and measures of privacy that provably prevent the violation of anonymity in the mining results, hence guaranteeing non-identifiability?*

We believe that a solution to this problem would be an enabling factor of many emerging applications such as medical and geographic applications, as described in Chapter 3, currently not possible because of privacy implications.

I.3 Thesis Organization

This thesis is organized as follows. In Chapter 1 we sketch some basic concepts on most data mining paradigms, and we will set up the notations and terminology used throughout this thesis. We also describe our novel approach to compute efficiently sets that are frequent in the data. In data mining they are called frequent itemsets, and we deeply use them both in our abductive framework (to compute association rules) and in our anonymity-preserving data mining framework (to compute inference channels). When the number of possible frequent itemsets is huge, only memory-aware algorithms, as the one we developed, can be adopted to compute them.

In Chapter 2 we briefly describe the state of the art of artificial intelligence on abductive systems and on hybrid systems that make a combined use of abduction and induction. Chapter 3 describes the work in literature on privacy-preserving data mining. Our contributions begin at Chapter 4, that is devoted to our results on the use of abductive reasoning for data mining, describing the way to construct a data mining framework able to answer simple abductive queries. We describe how classification problems can be mapped into abductive queries and we experimentally show that classification results are boosted by the system. Our definition and results on anonymity-preserving data mining are shown in Chapter 5. In each Chapter we also sketch some research topics we would like to explore in future work in order to extend our current results on abduction and anonymity in data mining.

Chapter 1

Background on Data Mining

Abstract

In this Chapter we will review the most important data mining paradigms and models: classification, association rules and clustering. We will also give the notation of frequent itemsets, used throughout this thesis, together with a survey of the classic work done in this field. Finally, we describe our novel approach to mine frequent itemsets that, for sparse datasets, requires little memory and a (small) constant number of passes over the input.

1.1 Data Mining

Data mining can be defined as the process of finding correlations or patterns among dozens of fields in large relational databases. It is an essential step of the knowledge discovery process where intelligent methods are applied in order to extract data patterns from very large databases [HK00]. In particular, the most important model to describe data in an informative way, developed in this area, are: the *classification* task, that is predicting categorical labels given some examples; *association rules*, that is finding out meaningful dependencies among set of attributes; *clustering*, that is finding a good partition of a set of elements (for instance, a partition of the dataset).

1.1.1 Classification by Decision Trees

Classification of data requires two sequential steps: the first one consists in building a model that describes a given set of examples by associating a class label to each of them; the second one concerns using the model to classify new examples (i.e., predict the categorical label). In literature, there are different models to compute the classification tasks, but the only model we are currently interested for its properties is the *decision tree*.

A decision tree is a tree structure in which each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represent classes. Decision tree induction consists in building such a tree from a training set of examples and then using it (following a path from the root to a leaf) to classify new examples given their attribute values. Because of their structure, it is natural to transform decision trees into classification rules, that can be easily inserted into a reasoning framework. Notice that some machine learning tools, such as C4.5 [Qui93], already include a class rulesets generator.

In Chapter 4 we present our work on how class rulesets can be embedded into an abductive reasoning framework that allows us, in some cases, to better classify new examples in presence of external information, such as specific domain knowledge.

Let us now set up the main notations and terminologies we will use throughout this proposal as far as decision trees are concerned. Let $\mathcal{A}$ be a set of attribute names and $\mathcal{C}$ be a set of classes (possible classifications). For simplicity, we assume that each attribute can be assigned a value over a finite set of values $\mathcal{V}$. An *example* e is a set of attribute/values pairs

$$e = \{a_1 = v_1, \ldots, a_n = v_n\}$$

where a_i is the i-th attribute name.

Definition 1. *A* d*ecision tree T over $\mathcal{A}$ and $\mathcal{C}$ is a tree such that:*

(i) each non-leaf node is labeled by an attribute $a \in \mathcal{A}$;

(ii) each leaf node is labeled by a class $c \in \mathcal{C}$;

(iii) each branch is labeled by a value $v \in \mathcal{V}$;

(iv) the values labeling all the branches exiting from a given node are all distinct;

(v) the labels of a path are all distinct.

Notice that *(v)* formalizes the fact that, in each path, only one test can be performed on each attribute.

Example 1. Let us consider a very well known example, taken from [Qui93]. Given a training set of examples which represent some situations, in terms of weather conditions, in which it is or it is not the case that playing tennis is a good idea, a decision tree is built which can be used to classify further examples as good candidates for playing tennis (class Yes) and bad candidates to play tennis (class No). Table 1.1 shows the original training set, given as a relational table over the attributes $\{Overlook, Temperature, Humidity, Wind\}$. The last column (*class* or *target attribute*) of the table represents the classification of each row. Several algorithms have been developed to mine a decision tree from datasets such as the one in Table 1.1. Almost all of them rely on the the basic recursive schema used

Overlook	Temperature	Humidity	Wind	Class
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rainy	Mild	High	Weak	Yes
Rainy	Cool	Low	Weak	Yes
Rainy	Cool	Low	Strong	No
Overcast	Cool	Low	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Low	Weak	Yes
Rainy	Mild	Low	Weak	Yes
Sunny	Mild	Low	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Low	Weak	Yes
Rainy	Mild	High	Strong	No

Table 1.1: Training set of examples on attributes *Overlook, Temperature, Humidity, Wind*. The values *Yes* and *No* represent the concept to learn

in the ID3 algorithm (see Algorithm 1). Differences between algorithms usually depend on the $information\text{-}gain$ procedure, that is responsible for the choice of the (local) best attribute to use as a node. Using such a standard decision tree inductive algorithm, we may obtain the decision tree in Fig. 1.1 from the above training set. As we have already pointed out, each internal node represents a test on a single attribute and each branch represents the outcome of the test. A path in the decision tree represents the set of attribute/value pairs that an example should exhibit in order to be classified as an example of the class labeled by the leaf node. For instance, given the above tree, the example $\{Overlook = Sunny, Humidity = Low\}$ is classified as Yes, whereas the example $\{Overlook = Sunny, Humidity = High\}$ is classified as No. Notice that not all the attribute values have to be specified in order to find the classification of an example. On the other hand, if an example is too under-specified, it may lead to different, possibly incompatible, classifications. For instance, the example $\{Overlook = Sunny\}$ can be classified both as Yes or No, following the two left-most branches of the tree: in this case many decision tree based classifiers make a choice by using probabilities assigned to each leaf. It is also worth noticing that the decision tree may not consider all the attributes given in the training set. For instance, the attribute $Temperature$ is not taken into account at all in this decision tree.

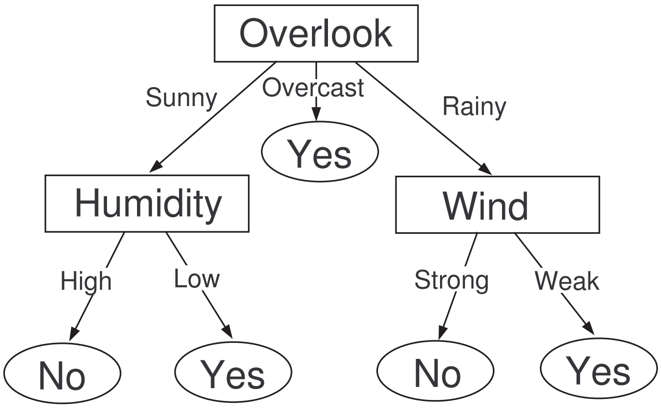


Figure 1.1: A well-know example of decision tree, obtained from the training set of Table 1.1

Algorithm 1 ID3(Examples, TargetAttribute, Attributes)

Input: Examples, TargetAttribute, Attributes

Output: the root of the decision tree;

```
if $\forall e \in Examples\ .\ TargetAttribute$ of $e$ is equal to $+$ then
    return a leaf node with $label \leftarrow +$;
if $\forall e \in Examples\ .\ TargetAttribute$ of $e$ is equal to $-$ then
    return a leaf node with $label \leftarrow -$;
if $Attributes = \emptyset$ then
    return a leaf with $label \leftarrow$ most frequent target attribute in the examples;
$A \leftarrow \max_{X \in Attribute} gain\text{-}information(X)$;
create a non-leaf node $N$ with $label \leftarrow A$;
for all $v_i \in A$ do
    add a tree branch to the node $N$, with $label \leftarrow A = v_i$;
    $Examples(v_i) \leftarrow \{e \in Examples \mid$ the value of $A$ in $e$ is equal to $v_i\}$;
    $N_{A=v_i} \leftarrow ID3(Examples(vi), TargetAttribute, Attributes \setminus \{A\}$;
    attach the subtree $N_{A=v_i}$ to the branch of $N$ with label $A = v_i$;
return node $N$;
```

1.1.2 Association Rules

Association rules are useful to determine correlations between attributes of large sets of data items. Association rules show attribute value conditions that occur frequently together in a given dataset [ZO98, Fre00]. A typical and widely-used example of association rule mining is Market Basket Analysis.

Example 2. In the Market Basket Analysis problem, data are collected using barcode scanners in supermarkets. Such market basket databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Managers would be interested to know if certain groups of items are consistently purchased together. They could use this data for adjusting store layouts (placing items optimally with respect to each other), for cross-selling, for promotions, for catalog design and to identify customer segments based on buying patterns.

Association rules provide information of this type in the form of *if-then* statements. These rules are computed from the data and, unlike the *if-then* rules of logic, association rules are probabilistic in nature. In fact, in addition to the antecedent (the *if* part) and the consequent (the *then* part), an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called itemsets and formally described later in this Chapter) that are disjoint (they have no item in common). The first number is called the *support* for the rule. The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule, although often it is expressed as a percentage of the total number of records in the database. The other number is known as the *confidence* of the rule. Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

Example 3. A supermarket database has $100,000$ point-of-sale transactions, out of which $2,000$ include both items A and B, and 800 of these include also item C. The association rule "If A and B are purchased then C is purchased on the same trip" has a support of 800 transactions (alternatively $0.8\% = 800/100,000$) and a confidence of 40% ($= 800/2,000$).

One way to think of support is that it is the probability that a randomly selected transaction from the database will contain all items in the antecedent and the consequent, whereas the confidence is the conditional probability that a randomly selected transaction will include all the items in the consequent given that the transaction includes all the items in the antecedent.

Lift is one more parameter of interest in association analysis [TKS02]. It is the ratio of Confidence to Expected Confidence, where the Expected Confidence is the Confidence in the case the antecedent does not influence the consequent. The Expected Confidence can be defined as the number of transactions that include the consequent divided by the total number of transactions.

Example 4. In our previous supermarket example, let be $5,000$ the number of transactions that contain C. Thus Expected Confidence is $5,000/100,000 = 5\%$. The Lift is:

$$\text{Lift} = \frac{\text{Confidence}}{\text{Expected Confidence}} = \frac{40\%}{5\%} = 8$$

Hence lift is a value that gives us information about the increase in probability of the *then* (consequent) given the *if* (antecedent) part.

Algorithm 2 *Apriori*$(L_k, C_k, \mathcal{D})$

```
Input: a minimum (absolute) support threshold $\sigma$, a dataset $\mathcal{D}$
Output: set of all frequent itemset;
$L_1 \leftarrow \{$ frequent 1-itemsets$\}$ //direct count on $\mathcal{D}$;
$k \leftarrow 2$;
while $L_{k-1} \neq \emptyset$ do
    $C_k \leftarrow \{I \mid (|I| = k) \wedge \forall J \subset I, |J| = k - 1.(J \in L_{k-1})\}$
    for all transactions $t \in \mathcal{D}$ do
        for all $s \in t$ s.t. $|s| = k$ do
            if $s \in$ then
                $count(s) \leftarrow count(s) + 1$;
    $L_k \leftarrow \{s \in C_k \mid count(s) \geq \sigma\}$
    $k \leftarrow k + 1$;
return $\bigcup_k L_k$;
```

The task of discovering association rules given (minimum) support and confidence thresholds can be decomposed into two subproblems:

- Find all sets of items (itemsets) that have transaction support above minimum support. The support for an itemset is the number of transactions that contain the itemset. Itemsets with support greater or equals to the threshold are called frequent itemsets[1], and are discussed in Section 1.2. This subtask is addressed by the algorithm *APRIORI* [AS94] shown in Algorithm 2;

- Generate all association rules with minimum support and confidence from the set of all large itemsets. This subtask can be addressed by a straightforward algorithm: For each large itemset l, find all non-empty subsets. For each such subset a of l, output the rule a $l \rightarrow a$, if and only if confidence $= \frac{\text{support}(l)}{\text{support}(a)} \geq$ minconf (confidence threshold).

Association rule mining has been used in order to solve also classification tasks (see for example [Coh95, JWBV, LHP01, Fre00]). See also [HGN00, GZ03] for a survey on association rules and frequent itemsets mining algorithms.

1.1.3 Clustering

Clustering is the partition of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. The task of clustering is to structure a given set of unclassified instances of an

[1] Sometimes they are also called *large itemsets*.

example language by creating concepts, based on similarities found on the training data.

Unlike association rules and classifiers, there is typically no specific outcome or attribute that must be predicted. This means that the search for clusters is unsupervised learning. The main difference respect to supervised learning is that there is neither a target predicate nor an oracle, dividing the instances of the training set into categories. The categories are formed by the learner itself.

The clustering task can be formalized in the following way.

Given: A set of (unclassified) instances of an example language.

Find: A set of concepts that cover all given examples, such that

1. the similarity between examples of the same concepts is maximized,
2. the similarity between examples of different concepts is minimized.

Clustering is the subject of active research in several fields such as economics, statistics, pattern recognition, and machine learning [Ber02, KPR99]. Data mining adds to clustering the complications of very large datasets with very many attributes of different types. This imposes unique computational requirements on relevant clustering algorithms.

For marketing purposes, some form of clustering are essential. It's difficult to find a single marketing message that appeals to all customers, or a single advertising medium that will reach all our customers at a reasonable price. Clustering can also yield opportunities: finding a group of similar customers may open up a way in which a product can be extended or combined specifically for that group of similar customers.

1.2 Frequent Itemset Mining

In this Section we set up the basic notations and terminology that we are going to adopt in the rest of this thesis, formally defining the *frequent itemset mining* problem. A survey of the most important results in this field is also given.

Let us first set up some notational conventions used in the sequel.

Set of items A set of items is a finite set, denoted by $\mathcal{I}$. Elements of $\mathcal{I}$ will be denoted by $i, i', i_1, \ldots$ and are referred to as *items.*

Itemset An itemset is a subset of $\mathcal{I}$. Itemsets will be denoted by $I, I', \ldots$.

k-itemset An itemset I is called a k-itemset if $|I| = k$, where $|I|$ denotes the cardinality of I.

Transaction A transaction T is an itemset, denoted by $T, T', \ldots$.

Dataset A *dataset* $\mathcal{D}$ is a multiset (a bag) of transactions. Given $\mathcal{D}$, the *maximal transaction length* of $\mathcal{D}$ is

$$m_{\mathcal{D}} = \max\{|T| \mid T \in \mathcal{D}\}.$$

In order to show some properties and theorems in the next sections, let us formally define the frequent itemsets mining problem.

Definition 2 (FIM Problem). *Let $\mathcal{D}$ be a dataset and I be an itemset. I is called a* frequent *itemset with respect to $\mathcal{D}$ and a support σ, with $0 < \sigma \leq 1$ if:*

$$|\{T \in \mathcal{D} \mid I \subseteq T\}| \geq \sigma|\mathcal{D}|.$$

Let $\mathcal{F}_k(\sigma, \mathcal{D})$ be the set of all k-itemset that are frequent w.r.t. σ and $\mathcal{D}$. Then, the FIM problem is defined as the task of determining $\mathcal{F}_k(\sigma, \mathcal{D})$ for each k such that $0 < k \leq m_{\mathcal{D}}$.

In the sequel we will often write simply $\mathcal{F}_k$ instead of $\mathcal{F}_k(\sigma, \mathcal{D})$, whenever the parameters σ and $\mathcal{D}$ are either clear from the context or irrelevant.

1.2.1 Related Work on Frequent Itemset Mining

Many algorithms for finding frequent itemsets are based on the level-wise generation of candidates of the *Apriori* algorithm [AS94]. The level-wise approach is performed in order to maintain the search space small enough to fit into the main memory. This strategy necessarily leads to several passes through the dataset.

Some other works in literature present different approaches in order to keep the number of passes through the dataset constant. In [SON95] the authors show a partitioning technique that needs two passes through the database. First, the dataset is partitioned into several parts which are small enough to fit into the memory. Every partition is elaborated using a level-wise algorithm and then the results of each partition are merged. This leads to a superset of the solution. A second scan then removes the false positive elements of the superset. Unfortunately, if the dataset is very large then the resulting partitions can be too small with respect to the dataset, leading to a huge superset, and this can reduce the effectiveness of the algorithm.

Another important approach to reduce the number of passes through the dataset is the one proposed by Toivonen in [Toi96] and then refined in [CHS02, BC02], based on the evaluation of a small random sample of the dataset. A set of patterns that are probably frequent in the whole dataset are generated, and then their exact frequencies are verified in the rest of the dataset. If a failure occurs in the generation of candidates (i.e., there are false negatives), a mechanism is provided which, in a second pass, computes the remaining frequent patterns. By decreasing the support threshold the probability of failure can be decreased, but for low probabilities this drastically increments the number of candidates. Furthermore, if we are dealing with very large datasets, it is possible that the (small) sample is not very representative of the whole dataset, and this means that there is a high probability of failure. In this case the candidates to be verified in the second pass can be too many to fit into the main memory (i.e., more than two passes are needed). For a survey on frequent itemset mining algorithms, see [Goe03].

Main memory usage of depth-first (i.e., non-levelwise) frequent pattern mining algorithms is discussed in [Goe04]: two state-of-the-art algorithms, FP-Growth [HPY00] and Eclat [Zak00], are tested and shown to be very memory consuming even for medium-size datasets. A simple improvement of Eclat, named Medic, is proposed but it is empirically shown to reduce the amount of memory needed of $\approx 50\%$ in the best case: the memory requirements still depend on the size of the dataset, and this fact leaves the algorithm impractical when datasets are very large. Another confirmation of the scalability limitations of current state-of-the-art algorithms for frequent itemset mining came from the First IEEE ICDM Workshop on Frequent Itemset Mining Implementations, FIMI 2003 [GZ03], where several well-known algorithms were implemented and independently tested. The results show that "*none of the algorithms is able to gracefully scale-up to very large datasets, with millions of transactions*".

Recently [AMT05] we developed a novel algorithm to mine frequent itemsets that, for sparse datasets, requires few memory while keeping a (small) constant number of passes over the input. It also overcomes the problem of dealing with millions of transactions by reducing the search space while maintaining the soundness. We describe this approach in the following Section.

1.3 Memory-Aware Frequent Set Mining

1.3.1 Contribution and Organization

In this Section we will focus on the problem of finding itemsets of a given size directly, i.e., without generating smaller itemsets as is done in level-wise approaches. Mining all the frequent itemsets would reduce memory and time efficiency; on the other hand, maximal itemset mining may not be sufficient to answer the query (in fact maximal itemsets do not allow us to compute the exact support of smaller itemsets). As shown later in the experiment section, in very large datasets with thousands of items and millions of small transactions our proposal is able to compute frequent k-itemsets while the current state-of-the-art algorithms fail due to huge main memory requirements.

First we show that the problem of finding frequent k-itemsets addressed here can be transformed into the problem of finding frequent symbols over a (huge) stream of symbols over a given alphabet, often referred to as the iceberg queries problem or the hot list analysis. Then, we exploit a recent algorithm for the iceberg queries problem that allows us to solve the original frequent itemset problem by only two sequential passes over the dataset plus a preprocessing step aimed at computing some statistics on the dataset (three passes in total). We will see that, for sparse datasets (i.e., datasets with few items per transactions w.r.t. the total number of possible items) the amount of main memory required by the proposed algorithm is very low and independent of both the number of items and the size of the dataset. This is validated by experiments we have conducted on a prototype implementation of the algorithm. Notice that, when looking for association rules of the form $i_1 \ldots i_n \Rightarrow i_{n+1} \ldots i_k$, with $k > n$, we need to determine frequent itemsets of cardinality n and k. Using standard level-wise algorithms such as *Apriori* [AS94], k passes through the dataset have to be performed. Using our approach, we can run two instances of the proposed algorithm in parallel, thus requiring three passes through the dataset overall. The main contribution of this Section is the development of an algorithm that, for sparse datasets, requires a limited amount of memory while keeping a (small) constant number of passes over the input. Furthermore, we sketch an extension of our algorithm that works over data streams. The contribution here is the development of an algorithm that, with limited memory consumption, is able to mine frequent k-itemsets over a window (a subset of the stream) with size proportional to the length of the stream read so far.

This Section is organized as follows. In Subsection 1.3.2 we set up the notation for iceberg queries, used in the following. In Subsection 1.3.3 we present a new approach to the problem of finding frequent itemsets of size k, based on the reduction to the iceberg queries problem. In Subsection 1.3.4 we present an algorithm that computes the exact set of k-itemsets reading sequentially the dataset only three times, or even two. Subsection 1.3.5 presents some theoretical results on the space complexity of the online frequent itemset problem. Subsection 1.3.6 is devoted to present some experiments we have conducted in order to show the effectiveness of our algorithm in terms of the amount of main memory needed. Subsection 1.3.7 sketches a possible extension of the algorithm to work over data streams. Finally, Subsection 1.3.8 contains some discussions on future works and directions we are planning to follow.

1.3.2 Iceberg Queries

The approach presented here computes frequent itemsets of size k directly (i.e. without computing smaller itemsets), performing two passes though the dataset. Moreover, the amount of main memory needed is known in advance and it is acceptable under the hypothesis that the given dataset is sparse.

Our technique is based on a novel approach to the Iceberg Queries problem, proposed in [KSP03]. The authors present a (surprisingly) simple algorithm able to find all queries with frequency greater than or equal to a given threshold ϑ, from a given stream of queries (i.e., iceberg queries) by using $O(1/\vartheta)$ main memory cells and performing two passes through the stream. Notice that, in the worst-case, the output size is exactly $1/\vartheta$. Furthermore, the algorithm proposed in [KSP03], that we will call *KSP*, does $O(1)$ operations per query (under the reasonable assumption that hash tables make $O(1)$ operations for insertion, search and deletion).

Our results are based on the reduction of the problem of frequent itemsets computation to the problem of finding iceberg queries. Let us define the so called *Iceberg Queries* problem (also known as *Hot List Analysis*).

Alphabet By $\mathcal{Q}$ we denote a finite alphabet. Elements of $\mathcal{Q}$ are denoted by $q, q', q_1, \ldots$ and are referred to as *queries*.

Stream A *stream of queries* is a sequence $s = \langle q_1, \ldots, q_n \rangle$, such that $q_i \in \mathcal{Q}$, for each $1 \leq i \leq n$; the length n of the stream is referred to as $|s|$.

Frequency Given a stream s and a query q, $f_s(q)$ denotes the number of occurrences of q in s.

Definition 3 (IQ Problem). *Let $\mathcal{Q}$ be a set of queries, s be a stream of queries and ϑ be a real number such that $0 < \vartheta \leq 1$. The IQ problem is defined as the task of determining the subset $Q(\vartheta, s)$ defined as follows:*

$$Q(\vartheta, s) = \{q \in \mathcal{Q} \mid f_s(q) > \vartheta|s|\}.$$

If a query q belongs to $Q(\vartheta, s)$ we will say that q is an *iceberg query* with respect to $\mathcal{Q}$, s and ϑ.

Before going on, it is worth pointing out that, in concrete applications of both FIM and IQ problems, the input ($\mathcal{D}$ and s, respectively) is usually huge and it can only be read sequentially (e.g., according to transaction identifiers in the first case and to the sequence order in the second case). Moreover, in FIM problems we usually have $|\mathcal{D}| \gg |\mathcal{I}|$ and in IQ problems we usually have $|s| \gg \mathcal{Q} \gg 1/\vartheta$.

The *KSP* Algorithm

A simple and exact algorithm to solve the IQ problem is described in [KSP03], and we will refer to it as the *KSP*-algorithm (see Algorithm 3). Given a stream s and a real number ϑ (called the *threshold*), the algorithm in [KSP03] requires one pass through the input stream in order to find a superset of the required $Q(\vartheta, s)$. A trivial second pass can be done to find exactly $Q(\vartheta, s)$, keeping the same performance characteristics. In particular, the authors show that their algorithm requires only $O(1/\vartheta)$ memory cells. As shown in Algorithm 3, the only data structures used by *KSP* is a set of queries K and a counter for each query in K.

Example 5. Suppose to have $s = \langle c, b, b, f, g \rangle$ and $\vartheta = 0.4$. This means that we are looking for queries that occur at least 2 times (40% of a stream of 5 queries). At the very beginning of the computation, the set K is empty. We first find c and insert it into K, and $count(c) = 1$. Since $|K| = 1 \not> 1/0.4 = 2.5$ we process the next query, b. Now we have $K = \{b, c\}$, $count(c) = 1$ and $count(b) = 1$. After the third query (another b) we have $K = \{b, c\}$, $count(c) = 1$ and $count(b) = 2$. With the fourth query, f, we first have $K = \{c, b, f\}$. But since $|K| = 3 > 2.5$ then every $count$ has

Algorithm 3 the *KSP*-algorithm

```
Input: $s, \vartheta$
Output: a superset K of $Q(\vartheta, s)$ s.t. $|K| \leq 1/\vartheta$
for all $q \in s$ do
  if $q \notin K$ then
    $K \leftarrow K \cup q$;
    $count(q) \leftarrow 0$;
  $count(q) \leftarrow count(q) + 1$ ;
  if $|K| > 1/\vartheta$ then
    for all $a \in K$ do
      $count(a) \leftarrow count(a) - 1$ ;
      if $count(a) = 0$ then
        $K \leftarrow K \setminus a$;
```

to be decremented: $count(f) = count(c) = 0$ while $count(b) = 1$. Every query in K with count equal to zero must be removed from K, so now we have $K = \{b\}$. By taking into account also the last query, g, we will have $K = \{b, g\}$. In fact, $\{b, g\}$ is a superset of the exact result $\{b\}$ with less than 2.5 elements. Another trivial pass through the stream s will show that $count(b) = 2$ and therefore it can be considered as a valid output while g is not frequent enough since $count(g) = 1$.

1.3.3 Transforming FIM problems into IQ problems

In this section we show how a FIM problem can be transformed into an IQ problem. Roughly speaking, the idea is to associate to each k-itemset a query and to construct a suitable stream $s_{\mathcal{D}}$ of queries starting from the given dataset $\mathcal{D}$, in such a way that the problem of determining $\mathcal{F}_k(\sigma, \mathcal{D})$ is transformed into the problem of determining $Q(\vartheta, s_{\mathcal{D}})$, where ϑ is a function of $\sigma, m_{\mathcal{D}}$ and k. Once we have defined such a transformation, we can adopt any algorithm for the IQ problem in order to solve the original FIM problem. In particular, we can adopt algorithms which keep the number of passes through the dataset as small as possible. In the next section, we will show such an algorithm based on the one proposed by [KSP03] for the IQ problem.

In the sequel, given a finite set S and a natural number k, we denote by S^k the set of all the subsets $P \subseteq S$ such that $|P| = k$. Moreover, given two sequences $s = \langle x_1, \ldots, x_n \rangle$ and $s' = \langle y_1, \ldots, y_m \rangle$, we denote by $s :: s'$ the sequence $\langle x_1, \ldots, x_n, y_1, \ldots, y_m \rangle$.

We first define the transformation which, given a FIM problem, constructs a corresponding IQ problem.

Definition 4 (FIM to IQ). *Let $\mathcal{I}$ be an itemset, $\mathcal{D}$ be a dataset, and k be a natural number such that $k \leq m_{\mathcal{D}}$. Then:*

(i) The alphabet $Q^{\mathcal{I}}$ is defined as the set $\mathcal{I}^k$ (each set in $\mathcal{I}^k$ is a symbol in the alphabet).

(ii) For each $T \in \mathcal{D}$, a stream associated with T is a sequence $s_T = \langle I_1, \ldots, I_{n_T} \rangle$ such that

- $\{I_1, \ldots, I_{n_T}\} = T^k$
- *each $I_j \in T^k$ occurs in s_T exactly once.*

(iii) If $\mathcal{D} = \{T_1, \ldots, T_n\}$, then a stream $s_{\mathcal{D}}$ associated with $\mathcal{D}$ is a sequence

$$s = s_{T_1} :: s_{T_2} :: \ldots :: s_{T_n}.$$

Notice that, in the above definition, we do not define *the* stream associated with a transaction, but rather *a* stream associated with it. Similarly we talk about *a* stream associated with the dataset $\mathcal{D}$. Indeed, given a transaction T_i there are many ways of constructing a stream s_{T_i} corresponding to it, and consequently, there may be many ways of constructing $s_{\mathcal{D}}$. Actually, the choice of $s_{\mathcal{D}}$ is irrelevant as far as the correctness of the transformation is concerned.

In the next theorem we show that a FIM problem $\mathcal{F}_k(\sigma, \mathcal{D})$ can be mapped into an IQ problem $Q(\vartheta, s_{\mathcal{D}})$ where $s_{\mathcal{D}}$ is any stream constructed as in the previous definition and ϑ is a suitable function of σ, k and $m_{\mathcal{D}}$.

Theorem 1. *Let $\mathcal{I}$ be an itemset, $\mathcal{D}$ be a dataset, k be a natural number such that $k \leq m_{\mathcal{D}}$, and σ be a real number such that $0 < \sigma \leq 1$. Let $Q^{\mathcal{I}}$ and $s_{\mathcal{D}}$ be the alphabet and a stream of queries as in Definition 4. Let also $\bar{e} = \binom{m_{\mathcal{D}}}{k}$.*
If an itemset I is a frequent k-itemset with respect to σ and $\mathcal{D}$, then I is an iceberg query with respect to $Q^{\mathcal{I}}$, $s_{\mathcal{D}}$, and $\vartheta = \frac{\sigma}{\bar{e}}$. Formally:

$$I \in \mathcal{F}_k(\sigma, \mathcal{D}) \Longrightarrow I \in Q(\vartheta, s_{\mathcal{D}}).$$

Proof. Let $|\mathcal{D}| = N$ and $\mathcal{D} = \{T_1, \ldots, T_N\}$. We observe:

(1) $|s_{\mathcal{D}}| = \sum_{i=1}^{N} \binom{|T_i|}{k} \leq \sum_{i=1}^{N} \binom{m_{\mathcal{D}}}{k} = \bar{e}N$

(2) By construction of $s_{\mathcal{D}}$, $|\{T \in \mathcal{D} | I \subseteq T\}| = f_{s_{\mathcal{D}}}(I)$.

$$\begin{array}{lll} & I \in \mathcal{F}_k(\sigma, \mathcal{D}) & \\ \Longrightarrow & & \{\text{By definition of } \mathcal{F}_k(\sigma, \mathcal{D})\} \\ & |\{T \in \mathcal{D} | I \subseteq T\}| \geq \sigma N & \\ \Longleftrightarrow & & \{\text{By observation (2)}\} \\ & f_{s_{\mathcal{D}}}(I) \geq \sigma N & \\ \Longleftrightarrow & & \{\text{algebra}\} \\ & f_{s_{\mathcal{D}}}(I) \geq \frac{\sigma}{\bar{e}} N \bar{e} & \\ \Longrightarrow & & \{\text{By observation (1) and definition of } \vartheta\} \\ & f_{s_{\mathcal{D}}}(I) \geq \vartheta |s_{\mathcal{D}}| & \\ \Longrightarrow & & \{\text{by definition of } Q(\vartheta, s_{\mathcal{D}})\} \\ & I \in Q(\vartheta, s_{\mathcal{D}}). & \end{array}$$

□

The previous theorem suggests a new technique for finding frequent k-itemsets, which can be sketched as follows:

(1) Construct a stream $s_{\mathcal{D}}$ corresponding to $\mathcal{D}$;

(2) Find the set of Iceberg Queries $Q(\vartheta, s_{\mathcal{D}})$.

It is worth noting that, in our proposal, step (1) and (2) are not performed sequentially, but rather they are interleaved.

Example 6. Let us give a simple example of the above transformation. Let $\mathcal{I} = \{a, b, c, d, e, f\}$ be the underlying set of items and let $\mathcal{D}$ be the following set of transactions:

$$\mathcal{D} = \{\{a, b, d\}, \{a, c, e\}, \{a, d, f\}, \{b, c\}, \{b, d, e\}, \{c, d, f\}\}$$

Assume that we are interested in 2-itemsets which occur in at least $\sigma = 1/3$ of the transactions. It is easy to see that, among all possible 2-itemsets ($\binom{|\mathcal{I}|}{k} = \binom{6}{2} = 15$), only $\{a, d\}$, $\{b, d\}$ and $\{d, f\}$ appear at least twice.

It is worth noting that anti-monotonicity[2] helps little in this example. In fact, every item (1-itemset) in this example is frequent: it means that 15 candidate 2-itemsets have to be tested by a levelwise algorithm (e.g., Apriori) in a second pass. Even constraint-based techniques (e.g., [BGMP03]) cannot be used here since there are no transactions containing less than 2 items (they would be removed since they do not generate any 2-itemset, possibly reducing the search space) and thus the constraint on the size of the itemsets cannot be exploited.

Let us consider now our *FIM-to-IQ* transformation. For each transaction T_i in $\mathcal{D}$, we build a stream s_i associated with it:

$$\begin{aligned}
s_1 = &\quad \langle\{a, b\}, \{a, d\}, \{b, d\}\rangle \\
s_2 = &\quad \langle\{a, c\}, \{a, e\}, \{c, e\}\rangle \\
s_2 = &\quad \langle\{a, d\}, \{a, f\}, \{d, f\}\rangle \\
s_4 = &\quad \langle\{b, c\}\rangle \\
s_5 = &\quad \langle\{b, d\}, \{b, e\}, \{d, e\}\rangle \\
s_6 = &\quad \langle\{c, d\}, \{c, f\}, \{d, f\}\rangle
\end{aligned}$$

The stream associated with $\mathcal{D}$ is then $s_{\mathcal{D}} = s_1 :: s_2 :: s_3 :: s_4 :: s_5 :: s_6$. Since $m_{\mathcal{D}} = 3$, in this case the threshold for the IQ problem is set to

$$\vartheta = \frac{\sigma}{\binom{m_{\mathcal{D}}}{k}} = \frac{\frac{1}{3}}{\binom{3}{2}} = \frac{1}{9}$$

Notice that in the stream $s_{\mathcal{D}}$, the "queries" $\{a, d\}$, $\{b, d\}$ and $\{d, f\}$ are the only ones occurring with a frequency of $1/8 \geq \vartheta$. In this case the *KSP* algorithm only

[2] The anti-motonicity property of the itemset frequency asserts that if I is frequent, then all the subsets of I have to be frequent.

requires $1/\vartheta = 9$ counters instead of the 15 needed by a levelwise algorithm for frequent itemset mining, and only two passes over the dataset (but we need to know $m_{\mathcal{D}}$ in advance). If $|I| > 10000$ and most of the items are frequent, then the number of candidate 2-itemsets can be too large to be fitted in main memory. In such a case an Apriori-like approach will lead to "out of memory errors", while our transformation approach is still effective (see Section 1.3.6).

1.3.4 The Proposed Algorithm

In this section we propose an instance of the technique based on the result of the previous section, which exploits an algorithm for determining Iceberg Queries recently proposed in [KSP03].

In order to exploit the *KSP*-algorithm, given a dataset $\mathcal{D}$, for each transaction $T \in \mathcal{D}$ we feed the *KSP*-algorithm by each k-itemset contained in T, where k is the given size of itemsets we are interested in. The parameters for the *KSP*-algorithm are set as suggested in Theorem 1, i.e., the threshold is set to $\frac{\sigma}{\bar{e}}$ (recall that $\bar{e} = \binom{m_{\mathcal{D}}}{k}$, where $m_{\mathcal{D}}$ is the maximal length of a transaction in $\mathcal{D}$). In this way we obtain a two-passes algorithm, with the drawback of having to know the value of $m_{\mathcal{D}}$ in advance. If $m_{\mathcal{D}}$ is not known, a "preprocessing pass" is required. Thus, we obtain a three-passes algorithm that saves more memory (w.r.t. the two-pass version) since the length of the stream $s_{\mathcal{D}}$ can be computed exactly in the preprocessing pass, instead of being roughly over-approximated in a pessimistic way by $\binom{m_{\mathcal{D}}}{k}|\mathcal{D}|$.

In the specification of our algorithm (see Algorithm 4), we will use the following notations:

- `KSP.init` sets up the data structures for the *KSP*-algorithm;
- `KSP.threshold` refers to the parameter ϑ of the *KSP*-algorithm;
- `KSP.send(I)` sends the itemset `I` to the *KSP*-algorithm as an element of the stream of queries;
- `KSP.output` denotes the set computed by the *KSP*-algorithm.

Notice that Pass 1 (lines 10–12) of Algorithm 4 basically corresponds to running *KSP* to compute a superset of the desired set of queries (itemsets in our case). The second, trivial pass of our algorithm (lines 15–22) filters out from this superset those itemsets which do not occur at least $\sigma \cdot |\mathcal{D}|$ times in the given dataset. In Pass 2, `count` is a data structure used to actually count the number of exact occurrences of an itemset in the given dataset. Notice that the same data structure used to represent the output of the *KSP*-algorithm (typically an hash table) can be used to implement the `count` data structure as well. This is indeed what we have done in our prototype implementation of the algorithm.

Algorithm 4 Stream Mining Algorithm

```
Input: $\mathcal{D}, \sigma, k$
Output: $\mathcal{F}_k(\sigma, \mathcal{D})$, $count(I)\ \forall I \in \mathcal{F}_k(\sigma, \mathcal{D})$
//Pre-processing pass: some statistics on $\mathcal{D}$ are computed
$streamLength \leftarrow 0$; $dbLength \leftarrow 0$;
for all $T \in \mathcal{D}$ do
    $dbLength \leftarrow dbLength + 1$;
    $streamLength \leftarrow streamLength + \binom{|T|}{k}$;
//Initialize the data structures for KSP
$KSP.threshold \leftarrow \sigma \frac{dbLength}{streamLength}$;
$KSP.init$;
//Pass 1: a superset of $\mathcal{F}_k(\sigma, \mathcal{D})$ is computed
for all $T \in \mathcal{D}$ do
    for all $I \subseteq T$ s.t. $|I| = k$ do
        $KSP.send(I)$;
//assert: KSP.output is a superset of $\mathcal{F}_k(\sigma, \mathcal{D})$
//Pass 2: the actual counts of frequent k-itemsets are obtained
for all $T \in \mathcal{D}$ do
    for all $I \subseteq T$ s.t. $|I| = k$ do
        if $I \in KSP.output$ then
            $count(I) \leftarrow count(I) + 1$;
//infrequent k-itemsets are pruned
for all $I \in KSP.output$ do
    if $count(I) \geq \sigma \cdot dbLength$ then
        $\mathcal{F}_k(\sigma, \mathcal{D}) \leftarrow \mathcal{F}_k(\sigma, \mathcal{D}) \cup I$;
```

The correctness of the algorithm is guaranteed by the correctness of the *KSP*-algorithm and by Theorem 1. As far as space complexity is concerned, the following theorems can be easily proven.

Theorem 2. *Pass 1 of the proposed algorithm computes a superset of $\mathcal{F}_k(\sigma, \mathcal{D})$ in $O(\bar{e}/\sigma)$ space with one sequential pass through the dataset.*

Proof. The data structures used in Pass 1 of the algorithm are those used by *KSP*. The space complexity of the latter, given a threshold ϑ, is $O(1/\theta)$. Since `KSP.threshold` is set to $\sigma/\bar{e}$ the result follows trivially. □

Trivially, Pass 2 of the algorithm computes the actual $\mathcal{F}_k(\sigma, \mathcal{D})$ reading the dataset once, but without requiring further data structures. Hence we have the following corollary.

Corollary 1. *The proposed algorithm computes $\mathcal{F}_k(\sigma, \mathcal{D})$ in $O(\bar{e}/\sigma)$ space with two sequential passes through the dataset.*

1.3.5 Space Complexity of Online Frequent Itemset Mining

In this section we study the space complexity and the number of passes over the dataset needed to compute the set of all frequent itemsets. As we will show, there is a trade-off between the space complexity (memory needed) and the number of passes required. One of the main result is to prove that online (i.e. one pass) frequent itemset mining is not possible in the worst case (under the usual and reasonable assumption that the memory cannot contain all the couple of items). This theoretical result justifies the use of approximated algorithms for stream mining of frequent itemset.

We are going to prove the theorem, we need the following proposition, proved in [KSP03]:

Lemma 1. *Any online algorithm for computing iceberg queries needs* $\Omega(|\mathcal{A}|)$ *space, where* $\mathcal{A}$ *is the alphabet (the set of all queries).*

Then we have:

Theorem 3. *Space complexity of online mining of* σ*-frequent 2-itemsets is* $\Theta(|\mathcal{I}|^2)$*.*

Proof. By contradiction. We prove that if the theorem is false then also online θ-frequent iceberg queries can be obtained in less than $O(|\mathcal{A}|)$, where $|\mathcal{A}|$ is the size of the alphabet, that is false as shown in [KSP03].

The idea is based on datasets composed by exactly 2 items per transaction. The number of different 2-itemsets is

$$\binom{\mathcal{I}}{2} = \frac{|\mathcal{I}| \cdot (|\mathcal{I}| - 1)}{2} \in \Theta(|\mathcal{I}|^2)$$

Seeing the database as a stream of queries where each transaction is a query, then we have a stream of length N in the alphabet of 2-itemsets, of size $\binom{\mathcal{I}}{2}$ Note that, until now, we didn't fix any dataset nor stream in particular, but only the size of the alphabet. Among all possible such streams let us take the worst case stream (therefore, now we are fixing a specific dataset). In this case, according to Lemma 1, we need at least cardinality of the alphabet cells to compute (in one pass) the most frequent queries (i.e. 2-itemsets). □

The following result trivially follows:

Corollary 2. *Space complexity of the exact online* σ*-frequent itemset mining problem is* $\Omega(|\mathcal{I}|^2)$*.*

1.3.6 Experiments

Even if our algorithm performs only two or three passes through the dataset (depending on whether we know $m_{\mathcal{D}}$ in advance or not, where $m_{\mathcal{D}}$ is the maximal transaction length of the dataset), both time and space complexity linearly depend on $\bar{e}$. The first thing to keep in mind is that, for the class of applications of the algorithm we have in mind, accessing the dataset is much more expensive than performing main memory operations, and if $\bar{e}$ is not too big, we can perform main memory operations while reading the dataset. For example, let us imagine a scenario in which a huge dataset is stored in different servers, and communications are encrypted (e.g., in the context of distributed data mining). It is easy to understand that, in this context, reducing the number of passes through the dataset is preferable, even at the cost of adding some more local main memory computations.

We know that $\bar{e} = \binom{m_{\mathcal{D}}}{k}$ is polynomial in $m_{\mathcal{D}}$ (it is $\Theta(n^k)$, where $n = m_{\mathcal{D}}$). This can lead to huge memory requirements, which is obviously not acceptable. The graph in Fig. 1.2 shows the amount of memory needed by our algorithm, when varying both $m_{\mathcal{D}}$ and k (the size of the frequent itemsets we are looking for), and keeping σ fixed to 1/1000 (be aware of the logarithmic scale used for the y axis). These results actually show the minimal requirements for the main hash table structure used by our algorithm. Of course, a real implementation will use some more memory in order to store other runtime data, but since the hash table is the only data structure in our algorithm, we can fairly expect to use such amount of memory by an actual implementation. As we will see later in this section, this is confirmed by experiments with our prototype implementation.

For example, the graph shows that if $m_{\mathcal{D}} = 16$ and $k = 4$ ($\sigma = 1/1000$) then we need less than 128 Mb. Notice that the space requirements are also linearly dependent on $1/\sigma$. Hence, in the previous example, if $\sigma = 1/100$ then we need only ≈ 13 Mb in order to compute the exact set of all frequent 6-itemsets. These results are obtained by using an hash table for *KSP*which requires 40 bytes per cell and a load factor of 0.75 (it is the only data structure used in Algorithm 4).

We believe that these results are quite satisfactory. Notice that they *do not depend on the size of the dataset or the number of possible items* (except for a necessary logarithmic factor due to the number of bits required for the counters) but only on σ, $m_{\mathcal{D}}$ and k.

Furthermore, the proposed algorithm scales polynomially in the size of the maximal transaction. If the dataset is sufficiently sparse it is possible to divide the transactions into two groups: a small group with the large transactions, and a large group where the proposed algorithm can be applied, containing short transactions. If the former group is small enough, the parameters of the iceberg algorithm would only need to be modified slightly, and the filtering pass would use the original parameter to select the actual frequent itemsets.

We also carried on a set of experiments in order to compare the scalability of our algorithm with the state-of-the-art algorithms FPGrowth [HPY00], Eclat [Zak00], Relim [Bor05] and the well-known Apriori [AS94]. We generated an artificial Retail-like dataset, by replicating the Retail dataset (from the FIMI Repository [GZ03]) and by inserting 2 different items in each transaction, with the following characteristics:

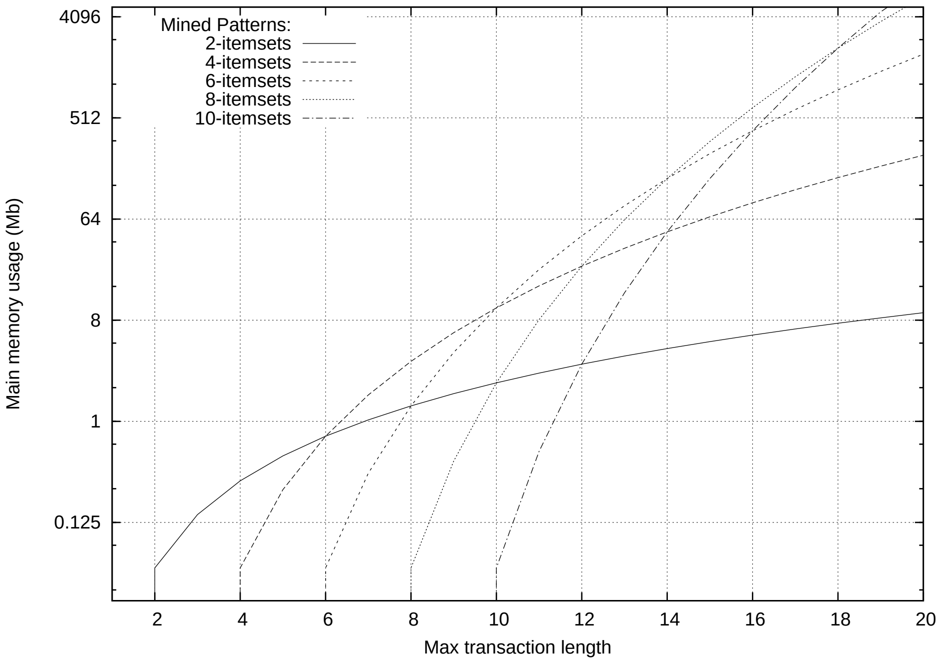


Figure 1.2: Memory usage is a function of m_d and k (in this graph σ is fixed to 1/1000)

Number of Transactions	12,497,500
Number of Items	16,470
Size of Largest Transaction	78
Size of Dataset	722Mb

Then, we truncated the dataset in order to limit the number of transactions at 1, 2, 3, 4 and 5 million. Results on main memory usage are showed in Fig. 1.3. On a 512Mb-Ram Pentium machine, Relim was able to find 2-frequent itemsets up to 3 million of transaction before crashing, while Apriori, FPGrowth and Eclat went up to 4 million. Only our three-passes algorithm was able to compute all the 2-frequent itemsets in the 5 million transaction dataset.

Memory requirements of almost all algorithms grow linearly with the size of the dataset. This is a big problem for huge datasets, as the one we used. Even the Apriori algorithm, whose space complexity doesn't really depend on the number of transactions, appears to grow linearly. This happens because in our dataset new transactions often contain new (i.e., not seen so far) pairs of items (while the number of different items is fixed). Anti monotonicity is not very exploitable since most of items are frequent, so almost every 2-itemset is a valid candidate. Our 3-pass algorithm shows a completely different behavior, actually being the only scalable algorithm with constant memory consumption.

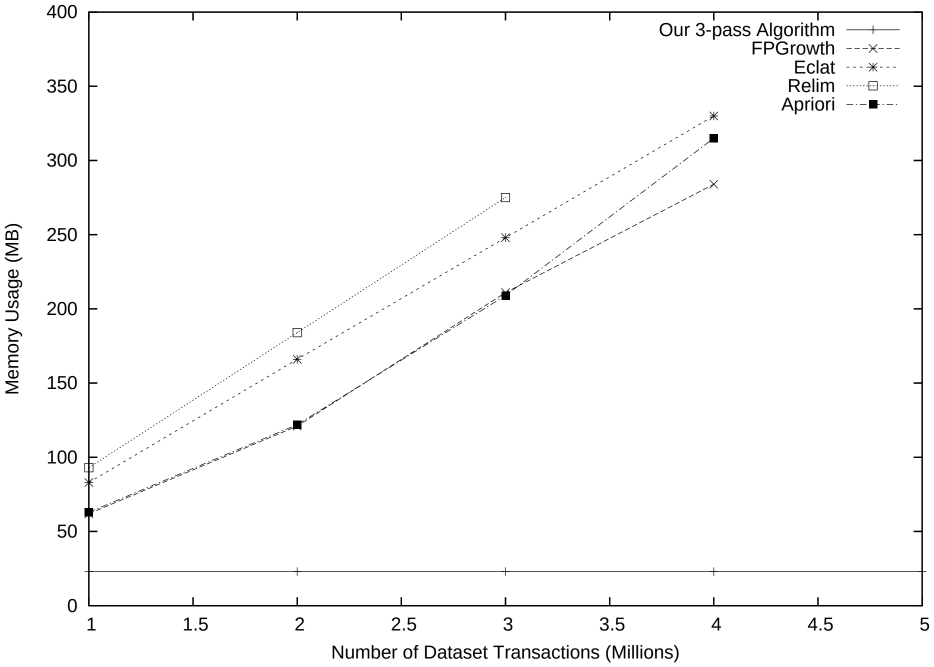


Figure 1.3: Memory actually used by FIM Algorithms ($\sigma = 1/100$)

With respect to time performance, our algorithm, in the current unoptimized Java implementation, is slower than the other algorithms (all implemented independently in C), but it comes out to be very scalable, by requiring 21 minutes for the 1 million dataset, and 41 minutes for the 2 millions dataset. We also noted that in all FIM algorithms we tested running time are very related with memory used and available.

1.3.7 Frequent k-Itemset Mining over Streams

In this section we describe how to exploit the above results in the case of *Stream Mining*. We assume to have a stream of transactions and we want to compute k-itemset that appear at least σ% times over the stream. Once we access a transaction, we cannot access to the previous one; in other words, the stream can be read only once, sequentially. Every problem over streams can be trivially reduced to the non-stream version by assuming to have enough memory to store the whole stream. Unfortunately, this assumption is not acceptable in most of the practical stream problems, it is rather impossible in some cases, since the stream is potentially infinitely long. Therefore, in the case of frequent itemset mining over streams, only online (i.e., 1-pass) algorithms are acceptable. Another common requirement in data stream algorithms is the ability to show anytime partial results (computed over the part of stream analyzed so far).

In Section 1.3.4 we showed (Corollary 2) that space complexity of exact online σ-frequent itemset mining problems is $\Omega(|\mathcal{I}|^2)$. As we mentioned, this means that only approximated algorithms are good for online σ-frequent itemset mining problems. The last part of the stream is usually more interesting since it contains the most recent trends in the stream itself. Many online algorithms are based on sliding windows, i.e., they compute the frequent (k-)itemsets over the last l transactions read so far (i.e., windows of size l), where l is a constant that depends on the memory available.

The following is an adaptation of our Algorithm 4 presented in Section 1.3.4 that is able to mine frequent k-itemsets online (over a stream), with the guarantee to compute the results considering a window with size proportional to the length of the stream read so far (not only a window of constant size).

Algorithm 5 Online Stream Mining Algorithm

```
Input: $\S, \sigma, k, m_\mathcal{D}$
Output: $\mathcal{F}_k(\sigma, \S)$, $count(I)\ \forall I \in \mathcal{F}_k(\sigma, \mathcal{D})$
KSP.threshold ← $\frac{\sigma}{\binom{m_\mathcal{D}}{k}}$;
KSP.init;
current ← 1;
window_size ← 1;
while true do
    for $i$ : 1 to window_size do
        use §[current] with our Pass1;
        current ← current + 1;
    for $i$ : 1 to window_size do
        use §[current] with our Pass2;
        current ← current + 1;
    window_size ← 2 · window_size;
```

Our *Online Stream Mining Algorithm* (Algorithm 5) alternates *Pass 1* and *Pass 2* of Algorithm 5 over exponential growing windows. First, a window of size 1 (i.e., only one transaction is considered) is used, applying Pass 1. Then the second transaction is read, applying Pass 2. Now the window size is doubled, and Pass 1 is applied to the next two transactions (transactions 3 and 4), then Pass 2 to transactions 5 and 6.

1.3.8 Conclusions and Future Work

We presented an algorithm to compute the exact set of frequent k-itemsets (and their counts) on a given dataset, with reasonable space requirements under the hypothesis that the dataset is sparse. The algorithm is obtained by transforming the set of transactions into streams of k-itemsets and by using an algorithm for iceberg queries in order to compute the output. As far as space complexity is concerned, the algorithm requires an acceptable amount of main memory even for huge datasets. We have actually implemented a first, non-optimized version of the algorithm, i.e., without making use of specific (e.g., compressed) data structures. Some experiments on concrete (sparse) datasets are encouraging also as far as time complexity is concerned, showing the effectiveness of the algorithm.

We plan to implement an optimized version of our algorithm that is able to compute all the frequent itemsets with only two passes (having the maximum transaction length in input), running some instances of the current version of the algorithm for different values of k, but saving memory because of the redundancies among different instances (e.g., if the 3-itemset $\{(a, b, c)\}$ is frequent we do not need to store that also the 2-itemsets $\{(a, b), (a, c), (b, c)\}$ are frequent).

Moreover, we plan to study hybrid algorithms which exploit our approach for the first few steps of a level-wise algorithm à la *Apriori*. For example, we can determine in advance suitable bounds for k which ensure that the amount of memory required by our algorithm keeps reasonable in computing x-itemsets, for each level $x \leq k$. Then, we can continue the generation of frequent h-itemsets, for $h > k$, by using a standard level-wise approach. Notice that, for each $x < k$ we can even adopt a simpler version of our algorithm, which performs only the first pass through the dataset (i.e., computing supersets of frequent x-itemsets).

Finally, from the theoretical point of view, we believe that the FIM to IQ transformation presented here can also be useful to get theoretical results on the trade-off between memory requirements and number of passes over the dataset in the FIM problem.

Chapter 2

Background on Abduction

Abstract

In this Chapter we introduce the field of abductive reasoning to the reader and review some systems developed using abductive logic. Cost-based abduction, the abductive reasoning framework used in this thesis, is also presented. Then, we give a description of two known systems that integrate abduction and induction.

2.1 Abductive Reasoning

2.1.1 Some forms of Reasoning

The philosopher Peirce first introduced the notion of abduction. In [Pei] he identified three distinguished forms of reasoning:

- Deduction, an analytic process based on the application of general rules to particular cases, with the inference of a result;
- Induction, synthetic reasoning which infers the rule from the case and the result;
- Abduction, another form of synthetic inference, but of the case from a rule and a result.

Example 1. Consider the Aristotelian syllogism Barbara[1]:

> All the beans from this bag are white. *(rule)*
> These beans are from this bag. *(case)*
> Therefore, these beans are white. *(result)*

[1] The reason for this name is that the word *Barbara* sounds like *Pa-Pa-Pa*, which means we have 3 (universal) affirmative predicates [Fab73].

By exchanging the result with the rule, one obtains a (deductively invalid) syllogism that can be seen as an inductive generalization:

> These beans are white. *(result)*
> These beans are from this bag. *(case)*
> Therefore, all the beans from this bag are white. *(rule)*

Alternatively by exchanging the result with the case we obtain an abductive syllogism:

> All the beans from this bag are white. *(rule)*
> These beans are white. *(result)*
> Therefore, these beans are from this bag. *(case)*

Here, the conclusion can be seen as an explanation of the result given the rule.

2.1.2 Abduction

As already said, abduction is a particular form of reasoning introduced by the philosopher Peirce [Pei] as a form of synthetic reasoning that infers the case from a rule and a result. In its simplest form, abduction can be described by the following rule:

$$\frac{B, \quad A \Rightarrow B}{A}$$

i.e., from the fact that A implies B and the observation B, we can abduce A. Abduction can be viewed as the probational adoption of an hypothesis as an explanation for observed facts, according to known laws. Obviously, as Peirce noticed, “it is a weak kind of inference, because we cannot say that we believe in the truth of the explanation, but only that it may be true”.

Let consider the following example:

Example 2. Consider the following theory T:

> *rained-last-night* → *grass-is-wet*
> *sprinkler-was-on* → *grass-is-wet*
> *grass-is-wet* → *shoes-are-wet*

If we observe that our shoes are wet, and we want to know why this is so, *rained-last-night* is a possible explanation, i.e., a set of hypotheses that together with the explicit knowledge in T implies the given observation. *sprinkler-was-on* is another alternative explanation.

Abduction consists of computing such explanations for observations. It belongs to non-monotonic reasoning, because explanations with one state of a knowledge base may become inconsistent with new information. In the example above the explanation *rained-last-night* may turn out to be false, and the alternative explanation *sprinkler-was-on* may be the true cause for the given observation. Abductive reasoning is characterized by the existence of multiple explanations, and the selection of a preferred explanation is an important related problem.

In recent years, abduction has been widely studied and adopted in the context of logic programming [KKT93], starting from the work [EK89]. In this thesis, we basically adopt the definitions and terminologies of [KM90, KMM00], that we are going to briefly review next.

2.1.3 Abductive Logic Programming

An Abductive Logic Programming (ALP) framework is defined as a triple $\langle P, A, Ic\rangle$ consisting of a logic program, P, a set of ground abducible atoms A and a set of logic formulas Ic, called the integrity constraints. The atoms in A are the possible abductive hypotheses that can be assumed in order to explain a given observation in the context of P, provided that these assumptions are consistent with the integrity constraints in Ic. In many cases, it is convenient to define A as a set of (abducible) predicate symbols, with the intended meaning that each ground atom whose predicate symbol is in A is a possible hypothesis.

Given an abductive framework as before, the general definition of abductive explanation is given as follows.

Definition 1. *Let $\langle P, A, Ic\rangle$ be an abductive framework and let G be a goal. Then an* abductive explanation *for G is a set $\Delta \subseteq A$ of ground abducible atoms such that:*

- $P \cup \Delta \models G$
- $P \cup \Delta \cup Ic$ *is consistent.*

Example 3. Let $\langle P, A, Ic\rangle$ be an abductive framework where:

- P is the logic program given by the three rules: $p \leftarrow a \qquad p \leftarrow b \qquad q \leftarrow c$
- $A = \{a, b, c\}$
- $Ic = \{\}$.

Then, there are three abductive explanations for the goal[2] (p, q) given by $\Delta_1 = \{a, c\}$, $\Delta_2 = \{b, c\}$ and $\Delta_3 = \{a, b, c\}$.

Consider now the abductive framework $\langle P, A, Ic'\rangle$, where P and A are as before and Ic contain the formula $\neg(a, c)$. In this new framework, there is only one explanation for the goal p, q, which is Δ_2 above, since the second potential explanation is inconsistent with the integrity constraint.

[2] As in standard logic programming, "," in rule bodies and goals denotes logical conjunction.

The given notion of abductive explanation can be easily generalized to the notion of abductive explanation given an initial set of abducibles Δ_0.

Definition 2. *Let $\langle P, Ab, Ic\rangle$ be an abductive framework, Δ_0 be a set of abducibles and G be a goal. We say that Δ is an abductive explanation for G* given Δ_0 *if $\Delta_0 \cup \Delta$ is an abductive explanation for G.*

Notice that this implies that the given set of abducibles Δ_0 must be consistent with the integrity constraints Ic. In some cases, we may be interested in *minimality* of abductive explanations.

Definition 3. *Let $\langle P, Ab, Ic\rangle$ be an abductive framework, Δ_0 be a set of abducibles and G be a goal. We say that Δ is a Δ_0-minimal explanation for G if Δ is an explanation for G given Δ_0 and for no proper subset Δ' of Δ ($\Delta' \subset \Delta$), Δ' is an explanation for G given Δ_0.*

For instance, in Example 3 with $Ic = \{\}$, Δ_0 is the empty set (we do not have an initial set of "pre-abduced" abducibles) and both Δ_1 and Δ_2 are Δ_0-minimal explanations for (p, q) while Δ_3 is not (in fact, $\Delta_1 \subset \Delta_3$).

The notion of abduction as described above has a natural computational counterpart within a proper extension of the standard SLD-based computational engine of logic programming. In particular, when integrity constraints are in the form of *denials*, i.e., in the form $\neg(p_1, \ldots, p_n)$, the KM-proof procedure can be successfully adopted to compute (minimal) abductive explanations. The details of the definition of the KM-proof procedure can be found in [KM90, KMM00].We just mention that the procedure is an interleaving of two phases: the *abductive* phase and the *consistency* phase. In the abductive *phase* the procedure basically performs SLD-resolution steps and collects the abductive hypotheses required to explain the given original goal. Whenever a new hypothesis is selected for computation, the procedure checks its consistency against the relevant integrity constraints[3], hence opening a new consistency phase. In the general case, it is easy to see that the consistency phase may require in turn to open a new abductive phase and so on.

As we already said in the Introduction, our goal is to show that abduction can be profitably used for data mining purposes. As far as we know, although both data mining and abductive reasoning have been independently explored through several works (most of them belonging to the field of Artificial Intelligence), this is the first attempt to join them together.

Some related works came from three workshops on abduction and induction, contained in a book [FK00]. None of them talks about data mining; anyway, they take into account links between induction (sometimes referred to Machine Learning) and abduction [LMMR99, KR00].

[3] The KM-proof procedure assumes that at least one abducible occurs in each integrity constraint.

We briefly survey some general-purpose Abductive Frameworks useful to solve abductive queries, and then we present some known systems where both abduction and induction are involved in the computation.

2.2 Abductive Frameworks

2.2.1 ACLP, Abductive Constraint Logic Programming

ACLP is a system which combines abductive reasoning and constraint solving by integrating the framework of Abductive Logic Programming (ALP) with that of Constraint Logic Programming (CLP). The ACLP framework grew as an attempt to address the problem of providing a high level declarative programming or modeling environment for problems of Artificial Intelligence which at the same time has an acceptable computational performance [KMM00].

In ACLP the role of abductive reasoning is that of providing an automatic reduction of the high level problem representation and goals to lower level computational tasks of a general problem independent form. The use of Constraint Logic Programming techniques is fundamental to enhance the efficiency of the computational process of abductive inference as this is applied on the high level representation of the problem at hand. It thus provides a bridge between the high level problem domain properties and domain-independent solving methods.

The general computation model of ACLP consists of a cooperative interleaving between hypotheses and constraint generation, via abductive inference, with consistency checking of abducible assumptions and constraint satisfaction of the generated constraints. The integration of abductive reasoning with constraint solving in ACLP is cooperative, in the sense that the constraint solver not only solves the final constraint store generated by the abductive reduction but also affects dynamically this abductive search for a solution. It enables abductive reductions to be pruned early by setting new suitable CLP constraints on the abductive solution that is constructed.

Currently, the system is implemented as a meta-interpreter on top of the CLP language of ECL^iPS^e using the CLP constraint solver of ECL^iPS^e to handle constraints over finite domains (integer and atomic elements). The architecture of the system is quite general and can be implemented in a similar way with other constraint solvers.

ACLP has been applied to several different types of problems. Initial applications have concentrated on the problems of scheduling, time tabling and planning. Other applications include optical music recognition where ACLP was used to implement a system that can handle recognition under incomplete information, resolving inconsistencies in software requirements where (a simplified form of) ACLP was used to identify the causes of inconsistency and suggest changes that can restore consistency of the specification and intelligent information integration where ACLP has been used as a basic framework in the development of information mediators for the semantic integration of information over web page sources. Although most of these applications are not of "industrial scale" (with the notable exception of a crew-scheduling application for the small sized company Cyprus Airways) they have been helpful in indicating some general methodological guidelines that can be followed when one is developing abductive applications [KM99]. The aircrew scheduling application produced solutions that were judged to be of good quality, comparable to manually generated solutions by people with many years of experience on the particular problem, while at the same time it provided a flexible platform on which the company could easily experiment with changes in policy and preferences.

2.2.2 $\mathcal{A}$-System

Despite a large number of applications for abduction and its potential benefits, general abductive logic frameworks usually focus on the importance of representation, while in general they do not deeply address the problems of computationally effectiveness for problems of practical size and complexity (see [BATJ91] for a theoretical study on the complexity of abductive frameworks).

The $\mathcal{A}$-System [AVNDB01, VNK01, KVND01] is a prolog-based abductive framework that aims to investigate scalability issues. It is obtained mainly merging the ideas coming from previous abductive procedures such as SLDNFA [DS98], IFF [FK97] and ACLP [KMM00],

Basically, it formulates the inference rules of SLDNFA in the form of state rewrite rules as done by the IFF procedure. The original rules of SLDNFA are extended with rules that handle (finite domain) constraint expressions as proposed by ACLP. Internally, at the level of implementation, the $\mathcal{A}$-System uses a similar scheme as finite domain constraint solvers: it will evaluate all deterministic information before it makes a non-deterministic choice. When all non-deterministic choices are successfully evaluated and all the constraint stores are consistent, a solution is found.

The constraint stores that are constructed during a run, participate actively in the search, i.e., the current state of a constraint store may invoke new derivations or backtracking. This is achieved by implementing for each constraint domain, that are equational reasoning on Herbrand terms and finite domain constraint expressions, handles that monitor the state of an expression w.r.t. the current constraint store. These handles take in the $\mathcal{A}$-System the form of reification. Since Sicstus Prolog (nor any other Prolog system) comes with a predefined module for equational reasoning on Herbrand terms, the $\mathcal{A}$-System comes with a novel equality solver for Herbrand terms which supports reification. Also the $\mathcal{A}$-System contains a layer on top of the finite domain constraint solver that will provide a similar functionality for general finite domain expressions.

2.2.3 PHA, Probabilistic Horn Abduction

Probabilistic Horn Abduction is a pragmatic combination of logic and probability proposed by David Poole [Poo93]. The general idea is to have a set of independent choices with a logic program that tells the consequences of choices. The logic is weak: it does not contain disjunction (all of the "uncertainty" is embodied in the choices), and it does not impose any probabilistic dependencies on the choices.

It is a simple framework for Horn-clause abduction, with probabilities associated with hypotheses. The framework incorporates assumptions about the rule base and independence assumptions amongst hypotheses. It is possible to show that any probabilistic knowledge representable in a discrete Bayesian belief network can be represented in this framework. The main contribution is in finding a relationship between logical and probabilistic notions of evidential reasoning. This provides a useful representation language in its own right, providing a compromise between heuristic and epistemic adequacy. It also shows how Bayesian networks can be extended beyond a propositional language and how a language with only (unconditionally) independent hypotheses can represent any probabilistic knowledge, and argues that it is better to invent new hypotheses to explain dependence rather than having to worry about dependence in the language (see Chap. 6.11 in [Mit97] for further information on Bayesian Belief Networks) .

In this framework we can define an abductive scheme as a pair $\langle F, H \rangle$ where:

F is a set of Horn clauses. Variables in F are implicitly universally quantified. Let F' be the set of ground instances of elements of F;

H is a set of (possibly open) atoms, called *assumables* or *possible hypothesis.* Let H' be the set of ground instances of elements of H.

The system makes the following assumption on the base theory:

1. There are no rules in F' whose head unifies with a member of H.

2. If F' is the set of ground instances of elements of F , it is possible to assign a natural number to every ground atom such that for every rule in F' the atoms in the body of the rule are strictly less than the atom in the head (i.e., F is an acyclic theory).

3. For every non-assumable a, if a is true then at least one of the bodies B_i such that $a \leftarrow B_i$ has to be true. We say that the rules in F' for every ground (non-hypothesis) represented atom are covering.

4. The bodies of the rules in F' for an atom are mutually exclusive.

Intuitively the logic program gives the consequences of the choices. This framework is abductive in the sense that the explanations of an observation g provide a concise description of the worlds in which g is true. Belief networks can be defined by having independent probability over the alternatives.

The PHA system does not explicitly handle integrity constraints. Its implementation is based on Prolog. It is implemented by a branch and bound search where the partial explanation with the least cost (highest probability) is considered at any time.

The implementation keeps a priority queue of sets of hypotheses that could be extended into explanations ("partial explanations"). At any time the set of all the explanations is the set of already generated explanations, plus those explanations that can be generated from the partial explanations in the priority queue.

2.2.4 Cost-based Abduction

Cost-based abduction (CBA) [HSAM93, Sti91] is an important problem in reasoning under uncertainty. It attempts to find the best explanation for a set of observed facts by finding a minimal cost proof for such facts. Sometimes referred to as *weighted abduction*, it has been shown to be very useful in the context of natural language interpretation [HSAM93, HMP$^+$00].

As in the other work on Abduction, the aim in cost-based abduction is to find from data describing observations or events, a set of hypotheses which best explains or accounts for the data.

Formally, a cost-based abduction framework is a knowledge representation of the domain of interest modeled as a 4-tuple

$$L = (\mathcal{H}, \mathcal{R}, c, \mathcal{G})$$

where

- $\mathcal{H}$ is a set of hypotheses or propositions;
- $\mathcal{R}$ is a set of rules of the form

  $$(h_1 \wedge h_2 \wedge \cdots \wedge h_{n_i}) \rightarrow h_{n_i+1}$$

  where $h_1, \ldots, h_{n_i+1} \in \mathcal{H}$;
- c is a function $c : \mathcal{H} \rightarrow \mathcal{R}^+$, where $c(h)$ is called the assumability cost of the hypothesis $h \in \mathcal{H}$;
- $\mathcal{G} \in \mathcal{H}$ is the goal set or the evidence that we would like to explain.

The final aim is to find the least cost proof for the evidence, where the cost of a proof is taken to be the sum of the costs of all hypotheses that must be assumed in order to complete the proof. The hypothesis can be made true in two ways:

1. it can be assumed to be true, by "paying" its assumability cost;

2. it can be proved.

Note that if an hypothesis occurs as the consequent of a rule R, then it can be proved, at no cost, to be true by making all the antecedents of R true, either by assumption or by proof. If an hypothesis does not appear as the consequent of any rule, then it cannot be proved, it can be made true only by being assumed. If the cost of an hypothesis is large enough, then it cannot be assumed, it can only be proved. Usually the set of hypothesis $\mathcal{H}$ is partitioned into two sets $\mathcal{H}^A$ and $\mathcal{H}^P$: the former contains hypothesis that cannot be proved (they never occur in the right-hand side of a rule); the latter contains all the other hypothesis.

As usual in abductive reasoning, a set A is a feasible solution for the least cost proof problem if it is a subset $\mathcal{H}^A$ which is sufficient to prove the goal set $\mathcal{G}$. An (optimal) solution to the least cost proof problem problem is a feasible solution that minimizes the total cost, i.e., the sum of the costs of hypothesis in A.

Efficiency of Cost-Based Abduction

Complexity. Although the general problem of cost-based abduction has been shown to be intractable in the worst case [CS94, Abd04, EG95], different approaches have been explored in the past decade. In [CS94, CH91] two different heuristics for the problem are described. A way to find abductive proofs of minimal cost by using 0-1 integer linear programming is given in [Jr.94], based on a transformation of the original problem into a linear constraint satisfaction problem.

Under some circumstances, cost-based abduction can be polynomially-solvable. In [SS96] a set of sufficient conditions for polynomial solvability of cost-based abduction is discussed, later extended in [OY97].

Other linear programming based approaches able to find near-optimal solutions are described in [IM02, MI02, OI97].

A parallel implementation of a cost-based abductive system can be found in [KSI97].

Decidability of the inference procedure. Prolog-style backward chaining, with an added factoring operation and without the literal ordering restriction (so that any, not just the leftmost, literal of a clause can be resolved on), is capable of generating all possible explanations that are consistent with the knowledge base. That is, every possible explanation consistent with the knowledge base is subsumed by an explanation that is generable by backward chaining and factoring.

It would be desirable if the procedure were guaranteed to generate no explanations that are inconsistent with the knowledge base. However, this is impossible; consistency of explanations with the knowledge base must be checked outside the abductive reasoning inference system. Note that not all inconsistent explanations are generated: the system can generate only those explanations that assume literals that can be reached from the initial formula by backward chaining. Therefore, determining consistency is undecidable in general, though decidable subcases do exist, and many explanations can be rejected quickly for being inconsistent with the knowledge base. For example, assumptions can be readily rejected if they violate sort or ordering restrictions, e.g., assuming $woman(John)$ can be disallowed if $man(John)$ is known or already assumed, and assuming $a > b$ can be disallowed if $a \leq b$ is known or already assumed. Sort restrictions are particularly effective in eliminating inconsistent explanations in natural language interpretation.

2.3 Integration between Abduction and Induction

Several frameworks in the context of logic programming have been already designed in order to integrate induction and abduction [FK00], although some of them have not been implemented. Two important systems (implemented by their authors) that rely on abductive and inductive techniques are briefly described next.

2.3.1 DemoII

DemoII is a meta-logic programming system developed by Henning Christiansen and Davide Martinenghi (Roskilde University, Denmark) [Chr00]. It covers a wide range of automated reasoning tasks including abduction and induction. These tasks are specified in terms of declarative meta-level statements using the proof predicate `demo`, which can be specified as follows.

`demo(P',Q')` iff `P'` and `Q'` are names of object program `P` and query `Q` and
there exists a substitution with $\texttt{P} \vdash \texttt{Q}$

DemoII provides an implementation of `demo` that is reversible in the sense that it works equally well for executing known programs as well as for generating new programs that make given goals provable.

Specifications may include also syntactic bias and integrity constraints. The system is a generic environment for experiments with different reasoning methods. Distinct for the system is the ease with which different methods can be combined, e.g., integrating abduction and induction with default rules.

These properties are obtained using constraint logic techniques implemented with the Constraint Handling Rules library included in SICStus Prolog 3.7, under which the system runs.

2.3.2 ACL, Abductive Concept Learning

ACL (developed by Antonis Kakas and Fabrizio Riguzzi) learns abductive theories [KR00]. The input of the system consists of an abductive theory $T = \langle P, A, I \rangle$ as background knowledge and two sets of ground atoms as positive and negative examples. ACL finds an abductive theory $T' = \langle P', A, I' \rangle$ with $P' \supset P$ and $I' \supset I$

such that $T' \models_A E^+$ and $\forall e^- \in E^-, T' \not\models_A e^-$ where E^+ stands for the conjunction of all positive examples (E^- stands for the disjunction of the negative ones).

Learning abductive theories allows to learn from incomplete knowledge, since abducible predicates can be used in order to represent incomplete knowledge.

The algorithm learns first all the rules and then all the constraints. Rule learning is performed by a top-down ILP algorithm where coverage testing is performed using an abductive proof procedure. The algorithm adopts a beam search strategy and a special heuristic function that gives different weights to examples covered with or without abduction. The output of the rule learning phase is a set of rules and a set of explanations: one explanation (possibly empty) for each positive example and all possible explanations (of which none is empty) for each negative examples. The requirement that no explanation for negative examples is empty is needed in order to be able to exclude all of them in the second phase by learning constraints.

Constraints are learned using ICL. The input to ICL consists of the abductive explanations for positive examples as positive interpretations and the abductive explanations for negative examples as negative interpretations. Rules previously learned together with those from the background knowledge constitute the background knowledge of ICL.

2.4 Collocation of Our Approach

To the best of our knowledge, our framework described in Chapter 4 is the first attempt to apply Abductive-based techniques to improve performaces of data mining algorithms. As we will see, it is developed on top of (cost-based) abductive frameworks, extracting and integrating intensional knowledge induced from row data. The most important difference of our proposal with the hybrid learner systems reviewed so far in this Chapter is about the purpose of the system. ACL, the most related abductive framework, is aimed at discovering new abductive theories or integrity constraints (i.e., a sort of induction) through inductive logic programming, given an initial knowledge. Our framework instead exploits frequent-itemset-based data mining algorithms in order to abduce new facts (unknown extentional knowledge), such as missing values. These abducte facts are then used to improve previous data mining results such as instance classification.

Chapter 3

Background on Privacy-Preserving Data Mining

Abstract

In this Chapter we introduce the field of privacy-preserving data mining and describe two possible scenarios arising in the context of data mining when private data is analyzed. Then we review the work in literature on privacy-preserving data mining.

A recent challenge for the data mining community is to design systems that protect the privacy and ownership of individual data without blocking the information flow. Since the primary task in data mining is the development of models about aggregated data, there is the need to develop accurate models which prevent the access to precise information in individual data records. Future data mining systems must include responsibility for the privacy of data they manage as a founding tenet; this is at same time an ethical goal and a prerequisite for social acceptance and broad deployment of data mining applications.

Privacy-preserving data mining has currently three main aims and their corresponding research directions:

- the definition of useful privacy constraints that the output of data mining algorithms has to respect [Mie03, AKSX02, ABE+99];
- the development of data mining techniques that enforce privacy constraints [AA01, ESAG02];
- the development of distributed algorithms able to compute data mining models without communicating (or hiding) private data [CKV+02b, KC02, DVEB01, Pin02].

Privacy-preservation is a young subfield of data mining; although continuously growing, only a few important results are currently available.

3.1 On the Definition of Privacy

Improving trust in the knowledge society is a key requirement for its development. Privacy-awareness, if addressed at a technical level and acknowledged by regulations and social norms, may foster social acceptance and dissemination of new emerging knowledge-based applications. This is true of data mining, which is aimed at learning patterns, models and trends that hold across a collection of data. While the potential benefits of data mining are clear, it is also clear that the analysis of personal sensitive data arouses concerns about citizen's privacy, confidentiality and freedom. Obtaining the potential benefits of data mining with a privacy-aware technology would enable a wider social acceptance of a multitude of new services and applications based on the knowledge discovery process. Source data of particular importance include, for instance, bio-medical patient data, web usage log data, mobility data from wireless networks: in all such cases the potential privacy threats are evident, as well as the potential usefulness of knowledge discovered from these data.

The awareness that privacy protection in data mining is a crucial issue has driven the attention of many researchers in the last few years, and consequently *privacy-preserving data mining*, i.e., the study of data mining side-effects on privacy, has rapidly become a hot and lively research area, which receives an increasing attention from the research community [VBF$^+$04].

The generic problem definition is how to produce valid mining models without disclosing "private" information. However, despite such efforts, we agree with [CKV02a] that a common understanding of what is meant by "privacy" is still missing. As a consequence, there is a proliferation of many completely different approaches of privacy-preserving data mining: some aim at *individual privacy*, i.e., the protection of sensitive individual data, while others aim at *corporate privacy*, i.e., the protection of strategic information at organization level; the latter is more a secrecy, rather than privacy, issue. To make the scene more complex, often the need to guarantee individual privacy leads an organization to adopt corporate privacy policies.

3.2 Killer Applications

Here we describe two killing application in the context of privacy-preserving data mining that can help to understand the impact of having mining tools that are privacy-aware.

Example 1 (Medical Knowledge Discovery).

Medical informatics has become an integral part of successful medical institution. Many modern hospitals and health care institutions are now well equipped with monitoring and other data-collection devices, and data is gathered and shared in inter – and intra – hospital information systems. This growth of the volume of medical data available based the right premises for the birth of *Medical Data Mining*, i.e., the application of data analysis techniques to extract useful knowledge for supporting decision making in medicine [KC01].

Because medical data are collected and used by human subjects, there is a large ethical and legal tradition designed to prevent the abuse and misuse of medical data. This legal tradition and the fear of lawsuits directed against the physicians and the health-care institutions, could represent a tough obstacle for the acceptance of Medical Data Mining.

The strength of the privacy requirements together with the incredible benefits that the whole society can achieve from it, make Medical Data Mining a challenging and unique research field; while the kind of privacy required (i.e., the anonymity of the patients in a survey), make it a perfect prototypical application instance for privacy-preserving data mining frameworks.

In concrete, consider a medical institution where the usual hospital activity is coupled with medical research activity. Since physicians are the data collectors and holders, and they already know everything about their patients, they have unrestricted access to the collected information. Therefore, they can perform real mining on all available information using traditional mining tools – not necessarily the privacy-preserving ones. This way they maximize the outcome of the knowledge discovery process, without any concern about privacy of the patients which are recorded in the data. But the anonymity of individuals patients becomes a key issue when the physicians want to share their discoveries (e.g., association rules holding in the data) with their scientific community.

Example 2 (Geographic Knowledge Discovery).

Spatio-temporal, geo-referenced datasets are growing rapidly, and will be more in the near future. This phenomenon is mostly due to the daily collection of telecommunication data from mobile phones and other location-aware devices. The increasing availability of these forms of geo-referenced information is expected to enable novel classes of applications, where the discovery of consumable, concise, and applicable knowledge is the key step. As a distinguishing example, the presence of a large number of location-aware wirelessly connected mobile devices presents a growing possibility to access space-time trajectories of these personal devices and their human companions: trajectories are indeed the traces of moving objects and individuals. These mobile trajectories contain detailed information about personal and vehicular mobile behavior, and therefore offer interesting practical opportunities to find behavioral patterns, to be used for instance in traffic and sustainable mobility management, e.g., to study the accessibility to services. Clearly, in these applications privacy is a concern. As an instance, an over-specific pattern may reveal the behavior of an individual (or a small group of individuals) even when the identities of people are concealed, since they it can be reconstructed by knowing that there is only one person living in a specific place that moves on a certain route at the specific time. Thus, privacy (and anonymity in particular) is not only related to identity hiding, but also to the size of population identified by a pattern, either directly or indirectly.

How can trajectories of mobile individuals be analyzed without infringing personal privacy rights? How can, out of privacy-sensitive trajectory data, patterns be extracted that are demonstrably anonymity-preserving?

Answering these questions will open up new important applications of data mining even when data to be analyzed is sensible.

3.3 Related Work

Privacy-preserving data mining, i.e., the analysis of data mining side-effects on privacy, has recently become a key research issue and is receiving a growing attention from the research community, as witnessed by the fact that major companies, like IBM and Microsoft, have been allocating their best resources to this problem in the past few years. However, despite such efforts, a common understanding of what is meant by "privacy" is still missing. This fact has led to the proliferation of many completely different approaches to privacy-preserving data mining, all sharing the same generic goal: producing a valid mining model without disclosing "private" data.

Similarly to the inference problem studied in statistical databases, privacy protection cannot be achieved by merely hiding individual identities, since often it is possible to reconstruct such factual information from mined rules. However, the inference problem becomes more complicated and therefore very hard to detect and resolve in the case of data mining.

In [VBF$^+$04] a survey on privacy-preserving data mining is described. Here we updated that survey by considering new work done in the field and gave a new taxonomy. In fact, we note that there are at least three orthogonal variables that can be used to classify the work on privacy-preserving data mining. They are:

Location of the data to be analyzed. Datasets can be centralized, i.e., owned by a single company/institution in a single database, or distributed over different sites of possibly different enterprises with different aims.

Information to hide. There can be different information to hide. Several works done in Privacy-preserving Data Mining aim at hiding a list of given patterns so that it is impossible to discover them: we called this approach *Intensional Knowledge Hiding*, since the knowledge we want not to disclose is intensional (rules, patterns, models that could be induced from the source database). In other works the focus is on the privacy of individuals, i.e., single rows of the source databases can not be disclosed, unless modifying them with a privacy-preserving algorithm.

Information to share/preserve. Also the information to share can be of two kinds: intensional or extensional. In the former case, we want to disclose patterns, or data mining models; in the second case, our aim is to share the whole database.

Although this taxonomy leads to eight classes, only three have been investigated enough [VBF+04]. We briefly review here the main aspects of those three classic approaches; then we describe a fourth emerging research theme, where our contribution, described in Chapter 5, is collocated, which focuses on the potential privacy breaches within the extracted patterns.

In the sequel, unless differently specified, we will omit the word *centralized* for the approaches in which the source information is stored in a single database.

3.3.1 Intensional Knowledge Hiding from Databases

This approach, also known as *sanitization*, is aimed at hiding some intensional knowledge (i.e., rules/patterns) considered sensitive, which could be inferred from the data which is going to be disclosed. This hiding is usually obtained by *sanitizing* the database in input in such a way that the sensitive knowledge can not be longer inferred, while the original database is changed as less as possible [AEI+99, CM00, DVEB01, OZ02, OZ03, SVC01]. Notice that in this context, what the data owner wants to share is the data.

This approach can be instantiated to association rules as follows: let $\mathcal{D}$ be the source database, R be a set of significant association rules that can be mined from $\mathcal{D}$, and let R_h be a set of rules in R. How can we transform database $\mathcal{D}$ into a database $\mathcal{D}'$, the released database, so that all rules in R can still be mined from $\mathcal{D}'$, except for the rules in R_h? This can be done by changing the database to decrease the support and confidence of some sensitive patterns (the one in R_h).

Note that there are two main drawbacks in this approach to privacy-preserving data mining:

1. in most of the applications we don't really want to share the whole database, but only some patterns;

2. it does not refer to any definition of privacy. In fact a formal definition of privacy is bypassed by giving as a input a set of sensitive rules R_h. Unfortunately no way is described to generate such a set of sensitive rules. Although for corporate privacy the hypothesis of having a set of sensitive intensional knowledge can hold, since only a small set of rules need to be kept secret and this set is actually decided by the management of the enterprises, in general it is not clear how to obtain them if we want to preserve the privacy of the individuals.

3.3.2 Extensional Knowledge Hiding from Databases

This approach, sometimes referred to as *distribution reconstruction*, addresses the issue of privacy-preservation by perturbing the data in order to avoid the identification of the original database rows, while at the same time allowing the reconstruction of the data distribution at an aggregate level, in order to perform the mining [AA01, AS00, ECB99, Evf02, EGS03, ESAG02, IB04, KDWS03, MS99, RH02]. In other words, the extensional knowledge in the dataset is hidden, but is still possible to extract valid intensional knowledge (note the opposite view and purposes with respect to the Intensional Knowledge Hiding approach previously described).

This approach can be instantiated to association rules as follows: let $\mathcal{D}$ be the source database, R be the set of significant (not sensitive!) association rules that can be mined from $\mathcal{D}$. Given a mining algorithm M, we want a sanitization algorithm P such that, if $P(\mathcal{D}) = \mathcal{D}'$ (where $\mathcal{D}'$ is a dataset that does not disclose any information on each singular row in $\mathcal{D}$) then we have that $M_P(\mathcal{D}') = R$ (i.e. the two datasets return the same set of patterns if mined).

This kind of perturbation has been already studied in context of statistical databases [AW91]. The most used form of perturbation is noise addiction. The main challenge is to find algorithms able to reconstruct the original distribution of the data from the perturbed one. In this way it is possible to share the perturbed (sanitized) version of the database making the receivers able to analyze the data, without disclosing the original database. There are few but nevertheless important drawbacks of noise addition techniques:

1. the quality of the computed statistics is not very high if a large-enough perturbation is used;

2. under some circumstances, the process can be reversed.. In fact recent works on spectral analysis shows that noise addition of random values is not really secure, and the original data can be computed from the perturbed one [KDWS05].

In the category of (Centralized) Extensional Knowledge Hiding we can also insert all the works done on k-anonymity [SS98, Swe02a, Swe02b]. The concept of k-anonymity is simple but effective: a database is k-anonymous if and only if every row appears at least k times. Actually, when checking for k-anonymity, only a subset of the database attributes are considered, the so-called quasi-identifiers. If not k-anonymous, a database can be modified (sanitized) to enforce k-anonymity. Any information extracted from a k-anonymous database regards at least k people, and this property make the data anonymous. Of course, larger values of k make the data more anonymous but at the same time, the quality of the data decreases. We believe that k-anonymity is a good way to formalize the privacy problems related to anonymity. Unfortunately, as showed in Chapter 5 (but also independently studied in [Agg05]), performing data mining over a large k-anonymized dataset is not useful since the k-anonymization process destroys most of the information on the association between attributes by generalizing their values. In this thesis (Chapter 5) we give an original way to shift the concept of anonymity from data to patterns, making k-anonymous data mining effective if we want to share intensional knowledge (such as patterns and data mining models).

3.3.3 Distributed Extensional Knowledge Hiding from Databases

This approach, also known as *Secure Multiparty Computation* or *Cryptographic Techniques* is aimed at computing a common data mining model from several distributed datasets, where each party owning a dataset does not communicate its extensional knowledge (its dataset) to the other parties involved in the computation [CHN$^+$96, CKV$^+$02b, DA01, DZ02, IGA02, KC02, Pin02, VC02]. The basic tools used to construct secure protocols for data mining are usually two:

1. Cryptography – especially using symmetric encryptions, where double encryption does not depend on the order, i.e. $E_{k1}(E_{k2}(x)) = E_{k2}(E_{k1}(x)))$.

2. Noise Addition – since some values are hidden by adding a random number.

This approach has been instantiated to association rules in two different way corresponding to two different data partitions: vertically and horizontally partitioned data. In the first case each site s holds a portion $\mathcal{I}_s$ of the whole vocabulary of items $\mathcal{I}$, and thus each itemset is split between different sites. Here, the key element for computing the support of an itemset is the "secure" scalar product of vectors representing the sub-itemsets in the parties. In the second case (horizontally partitioned data) the transactions of the database $\mathcal{D}$ are partitioned in n databases $\mathcal{D}_1, \ldots, \mathcal{D}_n$, each one owned by a different site involved in the computation. In such situation, the key elements for computing the support of itemsets are the "secure" union and "secure" sum privacy-preserving operations.

3.3.4 Intensional and Estensional Knowledge Hiding from Patterns

During the last year a novel problem has emerged in privacy-preserving data mining [FR04, KJC04, OZS04]. All the previous approaches were focused on producing a valid mining model without disclosing private data, but they still leave a privacy question open [KJC04]: do the data mining results themselves violate privacy?

As usual this issue has been investigated in completely different ways.

The work in [KJC04] compliments the line of research in distributed extensional knowledge hiding, but focusing on the possible privacy threat caused by the data mining results. In particular the authors study the case of a classifier trained over a mixture of different kind of data: *public* (known to every one including the adversary), *private/sensitive* (should remain unknown to the adversary), and *unknown* (neither sensitive nor known by the adversary). The authors propose a model for privacy implication of the learned classifier, and within this model, they study possible ways in which the classifier can be used by an adversary to compromise privacy.

The work in [OZS04] has some common aspects with the line of research in intensional knowledge hiding. But this time, instead of the problem of sanitizing the data, the problem of *association rule sanitization* is addressed. The data owner, rather than sharing the data prefer to mine it and share the discovered association rules. As usual for all works in intensional knowledge hiding, the data owner knows a set of restricted association rules that he does not want to disclose. The authors propose a framework to sanitize a set of association rules protecting the restricted ones by blocking some inference channels.

In [FR04] a framework for evaluating classification rules in terms of their perceived privacy and ethical sensitivity is described. The proposed framework empowers the data miner with alerts for sensitive rules which can be accepted or dismissed by the user as appropriate. Such alerts are based on an aggregate *Sensitivity Combination Function*, which assigns to each rule a value of sensitivity by aggregating the sensitivity value (an integer in the range $0 \dots 10$) of each attribute involved in the rule. The process of labeling each attribute with its sensitivity value must be accomplished by the domain expert, which knows what is sensitive and what is not.

3.3.5 Collocation of our Approach

The approach pursued in this thesis clearly collocates within the new emerging area described in Section 3.3.4, with distinctive original features [ABGP05a, ABGP05b, ABGP05c, ABGP06].

A common aspect of the three works in Section 3.3.4 is that they all require some *a priori* knowledge of what is sensitive and what is not. In our work we study when data mining results represent *per se* a threat to privacy, without any background knowledge of what is sensitive. The fundamental difference of these approaches with ours lies in generality: we propose a novel, objective definition of privacy compliance of patterns without any reference to a preconceived knowledge of sensitive data or patterns, on the basis of the rather intuitive and realistic constraint that the anonymity of individuals should be guaranteed.

It should also be noted the different setting if compared with the other works in privacy-preserving data mining: in our context no data perturbation or sanitization is performed on the original data, as we allow real mining on the real data, while focusing on the anonymity preservation properties of the extracted patterns.

Chapter 4

Abduction for Classification

Abstract

In this Chapter we present our results on the use of abductive reasoning for data mining purposes. We show how classification can be interpreted as an abductive logic programming task, allowing the use of domain specific constraints given by the user. Decision tree classification models are interpreted in an abductive way exploiting constraints and obtaining improvements where some data is missing. We also extend the framework in order to take into account probabilistic information on both abducibles and background theory. A cost-based abduction framework feasible for data mining applications is described and some experiments are described to show the improvements of the quality of the results.

4.1 Contribution and Organization

This Chapter contains results obtained using abductive reasoning in the field of data mining [AMT03]. Some classifiers, even if based on decision tree induction like C4.5 [Qui93], produce as output a set of rules in order to classify new given examples. Most of these rule-based classifiers make the assumption that at classification time we can know all about new given examples. Probabilistic approaches make rule-based classifiers able to get the most probable class, on the basis of the frequency of the missing attribute in the training set [Qui89]. This kind of assumption sometimes leads to wrong classifications. The abductive approach presented here is useful to (help to) choose which classification rule to apply when a new example with missing information needs to be classified, exploiting knowledge about the domain.

In Section 4.2 we show how to explain a given classification computing missing attribute values through abduction. We concentrated on a very focused goal: proving that a careful representation of the results of a data mining algorithm (a decision or classification tree), a careful representation of extra domain knowledge (constraints), and a careful choice of a reasoning technique (abduction) can substantially improve the behavior of the extracted model (classification). The method presented can handle some extra domain knowledge in form of integrity constraints, as shown in Section 4.3. Section 4.4 describes some experiments using machine learned association rules in order to use the previous approach when we deal with no external knowledge. Section 4.5 presents an extension of the abductive framework in order to take into account probabilistic knowledge as cost of hypothesis. In the knowledge base, the classification rules obtained by the paths of the decision tree are merged with the association rules mined from the dataset, and classification results in the experiments are coupled with an expressive explanation of the choices made at classification time.

4.2 Classification as an Abductive Problem

In this Section we show how the classification task using decision trees can be directly seen as an abductive problem. To do this, we first need to spend some words on the notation and then formally define what decision tree classification means and on the transformation of decision trees into abductive logic programs.

Given a tree T and a path π in T, we denote by $Attr(\pi)$ the example $\{a_1 = v_1, \ldots, a_k = v_k\}$ where each a_i is a non-leaf node in T and each v_i is the attribute value labeling the branch exiting from a_i. Moreover, we denote by $Class(\pi)$ the class labeling the leaf node of π.

The following defines the concept of classifying an example by means of a decision tree in case of missing information. The usual definition of classifying an example is given as a special case, in which there are no missing information.

Definition 1. *Let T be a decision tree,* e *be an example and* c *be a class. We say that the example* e **may be** *classified as* c *by T via δ, denoted by $T \overset{c,\delta}{\Longrightarrow}$* e *if there exists a path π in T with leaf node $Class(\pi) = c$ such that $e \cup Attr(\pi)$ is an example and $\delta = Attr(\pi) \setminus e$.*
If $T \overset{c,\{\}}{\Longrightarrow}$ e *we say that the example* e **is** *classified as* c *by T, and we simply write* $T \overset{c}{\Longrightarrow}$ e

Notice that, in $T \overset{c,\delta}{\Longrightarrow} e$, the condition that $e \cup Attr(\pi)$ must be an example ensures that the attribute values of e are compatible with the set of tests represented by π. In other words, if for some attribute a, $a = v \in e$ and $a = v' \in Attr(\pi)$, then $v = v'$. Moreover, notice that δ represents the extension to the example e which is needed in order to classify it as c from the chosen path.

Example 1. Let T be the decision tree of Example 1. We have:

- $T \overset{Yes}{\Longrightarrow} \{Overlook = Sunny, Wind = Strong, Humidity = Low\}$

- $T \overset{No,\delta}{\Longrightarrow} \{Overlook = Sunny\}$, where $\delta = \{Humidity = High\}$

Now, let us explain how to transform a decision tree into an abductive logic program. Let T be a decision tree and π be a path in T. The rule r_π associated with π is the Horn clause $c \leftarrow a_1(v_1), \dots, a_n(v_n)$ such that:

- $Class(\pi) = c$, and
- $Attr(\pi) = \{a_1 = v_1, \dots, a_n = v_n\}$

Notice that attribute names are viewed as unary predicate symbols, and that an attribute/value pair $a = v$ is mapped into an atom $a(v)$. Moreover, by *(v)* in Def. 1, given a path π, $Attr(\pi)$ is an example. The Horn clause program P_T associated with a decision tree T is the set of rules

$$P_T = \{r_\pi \mid \pi \text{ is a path in } T\}.$$

Finally, we associate with T the set IC_T of *canonical* integrity constraints containing a denial

$$\leftarrow a(x), a(y), x \neq y$$

for each attribute $a \in \mathcal{A}$.

As we will see next, attribute names are viewed as abducible predicates, and sets of abducibles will represent possible classifications. The integrity constraints we have just defined ensure that in each consistent set of abducibles, an attribute occurs at most once (cfr. case (v) of Def. 1 in Chapter 1).

Given an example $e = \{a_1 = v_1, \dots, a_k = v_k\}$, let Δ_e be the set of atoms

$$\Delta_e = \{a_1(v_1), \dots, a_k(v_k)\}.$$

We have now set up all the ingredients that are needed to associate an abductive framework to a decision tree T.

Definition 2. *Given a decision tree T, the abductive framework AB_T associated with T is the triple*

$$AB_T = \langle P_T, \mathcal{A}, IC_T \rangle$$

where P_T is the Horn program associated with T, IC_T is the set of canonical integrity constraints associated with T and $\mathcal{A}$ is the set of attribute names.

The following theorem formalizes the correspondence between classifications in T and abductive explanations in AB_T.

Theorem 1. *Let T be a decision tree and AB_T be the corresponding abductive framework. Let also* e *be an example. Then Δ is a Δ_e−minimal explanation for* c *with respect to AB_T if and only if for some path π in T we have $T \overset{c,\delta}{\Longrightarrow} e$ and $\Delta = \Delta_\delta$.*

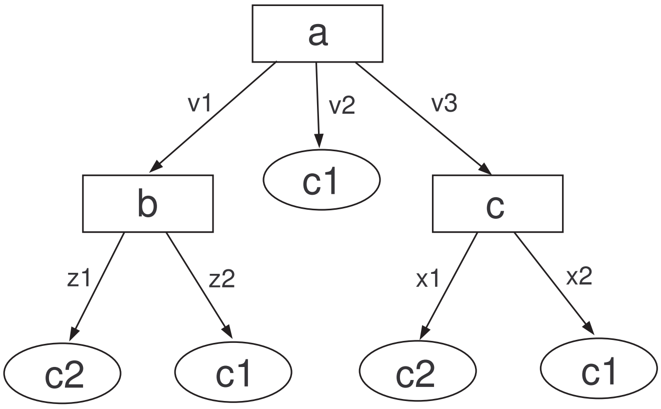


Figure 4.1: A simple example of decision tree

Proof.
($\Longleftarrow$) Assume that $T \overset{c,\delta}{\Longrightarrow} e$. Then, by Def. 1, for some path π in T, $e \cup Attr(\pi)$ is an example and $\delta = Attr(\pi) \setminus e$. Consider the rule r_π

$$c \leftarrow a_1(v_1), \ldots, a_n(v_n).$$

and let $body(r_\pi) = \{a_1(v_1), \ldots, a_n(v_n)\}$. It is clear that $body(r_\pi) = \Delta_{Attr(\pi)}$ and $P_T \cup body(r_\pi) \models c$. Let $\Delta = body(r_\pi) \setminus \Delta_e$: clearly, $P_T \cup \Delta_e \cup \Delta \models c$ and $\Delta = \Delta_\delta$. Moreover, since $e \cup Attr(\pi)$ is an example and $(\Delta_e \cup \Delta) \subseteq \Delta_{e \cup Attr(\pi)}$, $P_T \cup \Delta_e \cup \Delta \cup IC_T$ is consistent. Hence, Δ is an explanation for c given Δ_e. By construction, Δ is also a Δ_e-minimal explanation.
($\Longrightarrow$) Assume Δ is a Δ_e-minimal explanation for c. Clearly, $\Delta \cap \Delta_e = \{\}$ by minimality of Δ. Since $P_T \cup \Delta_e \cup \Delta \models c$, by construction of P_T there exists a path π in T such that r_π is a rule of the form

$$c \leftarrow a_1(v_1), \ldots, a_n(v_n)$$

and $\{a_1(v_1), \ldots, a_n(v_n)\} \subseteq \Delta_e \cup \Delta$.

Let $\delta = Attr(\pi) \setminus e$. By construction and minimality of Δ, we have $\Delta = \Delta_\delta = \{a_1(v_1), \ldots, a_n(v_n)\} \setminus \Delta_e$. Finally, it is clear that $e \cup Attr(\pi)$ is an example, by the consistency of $P_T \cup \Delta_e \cup \Delta \cup IC_T$ and the observation that $\Delta_{e \cup Attr(\pi)} = \Delta_e \cup \Delta$. □

Example 2. Let us consider the decision tree in Fig. 4.1. In the abductive framework $AB_T = \langle P_T, \mathcal{A}, IC_T \rangle$ associated with T we have:

- P_T is the following set of rules:

$$\begin{array}{ll} c1 \leftarrow a(v2). & c2 \leftarrow b(z1), a(v1). \\ c1 \leftarrow b(z2), a(v1). & c2 \leftarrow c(x1), a(v3). \\ c1 \leftarrow c(x2), a(v3). & \end{array}$$

- IC_T is the following set of canonical integrity constraints:

$$\leftarrow a(x), a(y), x \neq y. \qquad \leftarrow b(x), b(y), x \neq y. \qquad \leftarrow c(x), c(y), x \neq y.$$

- $\mathcal{A} = \{a, b, c\}$.

Let us consider the example $e = \{a = v1, b = z1\}$ and the corresponding initial set of abducibles $\Delta_e = \{a(v1), b(z1)\}$. Clearly $\Delta = \{\}$ is a Δ_e-minimal explanation for $c2$, and indeed it corresponds to the leftmost path in the tree. Notice that also $\Delta' = \{c(x1)\}$ is an explanation for $c2$ given Δ_e, but it is not Δ_e-minimal and indeed it does not correspond to any path in the tree matching the attribute/values pairs given in e.

4.3 Exploiting Domain Specific Knowledge

The abductive view of classification given in the previous Section can be seen as an alternative, though equivalent, way of performing classification given some decision tree. It is worth noting that the abductive reasoning required in this alternative view is very limited and does not fully exploit the potential power of abductive reasoning in logic programming as sketched in Sect. 2.1. In particular, the transformation requires the use of a pre-defined, canonical set of integrity constraints that simply avoid explanation to contain different values for the same attribute. We show in this Section that one way to exploit abductive reasoning is to add *domain specific* knowledge in order to improve the classification task. Domain specific knowledge may be taken into account during classification in many ways, e.g. to rule out some classifications or to prefer one particular classification over another. Up to our knowledge, taking into account domain specific knowledge in standard decision-tree based classification algorithms may be not straightforward and may require substantial modifications of these algorithms. On the other hand, abductive frameworks, and existing concrete implementations of them, are already equipped with mechanisms that can be directly exploited to represent and handle domain specific knowledge.

We exploit integrity constraints (beyond the canonical ones) in abductive frameworks as a way to express domain specific knowledge. Let us show some examples.

Example 3. Consider the decision tree T of Example 4.1. As we have already pointed out, the attributes which label the internal nodes of decision trees may not be all the attributes that examples are equipped with. In the current example, assume that the original set of attributes contained also an attribute d, beyond the attributes $\{a, b, c\}$ occurring in the tree. Assume now that an example is given $e = \{d = w1\}$: it is clear that, since d does not occur in the decision tree, any classification of e can be done using T. However, assume that we have extra knowledge on the domain at hand, expressing the fact that the value $w1$ for the attribute d is incompatible both with the value $v1$ of the attribute a and with the value $x1$ of the attribute c. It is easy to see that this extra knowledge can be used to classify the example e as belonging to class $c1$. In the abductive framework corresponding to T the extra knowledge can be easily coded by adding the following integrity constraints:

$$\neg(d(w1), a(v1)) \qquad \neg(d(w1), c(x1))$$

If we consider now the full abductive framework $\langle P, \mathcal{A}', Ic\rangle$, where P is the program obtained as in Example 4.1, $\mathcal{A}' = \{a, b, c, d\}$ and Ic are the integrity constraints we have just shown, we observe that $c2$ has no abductive explanation given $\Delta_e = \{d(w1)\}$, whereas the goal $c1$ has two Δ_e-minimal explanations given Δ_e, namely $\Delta_1 = \{a(v2)\}$ and $\Delta_2 = \{a(v3), c(x2)\}$. Notice the correspondence between these two solutions and the two right-most paths with $c1$ as the leaf node. From a computational point of view the two explanations can be computed using, e.g., the KM-proof procedure. The consistency checking phase of the proof procedure rules out the two potential Δ_e-minimal explanations $\{b(z1), a(v1)\}$ and $\{a(v3), c(x1)\}$ for the goal $c2$. Indeed, the first one is inconsistent with the integrity constraint $\neg(d(w1), a(v1))$, and the second one is inconsistent with the integrity constraint $\neg(d(w1), c(x1))$.

As the previous example points out, integrity constraints can be used to add knowledge relating the values of the various attributes, including those which do not occur in the original decision tree. This can help and improve the classification task. The next example shows that integrity constraints may add knowledge which is relevant to the attributes already occurring in the decision tree, although it is not explicit in the decision tree itself.

Example 4. Let us consider again the decision tree of Example 1. Assume that, as it is often the case, whenever there is strong wind the humidity is not high:

$$\neg(Humidity(High), Wind(Strong)).$$

It is important to point out that this kind of knowledge may not be implicit in the training set from which the original decision tree was built. Actually, the examples in the training set may even contradict this knowledge (see, e.g., Table 1.1). Indeed, this knowledge may arise from knowledge sources different from the ones which provide the training set (in this particular example this knowledge may be associated with typical weather forecast knowledge bases). Assume now that we want to classify an example described simply as $e = \{Overlook = Sunny, Wind = Strong\}$. In the corresponding abductive framework, given the initial set $\Delta_e = \{Overlook(Sunny), Wind(Strong)\}$, the classification Yes has an abductive explanation $\Delta_1 = \{Humidity(Low)\}$ and the classification No has an abductive explanation $\Delta_2 = \{Humidity(High)\}$. If we consider now the above integrity constraint,

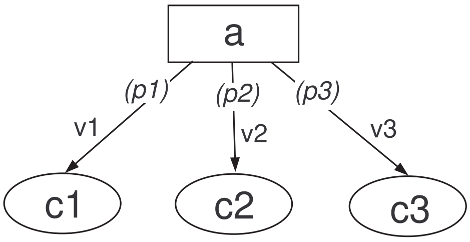


Figure 4.2: A simple example of a decision tree with probabilities associated to each branch

the abductive explanation Δ_2 is ruled out, due to the fact that the full explanation given by $\Delta_e \cup \Delta_2 = \{Overlook(Sunny), Wind(Strong), Humidity(High)\}$ is inconsistent. Thus, by adopting the very same computational mechanism we obtain a correct classification with respect to the extra domain specific knowledge.

4.3.1 Dealing with Probabilities

In many cases, decision trees can be equipped with probability measures on the outcome of attribute/values test. In other words, each branch of the tree is labeled both by a value corresponding to the attribute labeling the parent node, as well as with a probability measure which indicates how likely is that an observation exhibits this value for the given attribute. This kind of probability information is clearly useful when we try to classify new examples with missing attribute information. Even in this case, extra domain knowledge can be taken into account in the classification task in order to dynamically get better probability measures on possible classification of new under-specified examples. To show this let us consider a very simple example.

Example 5. Let us consider the simple decision tree in Fig. 4.2. There is only one internal node, the root, which represents a test on a 3-valued attribute named a. Probabilities are represented between parenthesis. The classification rules extracted from the above tree are the following:

$$c1 \leftarrow a(v1) \qquad c2 \leftarrow a(v2) \qquad c3 \leftarrow a(v3).$$

Notice that we could have more information about a new observation than the a attribute only, but the tree-based classification (and its associated class ruleset) will use only information about a. But what happens if we know nothing about a? If we try to classify a new example with no information about a we get as output the class with highest probability. Again, using integrity constraints in the corresponding abductive framework which makes explicit extra information about the domain, we can cut some branches during the classification task, thus obtaining better classifications and also dynamically improving probability values. For instance, let this domain knowledge be represented by the integrity constraint $\neg(a(v1), b(z2))$. In this case, if we are asked to classify the example $\{b = z2\}$ we will discard the classification $c1$ whose abductive explanation $\Delta_1 = \{a(v1)\}$ is inconsistent with the integrity constraint. On the other hand, we would compute the classifications $c2$ and $c3$ with the corresponding explanations $\Delta_2 = \{a(v2)\}$ and $\Delta_3 = \{a(v3)\}$. We can also dynamically re-compute the associated probabilities which will be $\frac{p2}{p2+p3}$ as far as $c2$ is concerned, and $\frac{p3}{p2+p3}$ as far as $c3$ is concerned.

4.4 Experiments on Association Rules as Integrity Constraints

The techniques we described so far need a specific external knowledge about the domain, that we use as integrity constraints in the abductive framework. In that work we talked about domain knowledge as rules that can came from the human experts.

Another way to get rules about the domain is to infer them from input data. In fact, as showed in [MT03], it is possible and also easy to handle association rules (they called them *Active Rules* in their work) in a general purpose abductive framework. This idea makes the construction of a completely automated abductive framework possible by joining together two of the most important data mining paradigms: decision tree based classification and association rules.

We began to analyze and study more deeply this kind of abductive techniques for data mining making use of association rules as integrity constraints, using a dataset containing real world data coming from web logs.

4.4.1 Dataset Description

The used dataset contains a table where every column corresponds to a binary attribute indicating whether a specific (depending on the attribute) set of web pages was accessed or not; every row represents a single registered user.

The list of attributes contained groups of pages such as CITYSEARCH, SPECIAL, SHOPPING, FINANCE, PHOTOGALLERY, SURVEYS, ...

The original dataset file, containing 55 Mb of, was reduced (in order to make experiments faster) to about 1 Mb, corresponding to 15265 examples.

4.4.2 Induced Decision Tree

The dataset was used to predict user accesses to new groups of pages, trying to induce several decision trees (using a popular tool called Weka). A tree we consider interesting for its size and number of leaves is the one able to predict the attribute called ARTICLES. This tree, generated using the algorithm C4.5, contains 44 leaves, 87 nodes: by using our Java application, it was automatically converted into a Prolog file containing rules, the definitions of abducibles and the canonical integrity constraints.

4.4.3 External Knowledge

Here the problem of finding integrity constraints has been solved by analyzing in an interactive way some association rules obtained with the Apriori algorithm. Because of the unbalance between negative and positive values of some attributes, the produced rules were of high *Confidence* and *Support* values but of no utility. *Lift* and *Leverage* metrics give us better results: the obtained association rules actually represent dependencies among attributes. In order to test the impact of using such rules (with high value of Confidence, Lift and Leverage) the following rule was chosen:

$$\text{CITYSEARCH=Y} \Rightarrow \text{SPECIAL=N}$$

The analysis of the entropy for those attribute shows that $H(\text{SPECIAL}) \approx 0.86 > H(\text{SPECIAL}|\text{CITYSEARCH=Y}) \approx 0.24$, so the information CITYSEARCH=Y significantly reduces the uncertainty on the value of SPECIAL.

4.4.4 Results

Tests on classification using an abductive framework starting from the decision tree as theory and association rules as integrity constraints give evidence of the capability to infer the right class value even when there are no missing values. Here the summary of the results:

- 9 internal nodes and 10 leaves of the original tree are considered not so useful, reducing of $\sim 21.8\%$ the size of the tree;
- on 15265 examples in the dataset, 6441 ($\sim 42.2\%$) can be classified even if there are missing values;
- in the subset of 6441 examples involved, 188 of them are wrongly classified, vs. 166 of the original tree; the error, on these instances, changes from $\sim 2.6\%$ to $\sim 2.9\%$ and globally the accuracy of the prediction decreases of $\sim 0.12\%$.

The results can be considered very promising, especially considering that we used only a single rule (the framework can handle several rules at the same time). The accuracy error slightly increased (the tree was obtained with the C4.5 algorithm, so it was already pruned!).

These first results make us claim that using a non-pruned tree (for instance, decision trees obtained using the popular ID3 algorithm) the abductive framework can obtain significantly better results with respect to the original induced tree.

4.4.5 Abductive Framework vs. Pruning

Constrained Abductive inference can be seen as an interpretative way to prune a decision tree. Anyway, in several cases abductive inference can be more powerful than pruning.

Let see the decision tree in Fig. 4.3.

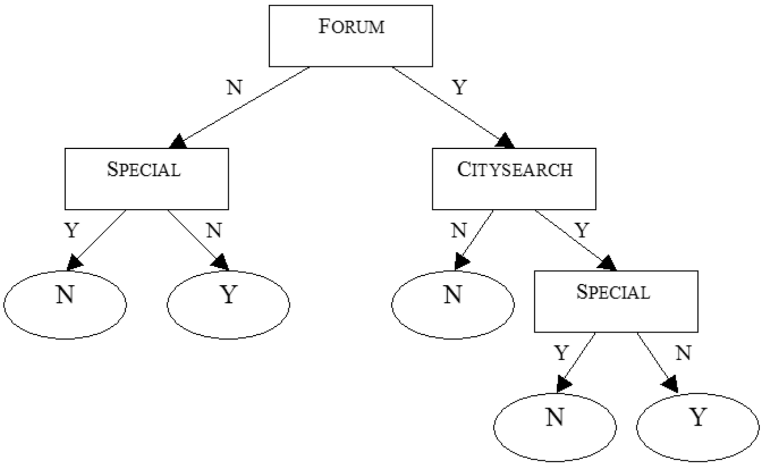


Figure 4.3: A subtree induced from the dataset.

Using this decision (sub)tree let consider, as external knowledge, the integrity constraint shown so far (CITYSEARCH=Y ⇒ SPECIAL=N). Fig. 4.4 shows the subtree resulting from a pruning step that takes into account this external knowledge. The node SPECIAL in the rightmost path of the tree has been pruned, and its leaf Y took its place. Using this external knowledge, no other pruning is possible.

Notice that the pruned tree in Fig. 4.4 is not equivalent to the inference performed by the abductive framework on the original tree. In fact, if we take into account an example with CITYSEARCH=Y then we can classify it as Y even if we know nothing about the value of the FORUM attribute, using constrained abduction. On the other hand, it is not possible to get the same result using the pruned tree, as in this case we need the value of FORUM in order to correctly classify the example.

4.5 Cost-Based Abduction in Data Mining

The main aim of this Section is to show a way to join the results of a data mining classification algorithm (decision-tree based) with some extra domain knowledge coming from an associative data mining model (interesting association rules), obtaining the input to an abductive environment able to classify but also to specify

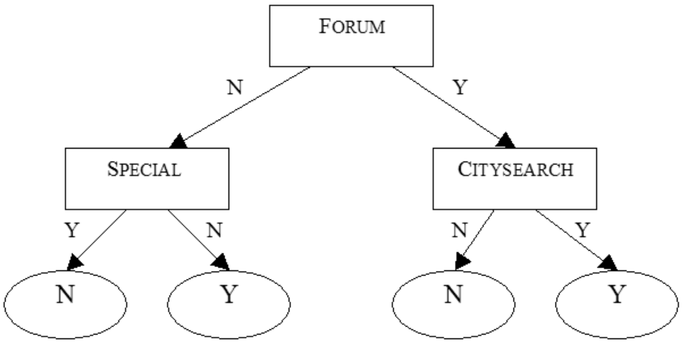


Figure 4.4: The subtree in Fig. 4.3 pruned using integrity constraint CITYSEARCH=Y ⇒ SPECIAL=N.

the reason of such a classification (by predicting the missing values). In particular, three new features are present in the new abductive scheme we used:

1. Goals can be assumable;

2. Assumption at various levels of specificity

3. Redundancy of assumptions must be taken into account (yielding more economic proofs);

Among the ones analyzed in Chapter 2, *cost-based abduction* showed to be best framework that fits those requirements. With respect to the logic-based abductive framework we make use of so far, we have costs (sometimes referred to as weights) associated to each abducible and to rules in the background knowledge, that represent the associated measures of uncertainty.

4.5.1 Overview of the Framework

The final diagram of the classification system is the one in Fig. 4.5. As it is shown, the inputs of the system are an instance to classify and a dataset. A decision tree is induced from the dataset (using a C4.5-like algorithm), while an instance of the Apriori algorithm extracts frequent items and association rules. The absolute support of frequent items is a parameter used to compute the costs for abducing them.

The framework is based on cost-based abduction (see Section 2.2.4), where each abducible has an associated cost that must be paid to be assumed. In our framework,

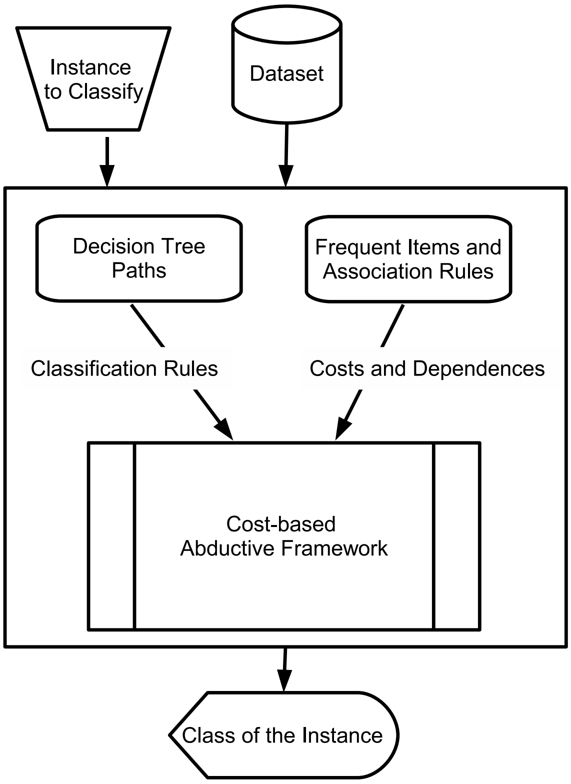


Figure 4.5: A classification system based on abduction that exploits association rules as domain knowledge

rules also have an associated cost to be paid if the rule is used to classify (the final class is the one implied by the minimal cost abductive proof).

We chose to extend the framework including costs for the following reasons:

1. we need a quantitative measure to choose among different possible solutions (abductive proofs). Note that there can be more than one explanation for the same classification;

2. we would like to exploit the probabilistic information in our knowledge base to discard some unlikely proofs.

The procedure we used to analyze the search space is a complete depth-first search, with some pruning done exploiting basic integrity constraint described later. Other constraints that should hold in any feasible solution are verified performing a post processing step.

Cost Characteristics in the Abductive Framework

In the following we show the characteristics of the abductive framework we used:

- every atom occurring in at least one rule is associated with an integer representing the its cost. For instance, in the following rule

  $$Q^{c(Q)} \leftarrow P_1^{c(P_1)} \wedge P_2^{c(P_2)}$$

  Q (*head*), P_1 and P_2 (*body*) are atoms occurring in the rule, and their costs are respectively $c(Q)$, $c(P_1)$ and $c(P_2)$;

- assuming the head of a rule has a cost lower or equal than assuming the atoms in the body

  $$c(Q) \leq c(P_1) + c(P_2)$$

- redundancies are exploited in order to find the least cost proof. E.g., in the knowledge base

  $$\begin{array}{c} Q_1^{12} \leftarrow P_1^6 \wedge P_2^8 \\ Q_2^{12} \leftarrow P_2^8 \wedge P_3^5 \end{array}$$

  the least cost solution for $\{Q_1, Q_2\}$ is $\{P_1, P_2, P_3\}$, with a cost of $6+8+5 = 19$ (since we pay $c(P_2)$ only once), while the feasible but non-optimal solution $\{Q_1, Q_2\}$ costs 22;

4.5.2 Cost Assignment

The costs we associated to each atom, rule or solution is related to the the probability of them, as showed in the following.

Cost of an atom

Each atom has a weight equals to its infrequency (the number of tuples in which the atom is false), that is $sup(|\mathcal{D}|) - sup(A) = sup(\neg A)$, where $|\mathcal{D}|$ is the number of tuples in the dataset $\mathcal{D}$. In terms of probability, the cost of the atom A is equal to $|\mathcal{D}|P(A)$.

Cost of a rule

Again, each rule has a cost equals to its infrequency (the number of tuples in which the rule is false). For instance, to use the rule $H \leftarrow B$ we must pay

$$c(H \leftarrow B) = sup(B \wedge \neg H)$$

In terms of rule confidence, we have:

$$c(H \leftarrow B) = sup(B \wedge \neg H) = |\mathcal{D}|P(B \wedge \neg H) =$$

$$= |\mathcal{D}|P(\neg H|B)P(B) = |\mathcal{D}|(1 - P(H|B))P(B) = (1 - conf(H \leftarrow B))sup(B)$$

Cost of an abductive solution

An abductive solution A for a goal G is a set of atoms $\{a_1 \wedge \cdots \wedge a_n\}$ s.t.: $A \rightarrow G$, where $A = \{a_1 \wedge \cdots \wedge a_n\}$.

A desiderata is that the cost paid for A would be (inversely) related to the frequency of the solution A.

Proposition 1. *By assuming mutual exclusion among negated atoms, we have*

$$|\mathcal{D}|P(\neg A) = c(A)$$

.

Proof. $|\mathcal{D}|P(\neg A) = |\mathcal{D}|P(\neg a_1 \vee \cdots \vee \neg a_n)$. By using the naïve hypothesis of mutual exclusion, we have $|\mathcal{D}|P(\neg a_1 \vee \cdots \vee \neg a_n) = |\mathcal{D}|(P(\neg a_1) + \cdots + P(\neg a_n)) = c(a_1) + \cdots + c(a_n) = c(A)$. □

This is a (far from tight) approximation of the probability $P(\neg A)$, since it is not even bounded to be in $[0, 1]$. Anyway, this is not a problem since we are not really interested in the actual value of infrequency, but only in solutions with low costs that are, in some sense, more likely. Proposition 1 shows the relation between costs and probability of an abductive solution.

4.5.3 A General Abductive Framework

The final framework is as general as possible:

1. all the predicates are abducibles, even whose in the head of the rules.
2. the knowledge base can have loops, i.e.: $A \rightarrow B \quad B \rightarrow C \quad C \rightarrow A$

The proof procedure we implemented can manage both requirements. It is an ad-hoc procedure developed on the top of SICStus Prolog v3 [SIC] that handles loops during the search step and prune some candidate proofs according to the weights of atoms to be proved. The rest, such as sorting the feasible solutions and remove some other integrity constraints on atoms, is done in a post processing step.

A three-step representation of the system is showed in Fig. 4.6; details on the offline computation, i.e., the construction of the knowledge base, are reported in Fig. 4.7.

4.5.4 Experiments

We conducted a set of experiments on various datasets from the UCI KDD Repository [HB99]. Here two examples of classification through our abductive framework are reported: they show that the framework helps to understand the reason why instances are associated to a specific class.

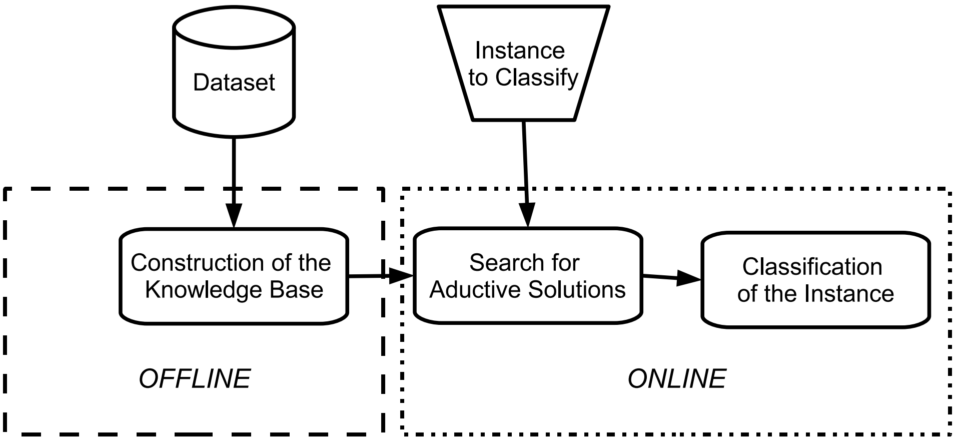

Dataset
Instance to Classify
Construction of the Knowledge Base
Search for Aductive Solutions
Classification of the Instance
OFFLINE
ONLINE

Figure 4.6: A representation of the abductive system for data mining.

Zoo

Zoo is a dataset that contains characteristics of 7 classes of animals. Each tuple, corresponding to a different animal, has 17 attributes (besides name): 15 are boolean and 2 are numeric-valued. Fig. 4.8 show the decision tree induced from the dataset. We try to classify the following instance:

feathers	milk	backbone	predator	legs
0 (false)	0 (false)	0 (false)	0 (false)	6

We want to classify an instance of an animal which has no feathers, does not suckle, has not backbone, is not a predator and finally we know it has 6 legs. According to the original decision tree the instance should be classified "6", i.e., *insect*. Since the decision tree has a reasonable size, it is possible to follow the paths followed by the classification algorithm. In this case there are two possible paths, shown in Fig. 4.8 through arrows: both of them leads to the class of insects, but it is not clear if the animal is *airborne* or not. The abductive system uses a knowledge base induced from the same dataset, with association rules computed using $sup = 70\%$ and $conf = 80\%$. The system classifies the given instance as an insect as well, but it also gives the following explanation (abductive proof):

```
feathers_abd(0,0), milk_abd(0,0), backbone_abd(0,0),
legs_abd(6,0), predator_abd(0,0), airborne_abd(0,24).
```

with cost 24. In other words, it assumed that the animal was not airborne (the first parameter of the predicate `airborne_abd` is "`0`") in order to complete the proof, by paying 24 (the second parameter is the cost). The other attributes of the animals have been provided by the user, so it is possible to prove them at no additional

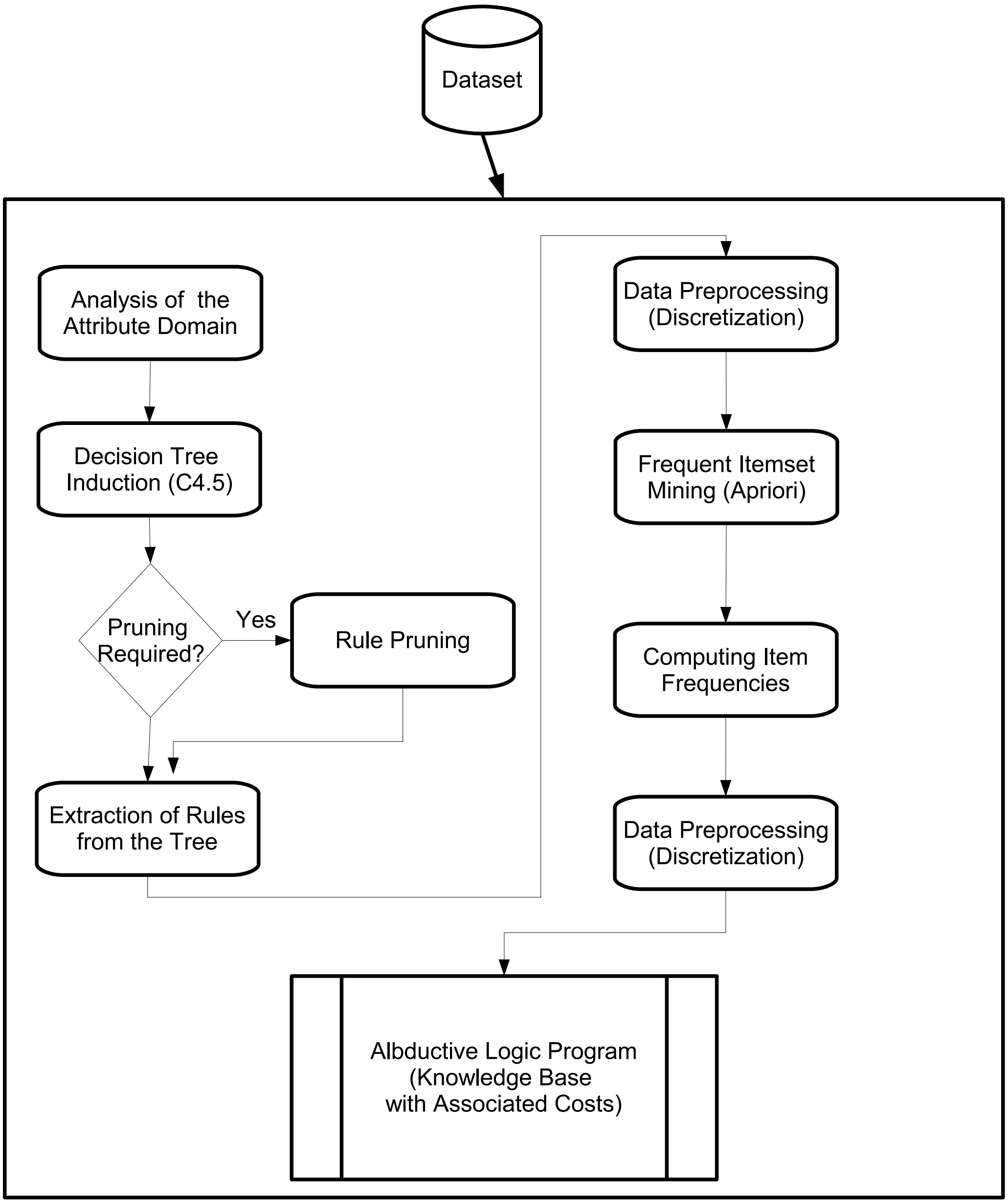

Dataset
Analysis of the Attribute Domain
Decision Tree Induction (C4.5)
Pruning Required?
Yes
Rule Pruning
Extraction of Rules from the Tree
Data Preprocessing (Discretization)
Frequent Itemset Mining (Apriori)
Computing Item Frequencies
Data Preprocessing (Discretization)
Albductive Logic Program (Knowledge Base with Associated Costs)

Figure 4.7: Details on the offline procedure responsible for the construction of the knowledge base.

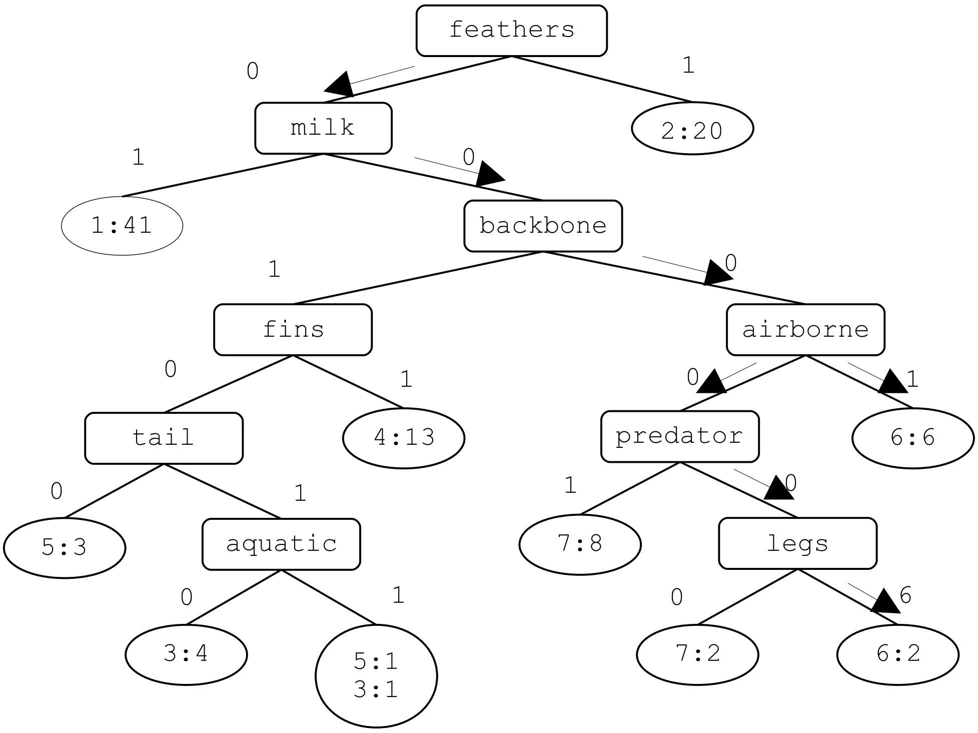


Figure 4.8: Induced decision tree using the Zoo dataset

costs. Remember that the cost of an attribute is equal to its "infrequency", in fact non airborne animals are 77 out of 101 animals in the zoo dataset.

Another example of classification is given by the following instance:

domestic	milk	tail	breathes	aquatic	toothed
0 (false)	0 (false)	0 (false)	1 (true)	1 (true)	1 (true)

The original decision tree based algorithm wrongly classifies it as a *bird* (class "2"), as shown in Fig. 4.9.

Instead, the abductive system classifies the instance as an *amphibian*, making the right choice, with the following explanation:

```
tail_abd(0,0), etc42(4), milk_abd(0,0), breathes_abd(1,0),
backbone_abd(1,18), etc1(0), toothed_abd(1,0), domestic_abd(0,0).
```

That is, the system assumed that the animal has a backbone, and proved that it doesn't have feathers. The atoms `etc1` and `etc42` are associated to rules 1 and 42 used in the abductive proof, and here reported:

```
feathers(0,20) :- toothed(1,40), etc1(0).

fins(0,17) :- breathes(1,21), domestic(0,13), etc42(4).
```

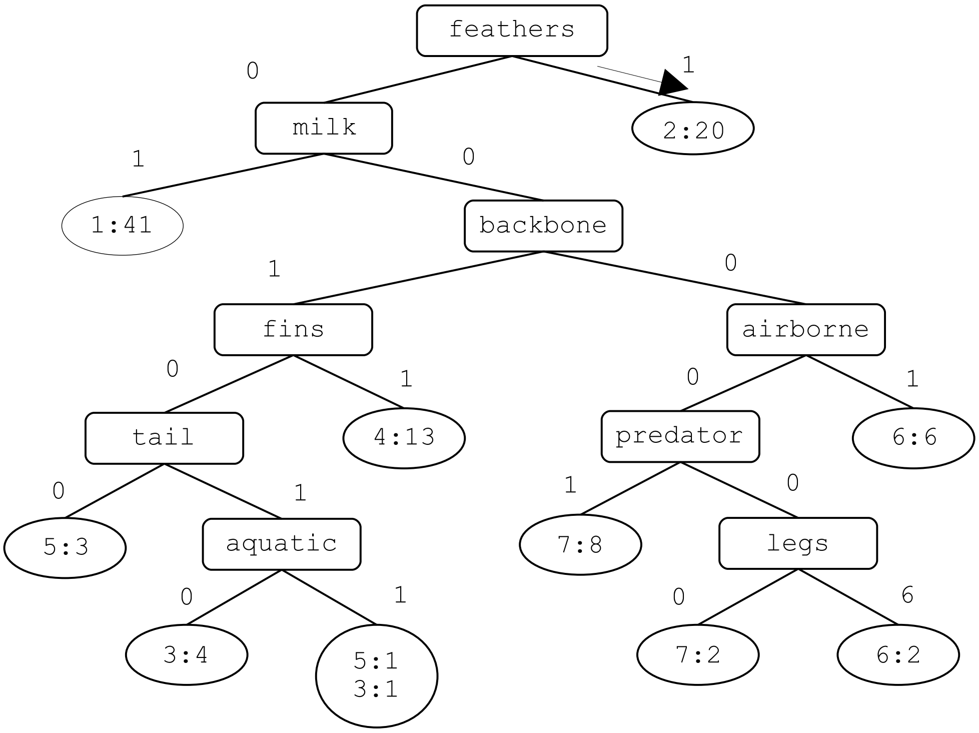


Figure 4.9: Induced decision tree using the Zoo dataset: the instance is erroneously classified as a *bird* (class "2")

Note that all the rules are automatically extracted from the dataset by using the Apriori algorithm; the only inputs are the dataset and the instance to classify. As the common sense suggests, toothed animals have no feathers. This information, first induced and then exploited by the abductive system, is completely ignored by the decision tree based classifier.

The abductive proof ends with the following classification rule

```
type(5,97) :- feathers(0,20), milk(0,41), backbone(1,18),
              fins(0,17), tail(0,75).
```

Since fins and feather have been proved, the only costs associated with this solution are related to the use of the rule 42 (rule 1 has a cost equal to zero due to its high confidence) and to the assumption of having a backbone.

This example shows that the abductive system is able to infer some information about the instance to classify, by exploiting the knowledge domain (induced from the dataset). The extra information derived can lead to classification improvements.

Adult

We also tested our system against the *Adult* dataset. The training set contains 32561 records, while the test set contains 16281 records. The task is to predict

age	continuous
workclass	private, self_emp_not_inc, self_emp_inc, federal_gov, local_gov, state_gov, without_pay, never_worked
fnlwgt	continuous
education	bachelors, some_college, 11th, hs_grad, prof_school, assoc_acdm, assoc_voc, 9th, 7th_8th, 12th, masters, 1st_4th, 10th, doctorate, 5th_6th, preschool
education_num	continuous
marital_status	married_civ_spouse, divorced, never_married, separated, widowed, married_spouse_absent, married_af_spouse
occupation	tech_support, craft_repair, other_service, sales, exec_managerial, prof_specialty, handlers_cleaners, machine_op_inspct, adm_clerical, farming_fishing, transport_moving, priv_house_serv, protective_serv, armed_forces
relationship	wife, own_child, husband, not_in_family, other_relative, unmarried
race	white, asian_pac_islander, amer_indian_eskimo, other, black
sex	female, male
capital_gain	continuous
capital_loss	continuous
hours_per_week	continuous
native_country	united_states, cambodia, england, puerto_rico, canada, germany, outlying_us(guam_usvi_etc), india, japan, greece, south, china, cuba, iran, honduras, philippines, italy, poland, jamaica, vietnam, mexico, portugal, ireland, france, dominican_republic, laos, ecuador, taiwan, haiti, columbia, hungary, guatemala, nicaragua, scotland, thailand, yugoslavia, el_salvador, trinadad&tobago, peru, hong, holand

Table 4.1: Attributes and their domains in the Adult dataset.

whether a person earns more than 50000$ or not; in Table 4.1 are reported the 14 attributes associated to each individual. The first tree we obtained with the decision tree algorithm has 143 levels and 8916 nodes. After pruning, the resulting tree has 67 levels and 804 nodes. It is still impossible to represent it here. It is clear that it is not a human-interpretable model to represent knowledge.

We tried to classify different instances, and in all cases we obtained the same classification from the decision tree and the abductive framework, but with sensible differences in the explanation of such results. Here we report two examples of classification:

age:	66	relationship:	husband
workclass:	private	race:	white
fnlwgt:	73522	sex:	male
education:	some_college	capital_gain:	0
education_num:	10	capital_loss:	0
marital_status:	married_civ_spouse	hours_per_week:	40
occupation:	sales	native_country:	*unknown*

Both systems rightly classify the instance as a person that earns less than 50000$, but the explanation is somewhat unclear in the decision tree model. In fact, by traversing the tree with the information we have about the instance at hand, we see that there is a unique path until the node labeled with "native_country", which has attached 10 leaves associated with "more than 50k" and 28 with "less than or equal to 50k" (with probability higher than 50%).

The abductive system gives the same classification results but with the following explanation:

```
age_ge_abd(35.5,0), fnlwgt_ge_abd(24965,0), fnlwgt_le_abd(388990,0),
education_num_le_abd(12.5,0), education_num_ge_abd(7.5,0),
marital_status_abd(married_civ_spouse,0), occupation_abd(sales,0),
capital_gain_le_abd(586.5,0), capital_loss_le_abd(209.5,0),
hours_per_week_ge_abd(34.5,0), hours_per_week_le_abd(71,0),
native_country_abd(united_states,3391)
```

In other words, the system abduces that the person is from united states, and this fact makes (by using just one classification rule/path) possible to infer "less than or equal to 50k".

Then we tried to classify the same instance without the information on the occupation attribute. In this case we obtain 62 possible paths in the decision tree for the instance we considered, each of them with 28 nodes in the average. The decision tree classifier leads to "less than or equal to 50k" since it is the most probable class among the leaves. Also the abductive system obtains the same result, but with the following explanation:

```
class_abd(le_50k,7841)
```

In this case the abductive system doesn't use any classification rule. The cost for the abduction of *occupation* = *sales* is 28911, that is not minimal. The meaning of such a classification explanation is that the information about the individual to classify are not enough to associate them to a specific target class. Therefore the most probable class is chosen.

age:	66	relationship:	husband
workclass:	private	race:	white
fnlwgt:	73522	sex:	male
education:	some_college	capital_gain:	0
education_num:	10	capital_loss:	0
marital_status:	married_civ_spouse	hours_per_week:	40
occupation:	*unknown*	native_country:	*unknown*

Classification Accuracy

Here we show some tests we have done in order to show the accuracy of our cost-based abductive classifier. The datasets used are: Iris, Vote, Soybean [HB99].

As usual, tests have been conducted by splitting the set of records into two different datasets: the training set and the test set. We created different splits in order to avoid bias due to a particular split. In the following we show some experimental results that shows that decision tree classification and abductive classification are almost equivalent w.r.t. classification accuracy.

Iris The Iris dataset has 1250 records, each of them labeled with 4 attributes and an attribute class out of 3 possible classes. All attribute values are numeric. The followings are three examples of records extracted from the dataset:

sepal length	**sepal width**	**petal length**	**petal width**	**iris type (class)**
5.1	3.5	1.4	0.2	iris setosa
4.9	3.0	1.4	0.2	iris setosa
4.7	3.2	1.3	0.2	iris setosa

Since Iris doesn't contain any missing value, the decision tree classifier and our cost-based abductive classifier are equivalent (they brings to the same results). Table 4.2 shows the accuracy of predicting the value of the class attribute.

training set	**support**	**confidence**	**AR**	**dtree**	**abduction**
50%	any	any	any	7 (9.33%)	7 (9.33%)
70%	any	any	any	1 (2.22%)	1 (2.22%)

Table 4.2: Accuracy of the class value prediction: decision tree classification and abductive classification are equivalent when there is no missing data in the test set.

Vote The dataset Vote has 436 records, and the value of the attribute *party* is the one to predict. Each record contains information about one voter, represented by the following 17 attributes:

handicapped infants
adoption of the budget resolution
el salvador aid
anti satellite test ban
mx missile
synfuels corporation cutback
superfund right to sue
duty free exports
party *(class attribute)*
water project cost sharing
physician fee freeze
religious groups in schools
aid to nicaraguan contras
immigration
education spending
crime
export administration act south africa

All the attributes are boolean, except for the class attribute, that has values in {*republican, democrat*}.

We show accuracy of decision tree based and abduction based classification systems in Table 4.3.

training set	**support**	**confidence**	**AR**	**dtree**	**abduction**
75%	90	90	0	8 errors (7.41%)	9 errors (8.33%)
75%	45	80	19	8 errors (7.41%)	9 errors (8.33%)
75%	43	80	42	8 errors (7.41%)	8 errors (7.41%)
75%	43	70	50	8 errors (7.41%)	8 errors (7.41%)
75%	40	80	96	8 errors (7.41%)	8 errors (7.41%)

Table 4.3: Accuracy of the class value for the *vote* dataset.

Soybean One may argue that a rule based classifier can describe results better than a decision tree based, and possibly improve classification results because of a smarter pruning strategy. Table 4.4 shows the opposite in the case of the Soybean dataset (307 record in the training set and 376 in the test set). Although decision tree outperforms the other classification models, abduction seems to behave better than rule based classification

decision tree	**rule based**	**abduction with no AR**
52 errors (13.83%)	118 errors (31.38%)	96 errors (25.53%)

Table 4.4: A comparison among decision tree, rule based and abduction based classification.

In Table 4.5 we reported the improvements of the abductive framework when we add new information (automatically, by association rule mining) in the knowledge base. We observe that association rules added to the knowledge base of the abductive

support	confidence	AR	dtree	abduction
90%	90%	2	52 errors (13.83%)	96 errors (25.53%)
85%	90%	11	52 errors (13.83%)	96 errors (25.53%)
83%	80%	22	52 errors (13.83%)	54 errors (14.36%)
81%	80%	47	52 errors (13.83%)	54 errors (14.36%)

Table 4.5: Accuracy improvements of the abductive classification by adding new association rules.

system make the system able to double the accuracy of the classification, obtaining results very similar to the decision tree based classifier.

4.6 Conclusions and Future Work

We showed how abductive reasoning can be useful in the context of classification, as a post processing step, in order to explain the reasons of a classification and to improve effectiveness, when we deal with incomplete data and with external domain knowledge.

This Section shows the approach we are following in order to join together different data mining paradigms such as Classification, Association Rules and Clustering through the use of a Constrained Abductive Framework.

4.6.1 Correlation Rules

The work presented so far included Association Rules both as integrity constraints inside the framework and as knowledge base rules for abductive proofs. We noticed that the usual framework in association rules analysis of Support/Confidence sometimes does not provide good results if the thresholds are not set up fine . The way we are planning to follow makes use of different metrics, in particular the lift, able to show correlation among attributes. Such rules are sometimes referred to in literature as Correlation Rules. It is an open problem to adapt the existing procedures able to find correlation rules to very large databases [HK00], and we are investigating also toward this direction.

This kind of correlation is very important in the field of abductive reasoning as they can better express the dependencies between causal events and effects.

4.6.2 Clustering

The idea of using clustering here is that of finding similarities between rules and then, through abduction, showing the differences between rules in the same cluster. The main difficulty in this task could be the definition of a metric able to compare different rules (see [LSW97, GSG99]).

Once we have a good metric, we can obtain through clustering several sets of rules. Fixing a set and comparing "similar" rules (i.e., similar tails) with different heads we can abductively infer the reason why the rules have different heads.

The scenario here is the problem of finding an answer to the following question:

Given a dataset of examples (with the right classification for each example), which attribute values should I try to change in order to obtain a different classification?

We think that with an accurate choice of the metric (and the associated algorithm) for clustering, together with a right abductive formulation of the problem, it is substantially possible to answer these kinds of questions.

One way to approach this problem (after the rules have been clustered) could be by automatically adding to the theory an abducible for each change we can do.

Example 6. Let us suppose our goal is to know which attribute values to change in order to classify as Y examples which are classified as X. The first pass consists in clustering all the ruleset. The second is to analyze one by one all the clusters. Consider for example the following cluster of rules:

A(a), B(b), C(c), D(d1), E(e1) → Y
A(a), B(b), C(c), D(d2), E(e2) → X
A(a), B(b), C(c), D(d1), E(e2) → X

As we can see, the cluster contains similar rules (in this case we have rules with $\{A(a), B(b), C(c)\}$ in the antecedent). Now we can specify which kind of changes we want to consider. In fact, we don't want or we cannot change some attribute values (for example, consider the attribute Age of the customers of a market). We can call them *canonical rules* and *user abducibles.*

Rules:
ChangeD(X,Y) → D(X), D(Y)
ChangeE(X,Y) → E(X), E(Y)

Abducibles: $ChangeD(X, Y), ChangeE(e2, e1)$

Notice that the canonical rules can be automatically added into the theory. The user abducibles instead (as the name suggests) have to be inserted by the user, in order to specify which kind of changes are enabled (in the case at hand we can modify the value of D and also the value of E, but only from $e2$ to $e1$).

Now we can consider the integrity constraints. In the beginning of this Chapter we considered the so-called canonical constraints, such as:

IC: $D(X), D(Y), X \neq Y$.

For this problem we need different 'canonical' integrity constraints:

IC: $ChangeD(X1, Y1), ChangeD(X2, Y2), \neg((X1 = X2 \wedge Y1 = Y2) \vee (X1 = Y2 \wedge Y1 = X2))$

We can have that both $D(d1)$ and $D(d2)$ hold, but in this case we have to abduce $ChangeD(d1, d2)$.

The idea is that frequent attribute values (in the example above, $\{A(a), B(b), C(c)\}$) can be considered true, i.e., they are added to the theory. Canonical rules and user abducibles also belong to the abductive theory. Then we look for an abductive explanation for $\{Y, X\}$ and we will find $\{ChangeD(d2, d1), ChangeE(e2, e1)\}$.

The approach described in the example above needs to be accurately extended in order to avoid some unwanted behavior such as the inference of predicates like $Change(e1, e1)$. Anyway, we think that it is possible to describe an abductive framework able to find what changes to do in order to get some specified results.

4.6.3 Abductive Relational Data Mining

The classification problem is one of the most important topic belonging to the field of Machine Learning [BD99] that consists, as we already said, in predicting the class of an individual given the characteristics of such individual and a set of examples of right and wrong classifications of other known individuals. So there is a mix of descriptive and predictive power in this problem.

Machine learning works have produced a lot of good algorithms able to classify new individuals very well according to the given examples (i.e. see [Qui93]), but the description of both examples and new individuals can be given only using poor languages, usually based on the idea of attributes. An individual can be described only in a flat way, e.g. consisting of attributes and their values, other than the class value.

In order to upgrade the language used to represent hypothesis and examples to the first order case, it is important to decide how exactly learning in first-order logic generalizes attribute-value learning. Some perspective about that have been proposed in literature [DR98, ZG98]. Currently, most of the state of the art systems use first order terms to give structure to individuals. In the propositional case individuals can be viewed as tuples of attribute values. In a first order language it is also possible to describe individuals with complex types at the top-level (e.g. sets, list) and at the intermediate levels before the atomic enumerated types are reached.

Most of the works in literature belong to the field of Inductive Logic Programming. Anyway, most of the results can be easily adapted to be embedded into an Abductive Framework.

In order to understand the powerfulness of first order description of individuals, let consider the following well known example.

Example 7. In the context of trains, let consider the Michalski's east- and westbound trains learning problem. At the bottom level we have some propositional attributes:

$Shape = \{rectangle, bucket, hexa, \ldots\}$
$Length = \{double, short\}$
$Roof = \{flat, jagged, peaked, arc, open\}$
$Load = \{circle, hexagon, triangle, \ldots\}$

A car is a 5-tuple describing its shape, length, number of wheels, type of roof an its load:

$Car = Shape \times Length \times Integer \times Roof \times Load$

The individual representing cars can be obviously described using the attributional language. But this learning problem concerns trains, which are defined as set of cars:

$Train = 2^{Car}$

An individual to be classified can be, for instance, the following:

$\{(bucket, short, 2, open, triangle), (rectangle, short, 2, flat, circle)\}$

Actually it is possible decompose the probability of an individual (a car in the above example) into those of its propositional attributes, or get the probability of a car counting the occurrence of such a car in the training set instead of the occurrence of its attributes. Until now it is not clear which is the best choice to generalize the propositional case.

The use of Logic Programming techniques for inductive tasks has raised the end of the enhancements on the descriptive side of the classification problem, using a subset of first order logic as its hypothesis description language. This approach, unfortunately, results in a vast search space. It was shown in the past that attributional learners often achieved better results than their relational siblings, even when latter were given more informative background knowledge. Part of the reason is probably that attributional learners, faced with smaller hypothesis space, explore the space more thoroughly. The second reason might be that propositional learners take into account additional information by probabilistically analyzing the hypothesis over the training set.

In this context, Relational Data Mining (i.e., data mining in a first-order description language) seems to be a good mix between Logic Programming descriptive power and relative small search space due to the use of probabilities. We think that probabilistic abductive frameworks can be more profitably used in a domain knowledge described in an non-flattened way, such as a first order language.

4.6.4 Concluding Remarks

Our results showed that Abductive frameworks can be seen as a layer able to accept user knowledge and requests in an high level language and then to compute the solution. We started from the idea that framing the data mining step into a logic framework provides us with the possibility of exploiting also domain knowledge in the knowledge extraction and exploitation process, then we showed that the result of a classification tree can be improved if the tree is not simply traversed, but it is visited in abductive mode.

Indeed, the basic observation is that the application of a classification tree to a new observation can be viewed as a straightforward abductive computation, as soon as the tree is represented as a collection of rules in the obvious way. This observation opens up a world of possibilities, since the abductive computations can become very sophisticated, and they can take into account several types of knowledge.

Chapter 5

Anonymity-Preserving Frequent Set Mining

Abstract

In this Chapter we illustrate a new approach to privacy-preserving data mining. By shifting and extending the k-anonymity concept from databases to pattern discovery, we develop a formal way to define anonymity breaches raised by the data mining results. We also give algorithms to discover such breaches and remove them from the final results, without compromising the quality of the output.

We concentrate of frequent patterns mining, and study anonymity in this setting. The key question is: can anonymity be guaranteed when a collection of pattern resulting from a data mining computation is disclosed? At a first sight, one might guess that each frequent pattern is related to a relatively large group of individuals in the source database, and therefore non-identifiability is guaranteed. Unfortunately, this is not the case: a malicious adversary can reason on the collection of frequent patterns, and deduce new patterns from the available ones, precisely identifying individuals or small groups. In other words it is not sufficient to constrain each pattern to be anonymous by itself. In fact we show how also the combination of patterns allows to infer new derived patterns, which may violate anonymity and hence break the non-identifiability requirement.

5.1 Contribution and Organization

In this Chapter we study the anonymity problem in the very general setting of patterns which are boolean formulas over a binary database.

A major contribution of our work is given in Section 5.2 where, after recalling association rule mining, we show by means of a simple example, that indeed the disclosure of a set of patterns produced by data mining computations may violate anonymity of individuals recorded in the source data. This simple example is the driving idea of our work, and thus it is very useful to make clear the objectives pursued in this Chapter.

In Section 5.3 we briefly recall the works on k-anonymity on databases, and discuss the benefits of shifting such concepts from the data to the extracted patterns. Such discussion leads to the formal definition of *k-anonymous patterns* in Section 5.4. Here we characterize k-anonymous patterns, and we study the possible channels of *inference* in a collection of patterns that may be exploited by a malicious adversary to threat anonymity of the individuals recorded in the source data. The formal definition of the anonymity preservation problem as a problem of logical inference is one of the major contributions of this thesis.

Next, two practical problems are addressed: how to detect the inference channels in a collection of patterns, and how to block them. In Section 5.5 is defined a first naïve algorithm to detect such potential threats which yields a methodology to check whether the mining results may be disclosed without any risk of violating anonymity. It should be noted that the capability of detecting the potential threats is extremely useful for the analyst to determine a trade-off among the quality of mining result and the privacy guarantee.

In Section 5.6 we introduce a condensed representation of the set of inference channels. A condensed representation is a subset of the original collection of patterns which contains the same information. The condensed representation we define has a twofold benefit, since it helps both the detecting and the blocking task. On one hand it reduces the computational cost of the detection task, yielding to an improved algorithm presented in Section 5.6. On the other hand, exploiting the condensed representation we avoid redundant sanitization and thus our blocking algorithms (developed in Section 5.7), geared on the condensed representation, introduce less distortion.

In Section 5.8 we report the experimental analysis that we have conducted in order to assess the distortion introduced by our sanitization strategies; to measure time needed by our sanitization framework and to compare empirically the differences between k-anonymizing the data and k-anonymizing the patterns. Finally, in Section 5.9 we describe some on-going and future works and we conclude.

We are aware that real solutions to the challenges posed in this Chapter can only be achieved through a combination of technical tools, legal regulations and social norms. On one side, a regulatory context poses challenges and constraints for novel technical solutions; in turn, new solutions might provide feedback and opportunities toward better norms and regulations.

The results described here can be considered as a preliminary step along a path that is crucial, both from the ethical point of view and that of social acceptance – data mining solutions that are not fully trustworthy will find insuperable obstacles to their deployment. On the other hand, demonstrably trustworthy solutions may open up tremendous opportunities for new knowledge-based applications of public utility and large societal and economic impact.

5.2 Data Mining Results Can Violate Anonymity

At a first sight, it may seem that data mining results do not violate the anonymity of the individuals recorded in the source database. In fact, data mining models and patterns, in order to ensure a required statistical significance, represent a large number of individuals and thus conceal individual identities. In the following we show that this belief is ill-founded, using *association rules* as prototypical example.

The idea of mining association rules [AIS93] originates from the analysis of *market-basket* data where we are interested in finding rules describing customers behavior in buying products. In particular, association rules seek for sets of products which are associated, in the sense that they are bought together quite frequently. Their direct applicability to business problems together with their inherent understandability, even for non data mining experts, made association rules a popular mining method.

As we already described in Chapter 1 an association rule is an expression $X \Rightarrow Y$ where X and Y are two sets of items. The association rule is said to be *valid* if:

1. the *support* of the itemset $X \cup Y$, i.e., the number of transactions in the database in which the set $X \cup Y$ appears, is greater than a given threshold;

2. the *confidence* (or accuracy) of the rule, defined as the conditional probability $P(Y \mid X)$, i.e., the support of $X \cup Y$ over the support of Y, is greater than a given threshold.

Since association rules in order to be valid must be common to a large number of individuals, i.e., their support must be larger than a given threshold, we might be tempted to conclude that, if such a threshold is large enough, we can always safely disclose the extracted association rules.

The next example shows that this is not true.

Example 1. Consider the following association rule:

$$a_1 \wedge a_2 \wedge a_3 \Rightarrow a_4 \quad [sup = 80,\ conf = 98.7\%]$$

where *sup* and $conf$ are the usual interestingness measures of *support* and *confidence* as defined above. Since the given rule holds for a number of individuals (80), which seems large enough to protect individual privacy, one could conclude that the given rule can be safely disclosed. But, is this all the information contained in such a rule? Indeed, one can easily derive the support of the premise of the rule:

$$sup(\{a_1, a_2, a_3\}) = \frac{sup(\{a_1, a_2, a_3, a_4\})}{conf} \approx \frac{80}{0.987} \approx 81.05$$

Given that the pattern $a_1 \wedge a_2 \wedge a_3 \wedge a_4$ holds for 80 individuals, and that the pattern $a_1 \wedge a_2 \wedge a_3$ holds for 81 individuals, we can infer that in our database there is just one individual for which the pattern $a_1 \wedge a_2 \wedge a_3 \wedge \neg a_4$ holds. The knowledge inferred is a clear threat to the anonymity of that individual: on one hand the pattern identifying the individual could itself contain sensitive information; on the other hand it could be used to re-identify the same individual in other databases.

It is worth noting that this problem is very general: the given rule could be, instead of an association, a classification rule, or the path from the root to the leaf in a decision tree, and the same reasoning would still hold. Moreover, it is straightforward to note that, unluckily, the more accurate is a rule, the more unsafe it may be w.r.t. anonymity.

We say that the two itemsets $\{a_1, a_2, a_3\}$ and $\{a_1, a_2, a_3, a_4\}$ represent an *inference channel*, for the anonymity of the individual corresponding to the pattern $a_1 \wedge a_2 \wedge a_3 \wedge \neg a_4$. This is a trivial kind of inference channel, but in general, much more complex kinds of inference channels exist, as studied in the following sections in this Chapter.

5.3 k-Anonymity: from Data to Patterns

When the objective of a data owner is to disclose the data, k-*anonymity* is an important method for protecting the privacy of the individuals recorded in the data. The concept of k-anonymity was introduced by Samarati and Sweeney in [SS98, Swe02a]. In these works, it is shown that protection of individual sources does not guarantee protection when sources are cross-examined: a sensitive medical record, for instance, can be uniquely linked to a *named* voter record in a publicly available voter list through some shared attributes. A recent study estimated that 87% of the population of the United States can be uniquely identified using the attributes gender, date of birth and 5-digit zip code [Swe00]. In fact, those three attributes were used to link Massachusetts voter registration records (which included the name, gender, zip code, and date of birth) to a supposedly anonymized medical dataset. This cross linkage attack managed to uniquely identify the medical records of the governor of Massachusetts in the medical data [Swe02a].

The objective of k-anonymity is to eliminate such opportunities of inferring private information through cross linkage. According to this approach, the data holder identifies all possible attributes in the private information that can be found in other public databases, and thus could be exploited by a malicious adversary by means of cross linkage. Sets of attributes (like gender, date of birth, and zip code in the example above) that can be linked with external data to uniquely identify individuals in the population are called *quasi-identifiers*. Once the quasi-identifiers are known, a "sanitization" of the source data takes place: the data is transformed in such a way that, for every combination of values of the quasi-identifiers in the sanitized data, there are at least k records that share those values. Such a sanitization is obtained by generalization of attributes (the quasi-identifiers) and, when needed, suppression of tuples [Swe02b].

As stated in the Introduction, in this Chapter we do not study how to safely disclose data, but instead we focus on the disclosure of patterns extracted by means of data mining. In our context the data owner is not willing to share the data – on the contrary, it is often legally responsible for protecting the data – and, instead, it is interested in publishing the knowledge discovered by mining the data. Therefore, a malicious adversary could only attack privacy exploiting the information which is present in the patterns, plus his own background knowledge about the technique used to extract the patterns.

A pattern produced by data mining techniques can be seen as a `select` query, which returns the set of tuples in the database which are captured by the given pattern. Thus we can shift the concept of k-anonymity from the data to the patterns in a straightforward way: we say that the result of a data mining extraction is k-anonymous if from any pattern inferred from such results is not possible to identify a group of tuples of cardinality less than k. More precisely:

- a single pattern p with support count $s > 0$ (i.e., occurring s times in the source database) is k-anonymous iff $s \geq k$, i.e., there are at least k tuples in the database satisfying p;

- a collection of patterns, each with support count, is k-anonymous iff each pattern in it is k-anonymous as well as any further pattern whose support can be inferred from the collection.

In the following we study the problem of how to produce a set of patterns that are k-anonymous, and of course, close as possible to the real patterns holding in the data. We set our investigations in the very general context where the source data is a binary database, and the kind of patterns extracted (and disclosed) are *frequent itemsets*, i.e., sets of attributes which appear all set to 1 in a number of tuples larger than a given frequency threshold. Therefore, with the aim of facilitating the theoretical investigation, and for sake of clarity of the presentation, for the moment we do not consider the semantics of the attributes, and the possibility of generalizing them. Moreover we do not distinguish between quasi-identifiers and other attributes: in our context all attributes are quasi-identifier. Note that these two assumptions do not weaken our contribution; on the contrary we develop a very general theory that could be easily instantiated to the more concrete case of categorical data originating from relational tables. Introducing the distinction between quasi-identifiers and non, as well as the possibility of generalizing some attributes, will just make our patterns sanitization task easier, and would reduce the amount of distortion needed to k-anonymize the patterns.

Finally, notice that a trivial solution to our problem would be to first k-anonymize the data using some well known technique, and then mine the patterns from the k-anonymized data. In fact, by mining a k-anonymized database no patterns threatening anonymity can be obtained. But such approach would produce patterns impoverished by the information loss which is intrinsic in the generalization and suppression techniques. Since our objective is to extract valid and interesting patterns, we propose to postpone k-anonymization after the actual mining step. In other words, we do not enforce k-anonymity onto the source data, but instead we move such a concept to the extracted patterns. Since there is a clear correspondence between patterns and data, k-anonymizing the patterns can be seen as k-anonymizing just the portion of interest of data, the portion corresponding to the patterns. Following this way we introduce much less distortion. This issue will be further analyzed in Section 5.8.3.

5.4 k-Anonymous Patterns

We start by defining binary databases and patterns following the notation in [HMS01].

Definition 1 (Binary Database). *A binary database $\mathcal{D} = (\mathcal{I}, \mathcal{T})$ consists of a finite set of binary variables $\mathcal{I} = \{i_1, \ldots, i_p\}$, also known as* items, *and a finite multiset $\mathcal{T} = \{t_1, \ldots, t_n\}$ of p-dimensional binary vectors recording the values of the items. Such vectors are also known as* transactions.

Definition 2 (Pattern). *A pattern for the variables in $\mathcal{I}$ is a logical (propositional) sentence built by AND ($\wedge$), OR ($\vee$) and NOT ($\neg$) logical connectives, on variables in $\mathcal{I}$. The domain of all possible patterns is denoted $\mathcal{P}at(\mathcal{I})$.*

A binary database $\mathcal{D}$ is given in Figure 5.1(a). In this context, a row or transaction of $\mathcal{D}$ is a tuple recording the values of some attributes (or items) of an individual. Therefore in this context, the objective of our analysis is the anonymity of transactions.

According to Definition 2, $e \wedge (\neg b \vee \neg d)$, where $b, d, e \in \mathcal{I}$, is a pattern. One of the most important properties of a pattern is its frequency in the database, i.e. the number of individuals (transactions) in the database which make the given pattern true[1].

Definition 3 (Support). *Given a database $\mathcal{D}$, a transaction $t \in \mathcal{D}$ and a pattern p, we write $p(t)$ if t makes p true. The support of p in $\mathcal{D}$ is given by the number of transactions which makes p true:*

$$sup_{\mathcal{D}}(p) = |\{t \in \mathcal{D} \mid p(t)\}|.$$

If for a given pattern this number is very low (i.e. smaller than an anonymity threshold k) but not null, then the pattern represents a threat for the anonymity of the individuals about which the given pattern is true.

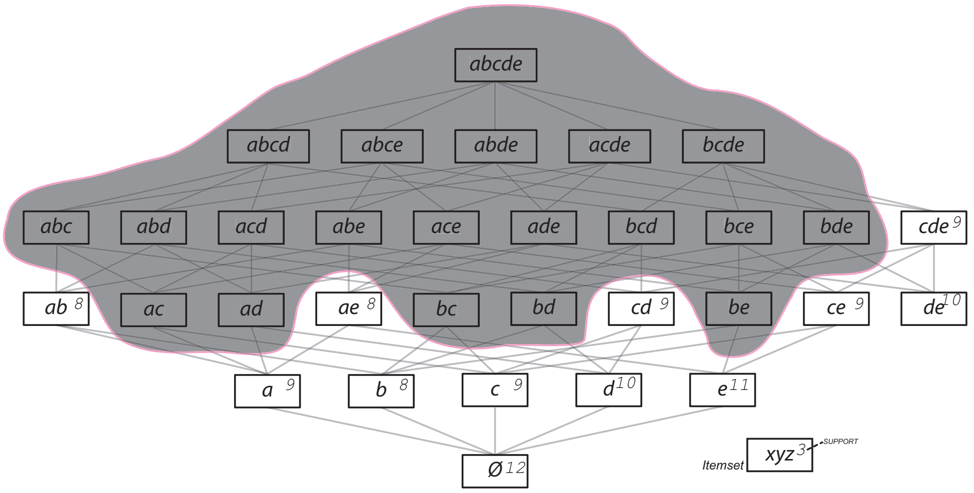
Definition 4 (k-Anonymous Pattern). *Given a binary database $\mathcal{D}$ and an anonymity threshold k, a pattern p is said to be k-*anonymous *if $sup_{\mathcal{D}}(p) \geq k$ or $sup_{\mathcal{D}}(p) = 0$.*

[1] The notion of truth of a pattern w.r.t. a transaction t is defined in the usual way: t makes p true iff t is a model of the propositional sentence p.

$\mathcal{D}$

	a	b	c	d	e	f	g	h
t_1	1	1	1	1	1	1	1	1
t_2	1	1	1	1	1	0	1	0
t_3	1	1	1	1	1	0	0	0
t_4	1	1	1	1	1	1	1	0
t_5	1	1	1	1	1	0	0	0
t_6	1	1	1	1	1	0	0	0
t_7	1	1	0	1	1	0	0	0
t_8	1	0	0	0	1	1	1	0
t_9	0	0	1	1	1	1	1	0
t_{10}	0	0	1	1	1	0	0	0
t_{11}	0	0	1	1	1	1	1	1
t_{12}	1	1	0	0	0	1	1	0

(a)



(b)

Figure 5.1: Running example: (a) the binary database $\mathcal{D}$; and (b) a graphical representation of the lattice of itemsets $2^{\mathcal{I}}$ for $\mathcal{I} = \{a, b, c, d, e\}$: the set of σ-frequent ($\sigma = 8$) itemsets over $\mathcal{D}$ is displayed together with their supports (i.e., $\mathcal{F}(\mathcal{D}, 8)$). This is what is disclosed, i.e., the whole information that a malicious adversary can use, while the gray area represents what is not known. The singleton items f, g and h, and all their supersets, are not displayed: since they are infrequent, the adversary can not even know that these items (or attributes) are present in the source data.

Notation: patterns	**Notation: itemsets**
$sup_{\mathcal{D}}(a \vee f) = 11$	$sup_{\mathcal{D}}(abc) = 6$
$sup_{\mathcal{D}}(e \wedge (\neg b \vee \neg d)) = 4$	$sup_{\mathcal{D}}(abde) = 7$
$sup_{\mathcal{D}}(h \wedge \neg b) = 1$	$sup_{\mathcal{D}}(cd) = 9$

Table 5.1: Notation: example of general patterns and itemsets with their support in the database $\mathcal{D}$ of Figure 5.1(a).

Our objective is to study when the output of a mining extraction could be exploited by a malicious adversary to identify *non* k-anonymous patterns, i.e., small groups of individuals. In particular, we focus on *frequent itemset mining*: one of the most basilar and fundamental mining tasks.

Itemsets are a particular class of patterns: conjunctions of positive valued variables, or in other words, sets of items. The retrieval of itemsets which satisfy a minimum frequency property is the basic step of many data mining tasks, including (but not limited to) association rules [AIS93, AS94].

Definition 5 (σ-Frequent Itemset). *The set of all itemsets $2^{\mathcal{I}}$, is a pattern class consisting of all possible conjunctions of the form $i_1 \wedge i_2 \wedge \ldots \wedge i_m$. Given a database $\mathcal{D}$ and a minimum support threshold σ, the set of σ-frequent itemsets in $\mathcal{D}$ is denoted:*

$$\mathcal{F} = \{\langle X, sup_{\mathcal{D}}(X)\rangle \mid X \in 2^{\mathcal{I}} \wedge sup_{\mathcal{D}}(X) \geq \sigma\}.$$

Frequent Itemset Mining (FIM), i.e., computing $\mathcal{F}$, is one of the most studied algorithmic problems in data mining: hundreds of algorithms have been developed and compared (see [Goe] for a good repository) since the first proposal of the well-known APRIORI algorithm [AS94]. As shown in Figure 5.1(b), the search space of the FIM problem is a lattice, which is typically visited breadth-first or depth-first, exploiting the following interesting property: $\forall X \subset Y \in 2^{\mathcal{I}}.sup_{\mathcal{D}}(X) \geq sup_{\mathcal{D}}(Y)$.

This property, also known as *"anti-monotonicity of frequency"* or *"Apriori trick"*, is exploited by the APRIORI algorithm (and by almost all FIM algorithms) with the following heuristic: if an itemset X does not satisfy the minimum support constraint, then no superset of X can be frequent, and hence they can be pruned from the search space. This pruning can affect a large part of the search space, since itemsets form a lattice.

Itemsets are usually denoted in the form of set of the items in the conjunction, e.g. $\{i_1, \ldots, i_m\}$; or sometimes, simply $i_1 \ldots i_m$. Table 5.1 shows the different notation used for general patterns and for itemsets.

Example 2. Given the binary database $\mathcal{D}$ in Figure 5.1(a), and a minimum support threshold $\sigma = 8$, we have that:

$$\begin{aligned}\mathcal{F}(\mathcal{D}, 8) = \{ \quad & \langle\emptyset, 12\rangle, \langle a, 9\rangle, \langle b, 8\rangle, \langle c, 9\rangle, \langle d, 10\rangle, \langle e, 11\rangle, \\ & \langle ab, 8\rangle, \langle ae, 8\rangle, \langle cd, 9\rangle, \langle ce, 9\rangle, \langle de, 10\rangle, \langle cde, 9\rangle \quad \}\end{aligned}$$

As already stated, the problem addressed is given by the possibility of inferring from the output of frequent itemset mining, i.e, $\mathcal{F}$, the existence of patterns with very low support (i.e., smaller than an anonymity threshold k, but not null): such patterns represent a threat for the anonymity of the individuals about which they are true.

Recall our motivating example: from the two disclosed frequent itemsets $\{a_1, a_2, a_3\}$ and $\{a_1, a_2, a_3, a_4\}$ (and their supports) it was possible to infer the existence of the *non* k-anonymous pattern $a_1 \wedge a_2 \wedge a_3 \wedge \neg a_4$.

Informally, we call *inference channel* any collection of itemsets (with their respective supports), from which it is possible to infer non k-anonymous patterns. In the following we formally define and characterize inference channels.

5.4.1 Inference Channels

Before introducing our anonymity preservation problem, we need to define the inference of supports, which is the basic tool for the attacks to anonymity.

Definition 6 (Database Compatibility). *A set S of pairs $\langle X, n\rangle$, where $X \in 2^{\mathcal{I}}$ and $n \in \mathbb{N}$, and a database $\mathcal{D}$ are said to be compatible if $\forall \langle X, n\rangle \in S . sup_{\mathcal{D}}(X) = n$.*

Definition 7 (Support Inference). *Given a set S of pairs $\langle X, n\rangle$, where $X \in 2^{\mathcal{I}}$ and $n \in \mathbb{N}$, and a pattern $p \in \mathcal{P}at(\mathcal{I})$ we say that $S \models sup(p) > x$ (respectively $S \models sup(p) < x$) if, for all databases $\mathcal{D}$ compatible with S, we have that $sup_{\mathcal{D}}(p) > x$ (respectively $sup_{\mathcal{D}}(p) < x$).*

Definition 8 (Inference Channel). *An inference channel $\mathcal{C}$ is a set of pairs $\langle X, n\rangle$, where $X \in 2^{\mathcal{I}}$ and $n \in \mathbb{N}$, such that:*

$$\exists p \in \mathcal{P}at(\mathcal{I}) : \mathcal{C} \models 0 < sup(p) < k.$$

As suggested by Example 1, a simple inference channel is given by any itemset X which has a superset $X \cup \{a\}$ such that $0 < sup_{\mathcal{D}}(X) - sup_{\mathcal{D}}(X \cup \{a\}) < k$. In this case the pair $\langle X, sup_{\mathcal{D}}(X)\rangle, \langle X \cup \{a\}, sup_{\mathcal{D}}(X \cup \{a\})\rangle$ is an inference channel for the non k-anonymous pattern $X \wedge \neg a$, whose support is directly given by $sup_{\mathcal{D}}(X) - sup_{\mathcal{D}}(X \cup \{a\})$. This is a trivial kind of inference channel.

Do more complex structures of itemsets exist that can be used as inference channels?

In general, the support of a conjunctive pattern $p = i_1 \wedge \cdots \wedge i_m \wedge \neg a_1 \wedge \cdots \wedge \neg a_n$ can be inferred if we know the support of itemsets $I = \{i_1, \ldots, i_m\}$, $J = I \cup \{a_1, \ldots, a_n\}$, and every itemset L such that $I \subset L \subset J$.

Lemma 1. *Given a pattern $p = i_1 \wedge \cdots \wedge i_m \wedge \neg a_1 \wedge \cdots \wedge \neg a_n$ we have that:*

$$sup_{\mathcal{D}}(p) = \sum_{I \subseteq X \subseteq J} (-1)^{|X \setminus I|} sup_{\mathcal{D}}(X)$$

where $I = \{i_1, \ldots, i_m\}$ and $J = I \cup \{a_1, \ldots, a_n\}$.

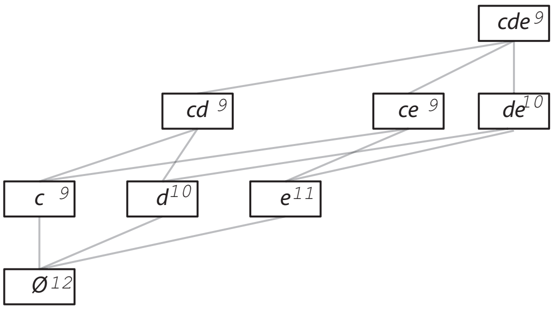


Figure 5.2: A detail of Figure 5.1(b): this is an inference channel, (i.e, $\mathcal{C}_{\emptyset}^{cde}$ for $k = 3$).

Proof. (Sketch) The proof follows directly from the definition of support and the well-known *inclusion-exclusion principle* [Knu97]. □

Following the notation in [CG02], we denote the right-hand side of the equation above as $f_I^J(\mathcal{D})$.

Example 3. In the database $\mathcal{D}$ in Figure 5.1(a) we have that $sup_{\mathcal{D}}(b \wedge \neg d \wedge \neg e) = f_b^{bde}(\mathcal{D}) = sup_{\mathcal{D}}(b) - sup_{\mathcal{D}}(bd) - sup_{\mathcal{D}}(be) + sup_{\mathcal{D}}(bde) = 8 - 7 - 7 + 7 = 1$.

Definition 9. *Given a database $\mathcal{D}$, and two itemsets $I, J \in 2^{\mathcal{I}}$, $I = \{i_1, \ldots, i_m\}$ and $J = I \cup \{a_1, \ldots, a_n\}$, if $0 < f_I^J(\mathcal{D}) < k$, then the set*

$$\{\langle X, sup_{\mathcal{D}}(X)\rangle \mid I \subseteq X \subseteq J\}$$

constitutes an inference channel for the non k-anonymous pattern $p = i_1 \wedge \cdots \wedge i_m \wedge \neg a_1 \wedge \cdots \wedge \neg a_n$. We denote such inference channel $\mathcal{C}_I^J$ and we write $sup_{\mathcal{D}}(\mathcal{C}_I^J) = f_I^J(\mathcal{D})$.

Example 4. Consider the database $\mathcal{D}$ of Figure 5.1(a), and suppose $k = 3$. We have that $\mathcal{C}_{\emptyset}^{cde}$ is an inference channel of support 1. In fact we got that:
$sup_{\mathcal{D}}(\mathcal{C}_{\emptyset}^{cde}) = f_{\emptyset}^{cde}(\mathcal{D}) = sup_{\mathcal{D}}(\emptyset) - sup_{\mathcal{D}}(c) - sup_{\mathcal{D}}(d) - sup_{\mathcal{D}}(e) + sup_{\mathcal{D}}(cd) + sup_{\mathcal{D}}(ce) + sup_{\mathcal{D}}(de) - sup_{\mathcal{D}}(cde) = 12 - 9 - 10 - 11 + 9 + 9 + 10 - 9 = 1$.

This means that there is only one transaction $t \in \mathcal{D}$ is such that $\neg c \wedge \neg d \wedge \neg e$ (transaction t_{12}). A graphical representation of this channel is given in Figure 5.2.

The next Theorem states that if there exists a non k-anonymous pattern, then there exists a pair of itemsets $I \subseteq J \in 2^{\mathcal{I}}$ such that $\mathcal{C}_I^J$ is an inference channel.

Theorem 1.

$$\forall p \in \mathcal{P}at(\mathcal{I}) : 0 < sup_{\mathcal{D}}(p) < k \,.\, \exists\, I, J \in 2^{\mathcal{I}} : \mathcal{C}_I^J$$

Proof. Let us consider a generic pattern $p \in \mathcal{P}at(\mathcal{I})$. Without loss of generality p is in *normal disjunctive form*: $p = p_1 \vee \ldots \vee p_q$, where each $p_1, \ldots, p_q$ is a conjunctive pattern, i.e., a pattern made only by conjunction and negation as the one in Lemma 1. We have that:

$$sup_{\mathcal{D}}(p) \geq \max_{1 \leq i \leq q} sup_{\mathcal{D}}(p_i).$$

Since $sup_{\mathcal{D}}(p) < k$ we have for all patterns p_i that $sup_{\mathcal{D}}(p_i) < k$. Moreover, since $sup_{\mathcal{D}}(p) > 0$ there is at least a pattern p_i such that $sup_{\mathcal{D}}(p_i) > 0$. Therefore, there is at least a conjunctive pattern p_i such that $0 < sup_{\mathcal{D}}(p_i) < k$.

From Lemma 1, we have that $\exists\, I \subseteq J \in 2^{\mathcal{I}} : sup_{\mathcal{D}}(p_i) = f_I^J(\mathcal{D})$. Since $0 < sup_{\mathcal{D}}(p_i) = f_I^J(\mathcal{D}) < k$ we have that $\mathcal{C}_I^J$ is an inference channel. □

Corollary 1. *Given a database $\mathcal{D}$, and an anonymity threshold k:*

$$\nexists\, I, J \in 2^{\mathcal{I}} : \mathcal{C}_I^J \Rightarrow \nexists p \in \mathcal{P}at(\mathcal{I}) : 0 < sup_{\mathcal{D}}(p) < k$$

5.5 Detecting Inference Channels

The problem addressed in the following is the detection of anonymity threats in the output of a frequent itemset extraction. From Corollary 1 we can conclude that by detecting and sanitizing all inference channels of the form $\mathcal{C}_I^J$, we can produce a k-anonymous output which can be safely disclosed. However, when we instantiate the general theory above to the concrete case in which we want to disclose a set of frequent itemsets (.i.e., $\mathcal{F}$ and not the whole $2^{\mathcal{I}}$), the situation is made more complex by the frequency threshold. In fact, frequency divides the lattice of itemsets $2^{\mathcal{I}}$ in two parts: the frequent part which is disclosed, and the infrequent part (i.e., the gray area in Figure 5.1(b)) which is not disclosed.

This division induces a distinction also on the kind of patterns:

- patterns made only by composing pieces of $\mathcal{F}$;
- patterns made also using infrequent itemsets.

The following definition precisely characterize the first kind.

Definition 10 (σ-vulnerable Pattern). *Given a general pattern $p \in \mathcal{P}at(\mathcal{I})$, we can assume without loss of generality that p is in normal disjunctive form: $p = p_1 \vee \ldots \vee p_q$, where each p_i is a conjunctive pattern. Given a database $\mathcal{D} = (\mathcal{I}, \mathcal{T})$ and a minimum support threshold σ, we define $\mathcal{P}at(\mathcal{D}, \sigma) \subseteq \mathcal{P}at(\mathcal{I})$ as the set of patterns $p = p_1 \vee \ldots \vee p_q$ such that, for each conjunctive pattern p_i, if we consider the set of all items in it, i.e., the itemset obtained removing the negation symbols from it, such itemset is frequent. We call σ-vulnerable each pattern in $\mathcal{P}at(\mathcal{D}, \sigma)$.*

Example 5. Given a database $\mathcal{D}$ and a minimum support threshold σ the pattern $(a \wedge \neg b) \vee (d \wedge \neg c)$ is in $\mathcal{P}at(\mathcal{D}, \sigma)$ if $\langle ab, sup_{\mathcal{D}}(ab)\rangle \in \mathcal{F}$ and $\langle cd, sup_{\mathcal{D}}(cd)\rangle \in \mathcal{F}$. In the same case the pattern $a \wedge d \wedge \neg c$ is not in $\mathcal{P}at(\mathcal{D}, \sigma)$ if $\langle acd, sup_{\mathcal{D}}(acd)\rangle \notin \mathcal{F}$.

The interest in the σ-vulnerable patterns lies in the consideration that a malicious attack starts from the delivered knowledge, i.e, the frequent itemsets.

We next project Theorem 1 on this class of patterns.

Theorem 2. *Given a database $\mathcal{D}$, a minimum support threshold σ and an anonymity threshold k, we have that $\forall p \in \mathcal{P}at(\mathcal{D}, \sigma) : 0 < sup_{\mathcal{D}}(p) < k$ there exist two itemsets $I, J \in 2^{\mathcal{I}}$ such that $\langle I, sup_{\mathcal{D}}(I)\rangle, \langle J, sup_{\mathcal{D}}(J)\rangle \in \mathcal{F}$ and $\mathcal{C}_I^J$.*

Proof. (Sketch) The proof follows directly from Theorem 1 and Definition 10. □

Corollary 2. *Given a database $\mathcal{D}$, and an anonymity threshold k:*

$$\nexists \langle I, sup_{\mathcal{D}}(I)\rangle, \langle J, sup_{\mathcal{D}}(J)\rangle \in \mathcal{F} : \mathcal{C}_I^J \Rightarrow \nexists p \in \mathcal{P}at(\mathcal{D}, \sigma) : 0 < sup_{\mathcal{D}}(p) < k.$$

Therefore, if we know that the set of frequent itemsets $\mathcal{F}$ that we have extracted does not contain any inference channel of the form $\mathcal{C}_I^J$, we can be sure that a malicious adversary will never infer from it a *non* k-anonymous, σ-vulnerable pattern. In the rest of the Chapter we focus on this essential kind of patterns: as stated above, a malicious adversary can easily find inference channels made up only of elements which are present in the disclosed output. However, these inference channels are not the unique possible source of inference: further inference channels involving also infrequent itemsets could be possibly discovered, albeit in a much more complex way. In fact, in [CG02] deduction rules to derive tight bounds on the support of itemsets are introduced. Given an itemset J, if for each subset $I \subset J$ the support $sup_{\mathcal{D}}(I)$ is known, such rules allow to compute lower and upper bounds on the support of J. Let l be the greatest lower bound we can derive, and u the smallest upper bound we can derive: if we find that $l = u$ then we can infer that $sup_{\mathcal{D}}(J) = l = u$ without actual counting. In this case J is said to be a *derivable itemset*. We transpose such deduction techniques in our context and observe that they can be exploited to discover information about infrequent itemsets (i.e., infer supports in the gray area of Figure 5.1(b)), and from these to discover inference channels crossing the border with the grey area, or even inference channels holding completely within the grey area.

This higher-order problem is discussed later in Section 5.9. However, here we can say that the techniques to detect this kind of inference channels and to block them are very similar to the techniques for the first kind of channels. This is due to the fact that both kinds of channels rely on the same concept: inferring supports of larger itemsets from smaller ones. Indeed, the key equation of our work (Lemma 1) is also the basis of the deduction rules proposed in [CG02]. For the moment, let us restrict our attention to the essential form of inference channel, namely those involving frequent itemsets only.

Problem 1 (Inference Channels Detection). Given a collection of frequent itemsets $\mathcal{F}$ and an anonymity threshold k, our problem consists in detecting all possible inference channels $\mathcal{C} \subseteq \mathcal{F} \; : \; \exists p \in \mathcal{P}at(\mathcal{D}, \sigma) : \mathcal{C} \models 0 < sup_{\mathcal{D}}(p) < k$.

Our mining problem can be seen as a second-order frequent pattern extraction with two frequency thresholds: the usual minimum support threshold σ for itemsets (as defined in Definition 5), and an anonymity threshold k for general patterns (as defined in Definition 2).

Note that an itemset with support less than k is itself a non k-anonymous, and thus dangerous, pattern. However, since we are dealing with σ-frequent itemsets, and since we can reasonably assume that $\sigma \gg k$, such pattern would be discarded by the usual mining algorithms.

We just stated that we can reasonably assume σ to be much larger than k. In fact σ, in real-world applications is usually in the order of hundreds, or thousands, or (more frequently) much larger. Consider that having a small σ on a real-world database, would produce an extremely large number of associations in output, or it would lead to an unfeasible computation. On the other hand, the required level of anonymity k is usually in the order of tens or even smaller. Therefore, it is reasonable to assume $\sigma \gg k$. However, for sake of completeness, if we have $\sigma < k$ then our mining problem will be trivially solved by adopting k as minimum support threshold in the mining of frequent itemsets.

From now on we will avoid discussing this case again, and we will always assume $\sigma > k$.

Definition 11. *The set of all $\mathcal{C}_I^J$ holding in $\mathcal{F}$, together with their supports, is denoted $\mathcal{C}h(k, \mathcal{F}) = \{\langle \mathcal{C}_I^J, f_I^J(\mathcal{D})\rangle \mid 0 < f_I^J(\mathcal{D}) < k \wedge \langle J, sup_{\mathcal{D}}(J)\rangle \in \mathcal{F}\}$.*

Example 6. Consider the database $\mathcal{D}$ in Figure 5.1(a) and suppose $\sigma = 8$ and $k = 3$. The following is the set of inference channels holding in $\mathcal{F}(\mathcal{D}, 8)$:
$\mathcal{C}h(3, \mathcal{F}(\mathcal{D}, 8)) = \{\langle \mathcal{C}_\emptyset^d, 2\rangle, \langle \mathcal{C}_\emptyset^{de}, 1\rangle, \langle \mathcal{C}_e^{de}, 1\rangle, \langle \mathcal{C}_d^{dc}, 1\rangle,$
$\langle \mathcal{C}_\emptyset^{dc}, 2\rangle, \langle \mathcal{C}_{de}^{dce}, 1\rangle, \langle \mathcal{C}_\emptyset^{dce}, 1\rangle, \langle \mathcal{C}_e^{dce}, 1\rangle, \langle \mathcal{C}_\emptyset^{ce}, 1\rangle, \langle \mathcal{C}_e^{ce}, 2\rangle,$
$\langle \mathcal{C}_a^{ae}, 1\rangle, \langle \mathcal{C}_a^{ab}, 1\rangle, \langle \mathcal{C}_\emptyset^{e}, 1\rangle\}$.

Algorithm 6 Naïve Inference Channel Detector

```
Input: $\mathcal{F}, k$
Output: $\mathcal{C}h(k, \mathcal{F})$
$\mathcal{C}h(k, \mathcal{F}) = \emptyset$
for all $\langle J, sup(J)\rangle \in \mathcal{F}$ do
    for all $I \subseteq J$ do
        compute $f_I^J$;
        if $0 < f_I^J < k$ then
            insert$\langle \mathcal{C}_I^J, f_I^J\rangle$ in $\mathcal{C}h(k, \mathcal{F})$;
```

Algorithm 6 detects all possible inference channels $\mathcal{C}h(k, \mathcal{F})$ that hold in a collection of frequent itemsets $\mathcal{F}$ by checking all possible pairs of itemsets $I, J \in \mathcal{F}$ such that $I \subseteq J$. This could result in a very large number of checks. Suppose that $\mathcal{F}$ is formed only by a maximal itemset Y and all its subsets (an itemset is maximal if none of its proper supersets is in $\mathcal{F}$). If $|Y| = n$ we get $|\mathcal{F}| = 2^n$ (we also count the empty set), while the number of possible $\mathcal{C}_I^J$ is $\sum_{1 \le i \le n} \binom{n}{i} (2^i - 1)$. In the following Section we study some interesting properties that allow to dramatically reduce the number of checks needed to retrieve $\mathcal{C}h(k, \mathcal{F})$.

5.6 A Condensed Representation

In this section we introduce a condensed representation of $\mathcal{C}h(k, \mathcal{F})$. A condensed representation of a collection of patterns (in our case of $\mathcal{C}h(k, \mathcal{F})$) is a subset of the collection which is more efficient to compute, and from which we can reconstruct the original collection without accessing the database again. In other words it removes redundancy while preserving all the information.

The benefits of having such condensed representation go far beyond mere efficiency in the detection phase. In fact, by removing the redundancy existing in $\mathcal{C}h(k, \mathcal{F})$, we also implicitly avoid redundant sanitization, when blocking the channels holding in $\mathcal{F}$, to produce a safe output (the issue of how to sanitize the inference channels found in $\mathcal{F}$ is addressed in Section 5.7) .

Consider, for instance, the two inference channels $\langle \mathcal{C}_{\emptyset}^{e}, 1 \rangle$ and $\langle \mathcal{C}_{\emptyset}^{de}, 1 \rangle$ holding in $\mathcal{C}h(3, \mathcal{F}(\mathcal{D}, 8))$ of our running example: one is more specific than the other, but they both uniquely identify transaction t_{12}. It is easy to see that many other families of equivalent, and thus redundant, inference channels can be found. *How can we directly identify one and only one representative inference channel in each family of equivalent ones?* The theory of *closed itemsets* can help us with this problem.

Closed itemsets were first introduced in [PBTL99] and received a great deal of attention especially by an algorithmic point of view [ZH02, PHW03]. They are a concise and lossless representation of all frequent itemsets, i.e., they contain the same information without redundancy. Intuitively, a closed itemset groups together all its subsets that have its same support; or in other words, it groups together itemsets which identify the same group of transactions.

Definition 12 (Closure Operator). *Given the function $f(T) = \{i \in \mathcal{I} \mid \forall t \in T, i(t)\}$, which returns all the items included in the set of transactions T, and the function $g(X) = \{t \in \mathcal{T} \mid X(t)\}$ which returns the set of transactions supporting a given itemset X, the composite function $c = f \circ g$ is the closure operator.*

Definition 13 (Closed Itemset). *An itemset I is closed if and only if $c(I) = I$. Alternatively, a closed itemset can be defined as an itemset whose supersets have a strictly smaller support. Given a database $\mathcal{D}$ and a minimum support threshold σ, the set of frequent closed itemsets is denoted: $\mathcal{C}l(\mathcal{D}, \sigma) = \{\langle X, sup_{\mathcal{D}}(X) \rangle \in \mathcal{F} | \nexists Y \supset X s.t. \langle Y, sup_{\mathcal{D}}(X) \rangle \in \mathcal{F}\}$.*

Example 7. Given the binary database $\mathcal{D}$ in Figure 5.1(a), and a minimum support threshold $\sigma = 8$, we have that:

$$\mathcal{C}l(\mathcal{D}, 8) = \{\langle \emptyset, 12 \rangle, \langle a, 9 \rangle, \langle e, 11 \rangle, \langle ab, 8 \rangle, \langle ae, 8 \rangle, \langle de, 10 \rangle, \langle cde, 9 \rangle\}.$$

Definition 14 (Maximal Frequent Itemset). *An itemset $I \in \mathcal{C}l(\mathcal{D}, \sigma)$ is said to be maximal if and only if $\nexists J \supset I$ s.t. $J \in \mathcal{C}l(\mathcal{D}, \sigma)$.*

Analogously to what happens for the pattern class of itemsets, if we consider the pattern class of conjunctive patterns we can rely on the *anti-monotonicity property of frequency.* For instance, the number of transactions for which the pattern $a \wedge \neg c$ holds is always larger than the number of transactions for which the pattern $a \wedge b \wedge \neg c \wedge \neg d$ holds. This can be straightforwardly transposed to inference channels.

Definition 15. *Given two inference channels $\mathcal{C}_I^J$ and $\mathcal{C}_H^L$ we say that $\mathcal{C}_I^J \preceq \mathcal{C}_H^L$ when $I \subseteq H$ and $(J \setminus I) \subseteq (L \setminus H)$.*

Proposition 1. *$\mathcal{C}_I^J \preceq \mathcal{C}_H^L \Rightarrow \forall \mathcal{D} \,.\, f_I^J(\mathcal{D}) \geq f_H^L(\mathcal{D})$.*

It follows that, when detecting inference channels, whenever we find a two itemsets $H \subseteq L$ such that $f_H^L(\mathcal{D}) \geq k$, we can avoid checking the support of all $\mathcal{C}_I^J \preceq \mathcal{C}_H^L$, since they will not be inference channels.

Definition 16 (Maximal Inference Channel). *An inference channel $\mathcal{C}_I^J$ is said to be* maximal *w.r.t. $\mathcal{D}$, k and σ, if $\forall H, L$ such that $I \subseteq H$ and $(J \setminus I) \subseteq (L \setminus H)$, $f_H^L = 0$. The set of maximal inference channels is denoted $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$.*

Proposition 2.

$$\mathcal{C}_I^J \in \mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma)) \Rightarrow I \in \mathcal{C}l(\mathcal{D}, \sigma) \wedge J \textit{ is maximal.}$$

Proof. i) $I \in \mathcal{C}l(\mathcal{D}, \sigma)$: if I is not closed then consider its closure $c(I)$ and consider $J' = J \cup (c(I) \setminus I)$. For the definition of closure, the set of transactions containing I is the same of the set of transactions containing $c(I)$, and the set of transactions containing J' is the same of the set of transactions containing J. It follows that $\mathcal{C}_{c(I)}^{J'} \succeq \mathcal{C}_I^J$ and $f_{c(I)}^J = f_I^J > 0$. Then, if I is not closed, $\mathcal{C}_I^J$ is not maximal.
ii) J is maximal: if J is not maximal then consider its frequent superset $J' = J \cup \{a\}$ and consider $I' = I \cup a$. It is straightforward to see that $f_I^J = f_I^{J'} + f_{I'}^{J'}$ and that $\mathcal{C}_I^{J'} \succeq \mathcal{C}_I^J$ and $\mathcal{C}_{I'}^{J'} \succeq \mathcal{C}_I^J$. Therefore, since $f_I^J > 0$, at least one among $f_I^{J'}$ and $f_{I'}^{J'}$ must be not null. Then, if J is not maximal, $\mathcal{C}_I^J$ is not maximal as well. □

Example 8. Consider the database $\mathcal{D}$ in Figure 5.1(a) and suppose $\sigma = 8$ and $k = 3$. The following is the set of maximal inference channels: $\mathcal{MC}h(3, \mathcal{C}l(\mathcal{D}, 8)) = \{\langle \mathcal{C}_\emptyset^{cde}, 1\rangle, \langle \mathcal{C}_a^{ab}, 1\rangle, \langle \mathcal{C}_a^{ae}, 1\rangle, \langle \mathcal{C}_e^{cde}, 1\rangle, \langle \mathcal{C}_{de}^{cde}, 1\rangle\}$.

The next Theorem shows how the support of any channel in $\mathcal{C}h(k, \mathcal{F})$ can be reconstructed from $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$.

Theorem 3. *Given $I \subseteq J \in 2^{\mathcal{I}}$, let M be any maximal itemset such that $M \supseteq J$. The following equation holds:*

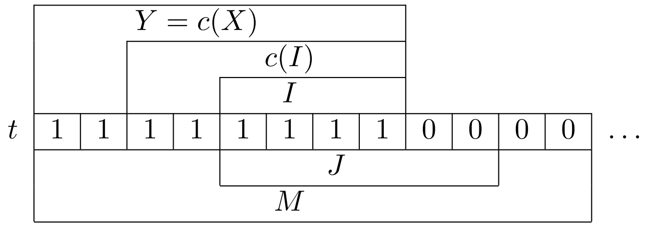
$$f_I^J(\mathcal{D}) = \sum_{c(X)} f_{c(X)}^M(\mathcal{D})$$

where $c(I) \subseteq c(X) \subseteq M$ and $c(X) \cap (J \setminus I) = \emptyset$.

Proof. By definition, f_I^J is equal to the number of transactions t s.t. $\mathcal{C}_I^J(t)$, i.e., all items in I are set to 1 and all items in $J \setminus I$ are set to 0. We prove that, in the summation in the right-hand side of the equation: (i) each of such transactions is counted once, (ii) only once, and (iii) no other transaction is counted.

$i)$ we must show that every such transaction t is considered at least once. This means that exists an itemset X such that $c(I) \subseteq c(X) \subseteq M$ and $c(X) \cap (J \setminus I) = \emptyset$.

Let Y denote the set of items in the M-projection of t that are set to 1. Y is necessarily a frequent closed itemset [2]. Let X be the itemset such that $c(X) = Y$. We have that $c(I) \subseteq c(X) \subseteq M$ and $c(X) \cap (J \setminus I) = \emptyset$.



$ii)$ we must show that in the summation we count each such transaction t exactly once: this means that $\mathcal{C}_{c(X)}^M$ (with M fixed and varying $c(X)$) forms a partition of the set of transactions t such that $\mathcal{C}_I^J(t)$. In fact, given $c(X_1) \subset c(X_2)$, we have that each item in $c(X_2) \setminus c(X_1)$ is set to 0 in the transactions t such that $\mathcal{C}_{c(X_1)}^M(t)$, and set to 1 in the transactions t such that $\mathcal{C}_{c(X_2)}^M(t)$. As a consequence, the same transaction can not be considered by both $\mathcal{C}_{c(X_1)}^M$ and $\mathcal{C}_{c(X_2)}^M$.

$iii)$ For a transaction t in order to be counted, it must exist an itemset X such that $c(X)$ holds in t. Since we require $c(I) \subseteq c(X) \subseteq M$ and $c(X) \cap (J \setminus I) = \emptyset$, no transaction having an item in I set to 0, or an item in $J \setminus I$ set to 1, can be counted. □

Corollary 3. *For all $\langle \mathcal{C}_I^J, f_I^J(\mathcal{D}) \rangle \in \mathcal{C}h(k, \mathcal{F})$ we have that, for any $c(X)$ s.t. $c(I) \subseteq c(X) \subseteq M$ and $c(X) \cap (I \setminus J) = \emptyset$, $0 \leq f_{c(X)}^M(\mathcal{D}) < k$.*

Proof. Since $\mathcal{C}_I^J \preceq \mathcal{C}_{c(X)}^M$, and $f_I^J(\mathcal{D}) < k$, we conclude that $f_{c(X)}^M(\mathcal{D}) \leq f_I^J(\mathcal{D}) < k$. Moreover, for at least one $c(X)$ we have that $f_{c(X)}^M(\mathcal{D}) > 0$, otherwise we get a contradiction to Theorem 3. □

From Corollary 3 we conclude that all the addends needed to compute $f_I^J(\mathcal{D})$ for an inference channel are either in $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$ or are null. Therefore, as the set of all closed frequent itemsets $\mathcal{C}l(\mathcal{D}, \sigma)$ contains all the information of $\mathcal{F}$ in

[2] By contradiction, if Y is not closed, then there is at least one other item a which is always set to 1 in all transactions t such that $Y(t)$. Since Y is the positive part the M-projection of t, it follows that a is not in M, hence $M \cup \{a\}$ is frequent, hence M is not maximal.

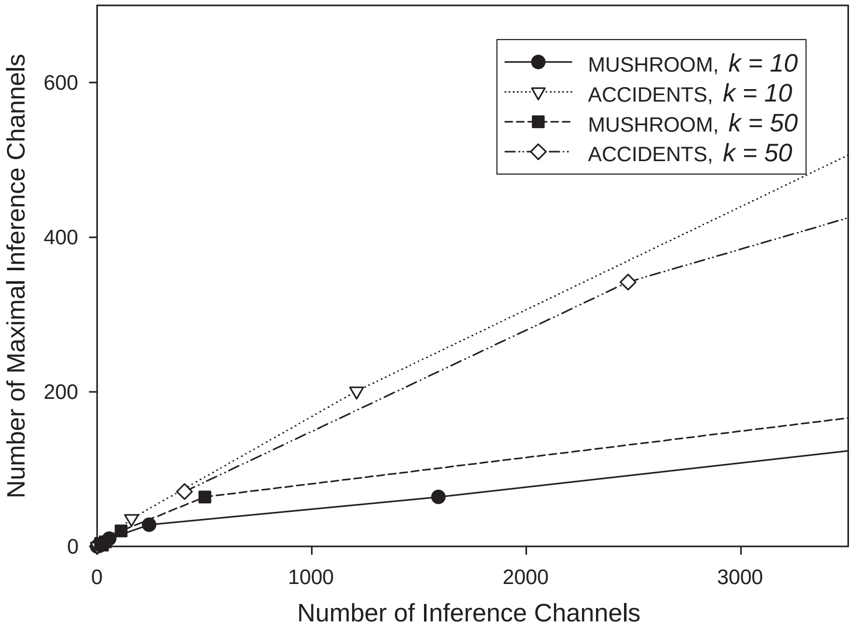


Figure 5.3: Benefits of the condensed representation: size of the representation

a more compact representation, analogously the set $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$ represents, without redundancy, all the information in $\mathcal{C}h(k, \mathcal{F})$.

In the database $\mathcal{D}$ of Figure 5.1(a), given $\sigma = 6$ and $k = 3$, we have that $|\mathcal{C}h(3, \mathcal{F}(\mathcal{D}, 6))| = 58$ while $|\mathcal{MC}h(3, \mathcal{C}l(\mathcal{D}, 6))| = 5$ (Figure 5.1(e)), a reduction of one order of magnitude. On the same database for $\sigma = 6$ and $k = 3$ (our running example, we got $|\mathcal{C}h(3, \mathcal{F}(\mathcal{D}, 6))| = 13$ while $|\mathcal{MC}h(3, \mathcal{C}l(\mathcal{D}, 6))| = 5$.

Such compression of the condensed representation is also confirmed by our experiments on various datasets from the FIMI repository [Goe], reported in Figure 5.3.

Another important benefit of our condensed representation, is that, in order to detect all inference channels holding in $\mathcal{F}$, we can limit ourselves to retrieve only the inference channels in $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$, thus taking in input $\mathcal{C}l(\mathcal{D}, \sigma)$ instead of $\mathcal{F}$ and thus performing a much smaller number of checks. Algorithm 7 exploits the anti-monotonicity of frequency (Proposition 1) and the property of maximal inference channels (Proposition 2) to compute $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$ from $\mathcal{C}l(\mathcal{D}, \sigma)$. Thanks to these two properties, Algorithm 7 dramatically outperform the naive inference channel detector (Algorithm 6), and scales well even for very low support thresholds, as reported in Figure 5.4. Note that the running time of both algorithms is independent from k, while it depends from the minimum support threshold.

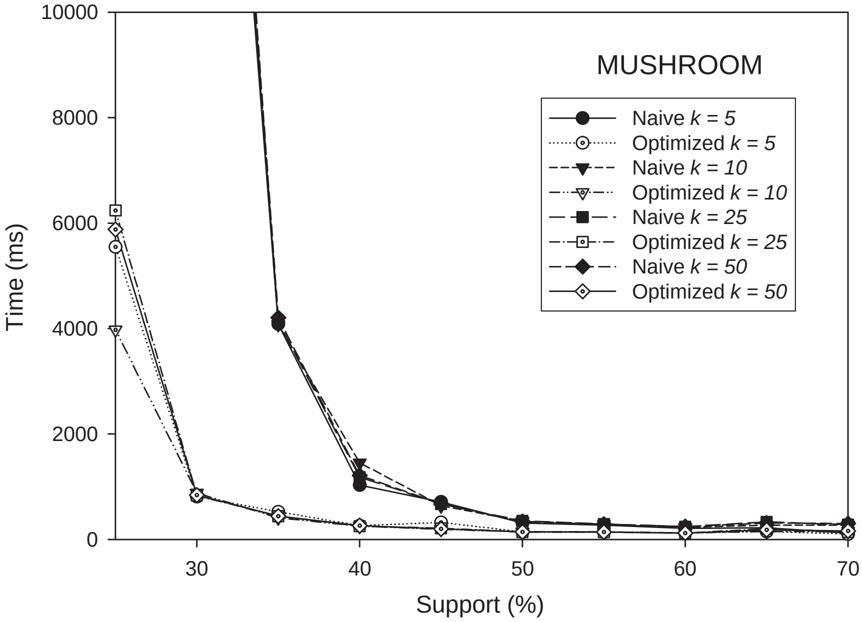


Figure 5.4: Benefits of the condensed representation: running time.

Algorithm 7 Optimized Inference Channel Detector

```
Input: $\mathcal{C}l(\mathcal{D}, \sigma), k$
Output: $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$
$M = \{I \in \mathcal{C}l(\mathcal{D}, \sigma) | I \ is \ maximal\}$;
$\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma)) = \emptyset$;
for all $J \in M$ do
  for all $I \in \mathcal{C}l(\mathcal{D}, \sigma)$ such that $I \subseteq J$ do
    compute $f_I^J$;
    if $0 < f_I^J < k$ then
      insert $\langle \mathcal{C}_I^J, f_I^J \rangle$ in $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$;
```

5.6.1 Anonymity vs. Accuracy: Empirical Observations

Algorithm 7 represents an optimized way to identify all threats to anonymity. Its performance revealed adequate in all our empirical evaluations using various datasets from the FIMI repository [Goe]; in all such cases the time improvement from the Naïve (Algorithm 6) to the optimized algorithm is about one order of magnitude, as reported in Figure 5.3(b).

This level of efficiency allows an interactive-iterative use of the algorithm by the analyst, aimed at finding the best trade-off among privacy and accuracy of the collection of patterns. To be more precise, there is a conflict among keeping the support threshold as low as possible, in order to mine all interesting patterns, and

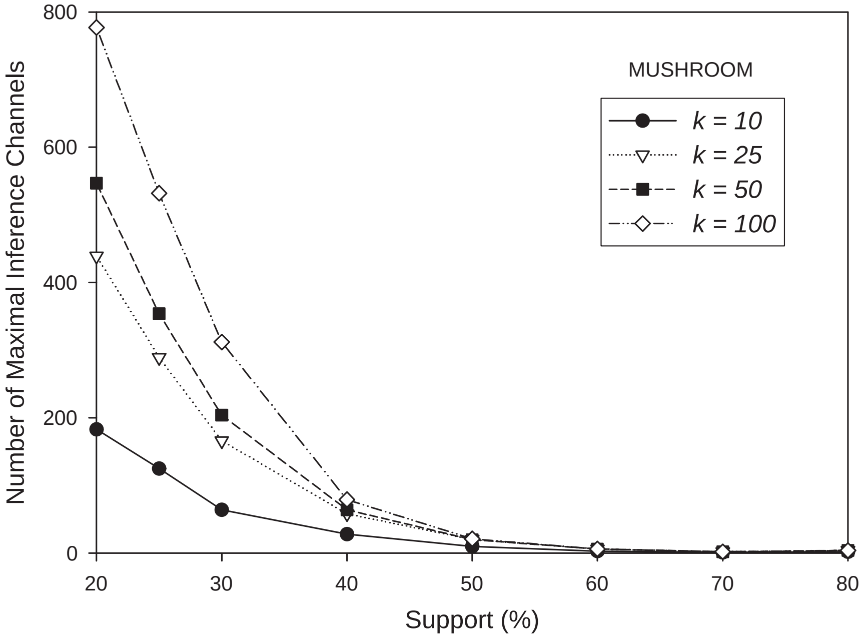


Figure 5.5: Experimental results on cardinality of $\mathcal{MCh}(k, \mathcal{Cl}(\mathcal{D}, \sigma))$ on the Mushroom dataset.

avoiding the generation of anonymity threats.

The best solution to this problem is precisely to find out the minimum support threshold that generates a collection of patterns with no threats, thus avoiding to introduce the distortion needed to block the threats, and thus preserving accuracy.

The plots in Fig. 5.5 and Fig. 5.6 illustrate this point: on the x-axis we report the minimum support threshold, on the y-axis we report the total number of threats (the cardinality of $\mathcal{MCh}(k, \mathcal{Cl}(\mathcal{D}, \sigma))$), and the various curves indicate such number according to different values of the anonymity threshold k. In Figure 5.5 we report the plot for the MUSHROOM dataset (a dense one), while in Figure 5.6 we report the plot for the KOSARAK dataset which is sparse. In both cases, it is evident the value of the minimum support threshold that represents the best trade-off, for any given value of k. However, in certain cases, the best support threshold can still be too high to mine a sufficient quantity of interesting patterns. In such cases, the only option is to allow lower support thresholds and then to block the inference channels in the mining outcome. This problem will be the central topic of the following sections.

5.7 Blocking Inference Channels

In the previous sections we have studied Problem 1: how to detect inference channels in the output of a frequent itemset extraction. Obviously, a solution to this problem

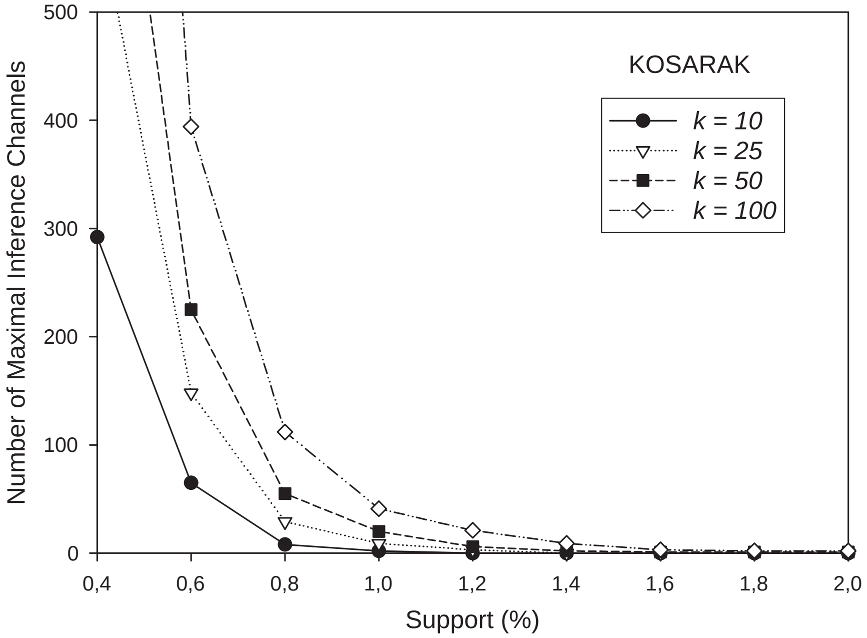


Figure 5.6: Experimental results on cardinality of $\mathcal{MCh}(k, \mathcal{Cl}(\mathcal{D}, \sigma))$ on the Kosarak dataset.

directly yields a method to formally prove that the disclosure of a given collection of frequent itemsets does not violate the anonymity constraint: it is sufficient to check that no inference channel exists for the given collection. In this case, the collection can be safely distributed even to malicious adversaries. On the contrary, if this is not the case, we can proceed in two ways:

- mine a new collection of frequent itemsets under different circumstances, e.g., higher minimum support threshold, to look for an admissible collection;
- transform (sanitize) the collection to remove the inference channels.

When it is needed to pursue the second alternative, we are faced with a second problem, which is addressed in this section.

Problem 2 (Inference Channels Sanitization). Given a collection of frequent itemsets $\mathcal{F}$, and the set of all its inference channels $\mathcal{C}h(k, \mathcal{F})$, transform $\mathcal{F}$ in a collection of frequent itemsets $\mathcal{O}^k$, which can be safely disclosed. $\mathcal{O}^k$ is the output of our problem, and it must satisfy the following conditions:

1. $\mathcal{C}h(k, \mathcal{O}^k) = \emptyset$;
2. $\exists \mathcal{D}' : \mathcal{O}^k = \mathcal{F}(\mathcal{D}', \sigma)$;

3. the effects of the transformation can be controlled by means of appropriate measures.

The first condition imposes that the sanitized output $\mathcal{O}^k$ does not contain any inference channel. Note that in the sanitization process, while introducing distortion to block the "real" inference channels holding in $\mathcal{F}$, transforming it in $\mathcal{O}^k$, we could possibly create some new "fake" inference channels (not existing in the original database and thus not violating the anonymity of real individuals). We do not allow this possibility: although fake, such inference channels could be the starting point for a backward reasoning of a malicious adversary, in other terms, could open the door to inverse mining attacks.

The second condition constraints the output collection of itemsets to be *"realistic"*; i.e., to be compatible with at least a database. Note that, according to Definition 7, we can infer everything from $\mathcal{F}$, if $\mathcal{F}$ itself is not compatible with any database, or, in other words, if it contains contradictions (for instance, $sup\_tdb(X) < sup_{\mathcal{D}}(Y)$ with $X \subset Y$). Disclosing an output which is not compatible with any database could represent a threat. In fact, a malicious adversary could recognize that the set of pattern disclosed is not "real", and he could exploit this leak by reconstructing the missing patterns, starting from those ones present in the output. We call this kind of threat *inverse mining attacks*.

The inverse mining problem, i.e. given a set of σ-frequent itemsets reconstruct a database compatible with it, has been shown NP-complete [Cal04]. However such a problem can be tackled by using some heuristics [WWWL05]. In this section, in order to avoid this kind of attacks, we study how to sanitize a set of patterns in such a way that the output produced is always compatible with at least one database. Doing so, we avoid the adversary to distinguish an output which as been k-anonymized from a non k-anonymized one.

Finally, the third condition of Problem 2 requires to control the distortion effects of the transform of the original output by means of appropriate distortion measures (see Section 5.7).

Note that our output $\mathcal{O}^k$ always contains also the number of individuals in the database, or at least a sanitized version of such number. In fact, since $\mathcal{O}^k$ must be realistic, for the anti-monotonicity of frequency it must always contain the empty itemset with its support, which corresponds to the number of transactions in the database. More formally, we can say that $\langle \emptyset, sup_{\mathcal{D}'}(\emptyset) \rangle \in \mathcal{O}^k$ and $sup_{\mathcal{D}'}(\emptyset) = |\mathcal{D}'|$, where $\mathcal{D}'$ is a database compatible with $\mathcal{O}^k$. The relevance of this fact is twofold. On one hand the size of the database in analysis is a important information to disclose: for instance, in a medical domain, the number of patients on which a novel treatment has been experimented, and to which the set of extracted association rules refers, can not be kept secret. On the other hand, disclosing such number can help a malicious adversary to guess the support of non k-anonymous patterns.

One naïve attempt to solve Problem 2 is simply to eliminate from the output any pair of itemsets I, J such that $\mathcal{C}_I^J$ is an inference channel. Unfortunately, this kind of sanitization would produce an output which is, in general, not compatible with any database. As stated before, we do not admit this kind of solution, because disclosing an inconsistent output could open the door to inverse mining attacks.

In this Section, under the strong constraints imposed above, we develop two dual strategies to solve Problem 2. The first one blocks inference channels by increasing the support of some itemsets in $\mathcal{F}$. Such support increasing is equivalent to adding transaction to the original database $\mathcal{D}$, and thus such strategy is named *additive sanitization*. The second strategy, named *suppressive sanitization*, blocks inference channels by decreasing the support of some itemsets (equivalent to suppress transactions from $\mathcal{D}$).

5.7.1 Avoiding Redundant Distortion

Using our condensed representation of maximal inference channels introduced in Section 5.6, we remove the redundancy existing in $\mathcal{C}h(k, \mathcal{F})$, but this means to implicitly avoid redundant sanitization, and thus we dramatically reduce the distortion needed to block all the inference channels. In fact, to block an inference channel $\mathcal{C}_I^J \in \mathcal{C}h(k, \mathcal{F})$ we have two main options:

- making the inference channel anonymous enough, i.e., forcing $f_I^J(\mathcal{D}) \geq k$;

- making the inference channel disappear, i.e., forcing $f_I^J(\mathcal{D}) = 0$.

The following two propositions show that, whichever option we choose, we can just block the channels in $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$, obtaining to block all the inference channels in $\mathcal{C}h(k, \mathcal{F})$.

Proposition 3. *Given a database $\mathcal{D}$, consider a database $\mathcal{D}'$ s.t. $\forall \langle \mathcal{C}_H^L, f_H^L(\mathcal{D})\rangle \in \mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$ it holds that $f_H^L(\mathcal{D}') \geq k$. Then from Proposition 1 it follows that $\forall \langle \mathcal{C}_I^J, f_I^J(\mathcal{D})\rangle \in \mathcal{C}h(k, \mathcal{F})$, $f_I^J(\mathcal{D}') \geq k$.*

Proposition 4. *Given a database $\mathcal{D}$, consider a database $\mathcal{D}'$ s.t. $\forall \langle \mathcal{C}_H^L, f_H^L(\mathcal{D})\rangle \in \mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$ it holds that $f_H^L(\mathcal{D}') = 0$. Then from Proposition 3 it follows that $\forall \langle \mathcal{C}_I^J, f_I^J(\mathcal{D})\rangle \in \mathcal{C}h(k, \mathcal{F})$, $f_I^J(\mathcal{D}') = 0$.*

In the following we exploit these properties to reduce the distortion needed to sanitize our output.

5.7.2 Additive Sanitization

Given the set of frequent itemsets $\mathcal{F}$, and the set of channels holding in it, the simplest solution to our problem is the following: for each $\mathcal{C}_I^J \in \mathcal{C}h(k, \mathcal{F})$, increase the support of the itemset I by k to force $f_I^J > k$. In order to maintain database-compatibility, the support of all subsets of I is increased accordingly. This is equivalent to add k transactions $t = I$ to the original database $\mathcal{D}$. Clearly, we are not really adding transactions: this is just to highlight that the transformed set of frequent itemsets maintains database-compatibility. Moreover, it will contain exactly the same itemsets, but with some supports increased.

One could observe that it is not strictly necessary to increase the support of I by k, but it is sufficient to increase it by $k - f_I^J$ to block the inference channel (making it reach the anonymity threshold k). This solution has two drawbacks. First, it creates new fake inference channels (which are not allowed by our problem definition). Second, it would produce an output on which a malicious adversary, could compute and find out a lot of $f_I^J = k$. This could suggest to the adversary: (i) that a k-anonymization has taken place, (ii) the exact value of k, and (iii) the set of possible patterns which have been distorted. Increasing the support of I by k we avoid all these problems.

The third requirement of our problem is to minimize the distortion introduced during the anonymization process. Since in our sanitization approach the idea is to increment supports of itemsets and their subsets (by virtually adding transactions in the original dataset), minimizing distortion means reducing as much as possible the increments of supports (i.e., number of transactions virtually added). To do this, we exploit the anti-monotonicity property of patterns, and the condensed representation introduced in Section 5.6. Therefore we will actually feed the sanitization algorithm with $\mathcal{C}l(\mathcal{D}, \sigma)$ and $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$, instead of $\mathcal{F}$ and $\mathcal{C}h(k, \mathcal{F})$. But we can do something more. In fact, when adopting an additive strategy, some redundancy can be found and avoided, even in $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$, as described in the following.

Example 9. Consider the two maximal inference channels $\langle \mathcal{C}_a^{ab}, 1\rangle$ and $\langle \mathcal{C}_a^{ae}, 1\rangle$ in $\mathcal{MC}h(3, \mathcal{C}l(\mathcal{D}, 8))$ (Example 8). According to the additive strategy we should virtually add 3 transactions $'a'$ for the first channel, and other 3 transactions $'a'$ for the second channel. Obviously we can just add 3 transactions to block both channels.

Definition 17 (Maximal Channels Merging). *Two inference channels $\mathcal{C}_I^J$ and $\mathcal{C}_H^L$ can be merged if $I \subseteq H$ and $H \cap (J \setminus I) = \emptyset$; or vice versa, $H \subseteq I$ and $I \cap (L \setminus H) = \emptyset$. Their merge is $\mathcal{C}_I^J \bowtie \mathcal{C}_H^L = \mathcal{C}_{I \cup H}^{J \cup L}$.*

Example 10. Mining the MUSHROOM dataset (8124 transactions) at 60% minimum support level (absolute support $\sigma = 4874$) we obtain 52 frequent itemsets (counting also the empty set). With $k = 10$ the detection algorithm can find 20 inference channels. Among them, only 3 are maximal:

$$\mathcal{C}_{\{85,34\}}^{\{85,86,39,34\}}, \mathcal{C}_{\{85,34\}}^{\{85,59,86,34\}}, \mathcal{C}_{\{85,90,34\}}^{\{85,86,90,36,34\}}$$

In this case all of them can be merged into a unique inference channel:

$$\mathcal{C}_{\{85,34,90\}}^{\{85,59,86,39,34,90,36\}}.$$

Therefore, increasing the support of the itemset $\{85, 34, 90\}$ and of all its subsets by 10, we remove all the 20 inference channels holding in the output of frequent itemset mining on the MUSHROOM dataset at 60% of support.

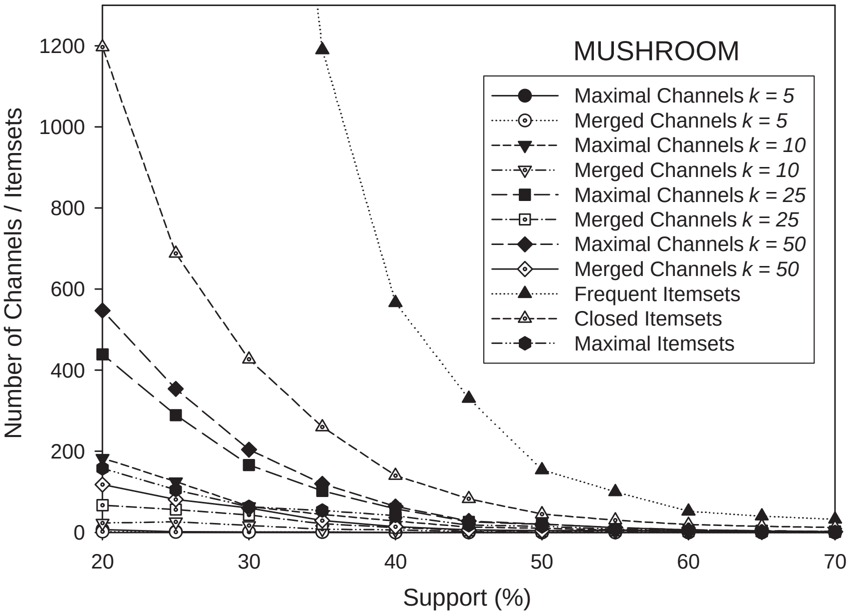


Figure 5.7: The benefits of the channels merging operation.

Algorithm 8 implements the additive sanitization just described: it takes in input $\mathcal{C}l(\mathcal{D}, \sigma)$ and $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$, and returns $\mathcal{O}^k$, which in this case is the sanitized version of $\mathcal{C}l(\mathcal{D}, \sigma)$. Obviously, if we want to output a sanitized version of $\mathcal{F}$ instead of $\mathcal{C}l(\mathcal{D}, \sigma)$, we can simply reconstruct it from the sanitized version of $\mathcal{C}l(\mathcal{D}, \sigma)$ (recall that $\mathcal{F}$ and $\mathcal{C}l(\mathcal{D}, \sigma)$ contain exactly the same information). The algorithm is composed by two phases: during the first phase all maximal channels are merged as much as possible, according to Definition 17; then the resulting set of merged channels is used in the second phase to select the itemsets whose support must be increased.

Example 11. Consider $\mathcal{MC}h(3, \mathcal{C}l(\mathcal{D}, 8))$ in Example 8. The 5 maximal inference channels can be merged giving 3 channels: $\mathcal{C}_a^{abcde}, \mathcal{C}_e^{cde}$ and $\mathcal{C}_{de}^{cde}$. Therefore, according to the additive strategy, we sanitize $\mathcal{C}l(\mathcal{D}, 8)$ (Example 7) virtually adding 3 transactions $'a'$, 3 transactions $'e'$, and 3 transactions $'de'$. The resulting set is $\mathcal{O}^3 = \{\langle\emptyset, 21\rangle, \langle a, 12\rangle, \langle e, 17\rangle, \langle ab, 8\rangle, \langle ae, 8\rangle, \langle de, 13\rangle, \langle cde, 9\rangle\}$.

5.7.3 Suppressive Sanitization

The basic idea of Suppressive Sanitization is to hide inference channels, sending their support to 0: this can be done by removing transactions t s.t. $I \subseteq t \wedge (J \setminus I) \cap t = \emptyset$. Unfortunately, we can not simulate such suppression of transactions simply by decreasing the support of the itemset I by f_I^J for each $\mathcal{C}_I^J \in \mathcal{C}h(k, \mathcal{F})$, since we

Algorithm 8 Additive Sanitization

```
Input: $\mathcal{C}l(\mathcal{D},\sigma), \mathcal{MC}h(k, \mathcal{C}l(\mathcal{D},\sigma))$
Output: $\mathcal{O}^k$
//Merging phase
$S \leftarrow \emptyset$;
for all $\langle \mathcal{C}_I^J, f_I^J \rangle \in \mathcal{MC}h(k, \mathcal{C}l(\mathcal{D},\sigma))$ do
    if $\exists \mathcal{C}_{I'}^{J'} \in S$ s.t. $\mathcal{C}_I^J$ and $\mathcal{C}_{I'}^{J'}$ can be merged then
        $S \leftarrow S \setminus \{\mathcal{C}_{I'}^{J'}\}$;
        $S \leftarrow S \cup (\mathcal{C}_I^J \bowtie \mathcal{C}_{I'}^{J'})$;
    else
        $S \leftarrow S \cup \{\mathcal{C}_I^J\}$;
//Distortion phase
for all $\langle I, sup_{\mathcal{D}}(I) \rangle \in \mathcal{C}l(\mathcal{D},\sigma)$ do
    for all $\mathcal{C}_{I'}^{J'} \in S$ s.t. $I \subseteq I'$ do
        $sup_{\mathcal{D}}(I) \leftarrow sup_{\mathcal{D}}(I) + k$;
$\mathcal{O}^k \leftarrow \mathcal{C}l(\mathcal{D},\sigma)$
```

would lose database-compatibility due to the other items appearing in the dangerous transactions. Consider for instance a transaction $I \cup \{x, y, z\}$: removing it we reduce the support of I, but as uncontrolled side effect, we also reduce the support of the itemset $I \cup \{x\}$. Therefore, in order to maintain database-compatibility, we must take into account these other items carefully. On way of achieving this is to really access the database, suppress the dangerous transactions, and reduce the support of all itemsets contained in the suppressed transactions accordingly. But this is not enough. In fact, while in the additive strategy described before, is sufficient to raise the supports by k to be sure that no novel (fake) inference channel is created, the case is more subtle with the suppressive strategy. The unique solution here is to perform again the detection algorithm on the transformed database, and if necessary, to block the novel inference channels found. Obviously, this process can make some frequent itemsets become infrequent. This is a major drawback of the suppressive strategy w.r.t. the additive strategy which has the nice feature of maintaining the same set of frequent itemsets (even if with larger supports).

Algorithm 9 implements the suppressive sanitization which access the database $\mathcal{D}$ on the basis of the information in $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D},\sigma))$, and adjust $\mathcal{C}l(\mathcal{D},\sigma)$ with the information found in $\mathcal{D}$. As for the additive strategy, the following pseudo-code outputs a sanitized version of $\mathcal{C}l(\mathcal{D},\sigma)$ but nothing prevents us from disclosing a sanitized version of $\mathcal{F}$.

Example 12. Consider $\mathcal{C}l(\mathcal{D},8)$ and $\mathcal{MC}h(3, \mathcal{C}l(\mathcal{D},8))$ in Fig.5.1(d) and (e). Suppressive Sanitization removes transactions which are directly involved in maximal inference channels. In our example we got 5 maximal inference channels $\langle \mathcal{C}_{\emptyset}^{cde}, 1\rangle$, $\langle \mathcal{C}_{a}^{ab}, 1\rangle$, $\langle \mathcal{C}_{a}^{ae}, 1\rangle$, $\langle \mathcal{C}_{e}^{cde}, 1\rangle$ and $\langle \mathcal{C}_{de}^{cde}, 1\rangle$ corresponding to transactions t_{12}, t_8, t_{12}, t_8 and

Algorithm 9 Suppressive Sanitization

```
Input: $\mathcal{C}l(\mathcal{D},\sigma), \mathcal{MC}h(k,\mathcal{C}l(\mathcal{D},\sigma)), \mathcal{D}$
Output: $\mathcal{O}^k$
 1: while $\mathcal{MC}h(k,\mathcal{C}l(\mathcal{D},\sigma)) \neq \emptyset$ do
 2:   //Scan the database
 3:   for all $t \in \mathcal{D}$ do
 4:     if $\exists \langle \mathcal{C}_I^J, f_I^J \rangle \in \mathcal{MC}h(k,\mathcal{C}l(\mathcal{D},\sigma))$ s.t.
        $I \subseteq t$ and $(J \setminus I) \cap t = \emptyset$ then
 5:       //Transaction suppression
 6:       for all $\langle X, sup_{\mathcal{D}}(X) \rangle \in \mathcal{C}l(\mathcal{D},\sigma)$ s.t. $X \subseteq t$ do
 7:         $sup_{\mathcal{D}}(X) \leftarrow sup_{\mathcal{D}}(X) - 1$;
 8:   //Compact $\mathcal{C}l(\mathcal{D},\sigma)$
 9:   for all $\langle X, s \rangle \in \mathcal{C}l(\mathcal{D},\sigma)$ do
10:     if $\exists \langle Y, s \rangle \in \mathcal{C}l(\mathcal{D},\sigma)$ s.t. $Y \supset X$ or $s < \sigma$ then
11:       $\mathcal{C}l(\mathcal{D},\sigma) \leftarrow \mathcal{C}l(\mathcal{D},\sigma) \setminus \langle X, s \rangle$;
12:   detect $\mathcal{MC}h(k,\mathcal{C}l(\mathcal{D},\sigma))$ in $\mathcal{C}l(\mathcal{D},\sigma)$;
13: $\mathcal{O}^k \leftarrow \mathcal{C}l(\mathcal{D},\sigma)$;
```

t_7 respectively. The suppression of these 3 transactions reduces the support of some closed itemsets. In particular, at the end of the suppression phase (line 8 of Algorithm 9) we got that $\mathcal{C}l(\mathcal{D}, 8) = \{\langle\emptyset, 9\rangle, \langle a, 6\rangle, \langle e, 9\rangle, \langle ab, 6\rangle, \langle ae, 6\rangle, \langle de, 9\rangle, \langle cde, 9\rangle\}$. Compacting $\mathcal{C}l(\mathcal{D}, 8)$ means to remove from it itemsets which, due to the transactions suppression, are no longer frequent or no longer closed (lines 9 − 12), i.e., $\mathcal{C}l(\mathcal{D}, 8) = \{\langle cde, 9\rangle\}$. At this point Algorithm 9 invokes the optimized detection algorithm (Algorithm 7) to find out the maximal channels in the new $\mathcal{C}l(\mathcal{D}, 8)$, and if necessary, starts a new suppression phase. In our example this is not the case, since we have no more inference channels. Therefore the resulting output, which can be safely disclosed, is given by the itemset *'cde'* and all its subsets, all having the same support 9.

5.8 Experimental Analysis

In this section we report the results of the experimental analysis that we have conducted in order to:

- assess the distortion introduced by our sanitization strategies;
- measure time needed by our sanitization framework (inference channels detection plus blocking);
- compare empirically the differences between k-anonymizing the data and k-anonymizing the patterns.

5.8.1 Distortion Empirical Evaluation

In the following we report the distortion empirical evaluation we have conducted on three different datasets from the FIMI repository [Goe], with various σ and k thresholds, recording the following four measures of distortion:

1. Absolute number of transaction virtually added (by the additive strategy) or suppressed (by the suppressive strategy).

2. The fraction of itemsets in $\mathcal{F}$ which have their support changed in $\mathcal{O}^k$:

$$\frac{|\{\langle I, sup_{\mathcal{D}}(I)\rangle \in \mathcal{F} \; : \; sup_{\mathcal{O}^k}(I) \neq sup_{\mathcal{D}}(I)\}|}{|\mathcal{F}|}$$

   where $sup_{\mathcal{O}^k}(I) = s$ if $\langle I, s\rangle \in \mathcal{O}^k$; 0 otherwise.

3. The average distortion w.r.t. the original support of itemsets:

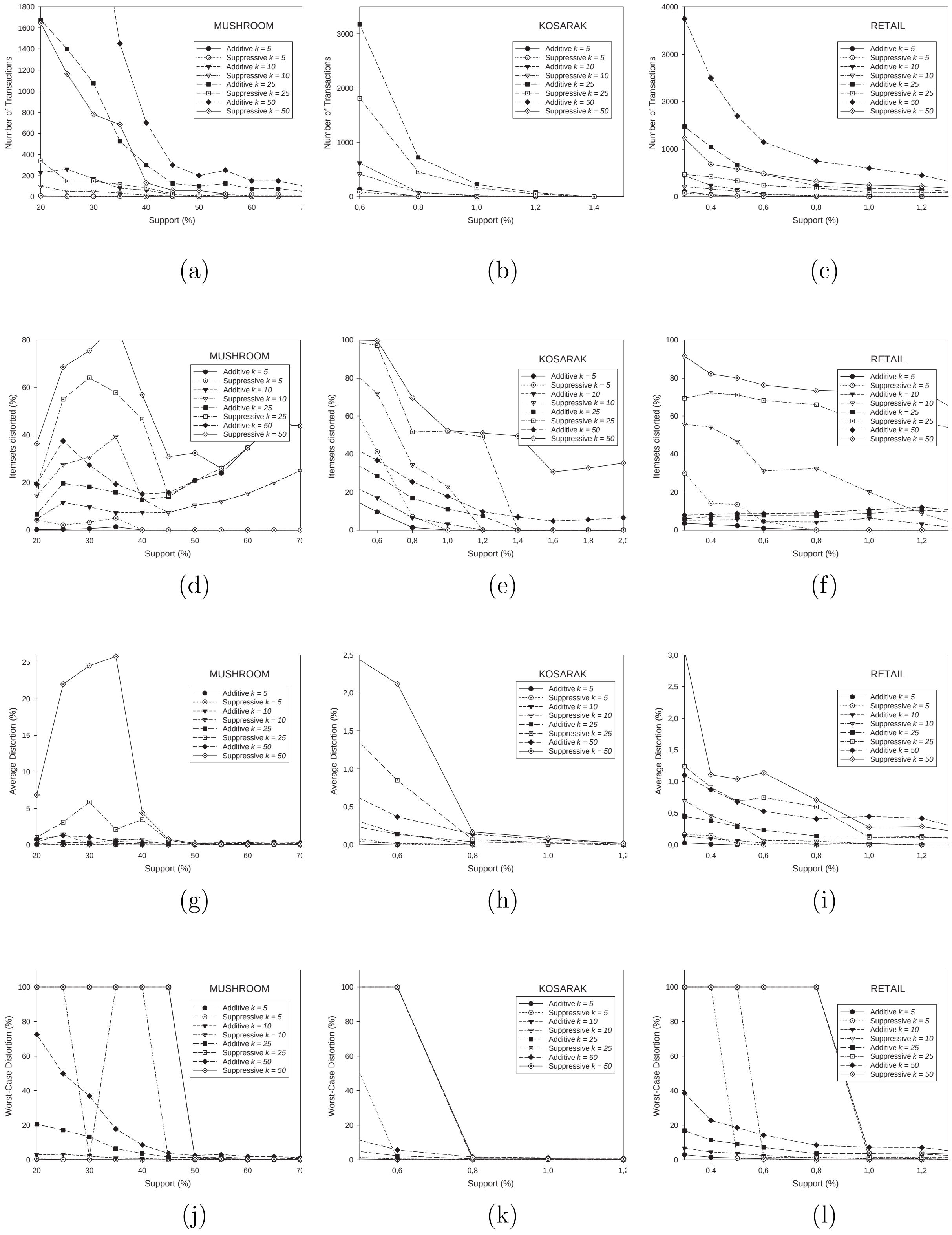
$$\frac{1}{|\mathcal{F}|} \sum_{I \in \mathcal{F}} \frac{|sup_{\mathcal{O}^k}(I) - sup_{\mathcal{D}}(I)|}{sup_{\mathcal{D}}(I)}$$

4. The worst-case distortion w.r.t. the original support of itemsets:

$$\max_{I \in \mathcal{F}} \left\{ \frac{|\, sup_{\mathcal{O}^k}(I) - sup_{\mathcal{D}}(I)\, |}{sup_{\mathcal{D}}(I)} \right\}$$

The first row of plots in Figure 5.8 reports the first measure of distortion. Since the size of $\mathcal{MC}h(k, \mathcal{C}l(\mathcal{D}, \sigma))$ grows for larger k and for smaller σ, the number of transactions added (by the additive strategy) or suppressed (by the suppressive strategy) behaves accordingly. However, in few cases, the additive sanitization exhibits a non-monotonic behavior (e.g., the local maximum for $\sigma = 55\%$ and $k = 50$ in Figure 5.8(a)): this is due to the non-monotonicity of the merge operation (Definition 17 and lines 4–6 of Algorithm 1). In general, for reasonable values of σ, the number of transactions involved in the sanitization is always acceptable (the total number of transactions in MUSHROOM, KOSARAK and RETAIL are respectively 8124, 990002 and 88162).

The percentage of itemsets whose support has been distorted is reported in the second row of plots of Figure 5.8. Additive sanitization always behaves better than suppressive: in fact, for each inference channel $\mathcal{C}_I^J$, additive only changes the support of I and its subsets, while suppressive changes the support of other itemsets contained in the dangerous transaction. Non-monotonicity (evident in Figure 5.8(d), but also present in Figure 5.8(e) and (f)) is due to the fact that lowering the value of σ some infrequent itemsets became frequent but they do not contribute to new inference channels.



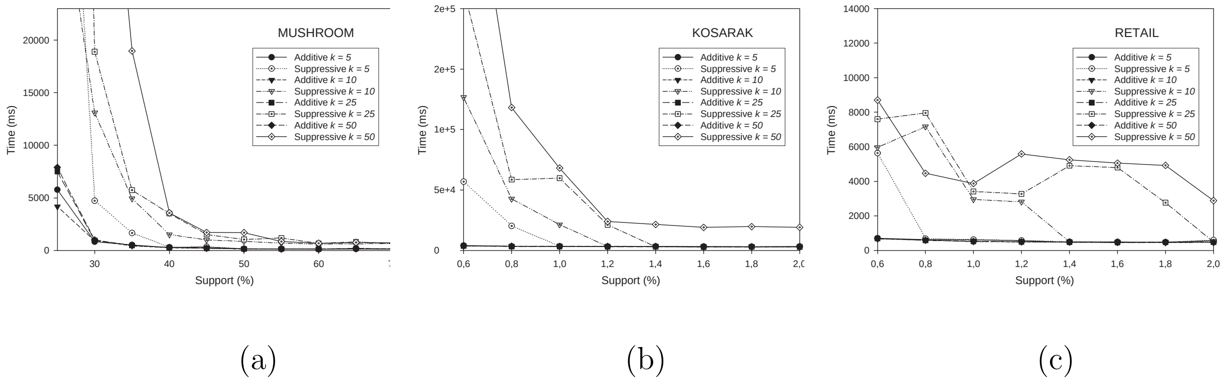
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 5.8: Distortion empirical evaluation.



(a) (b) (c)

Figure 5.9: Experiments on run-time (both detection and sanitization).

The third row of plots of Figure 5.8 reports the average distortion introduced. Notice that this measure drastically penalizes the suppression approach that possibly makes some itemsets become infrequent: for this itemsets we count a maximum distortion of 1 ($sup_{\mathcal{O}^k}(I) = 0$).

It is worth noting that, while the number of itemsets distorted is usually very large, the average distortion on itemsets is very low: this means that quite all itemsets are touched by the sanitization, but their supports are changed just a little. Consider, for instance, the experiment on KOSARAK (Figure 5.8(e) and (h)) with $k = 50$ and $\sigma = 0.6\%$: the additive strategy changes the support of the 40% of itemsets in the output, but the average distortion is 0.4%; the suppressive sanitization changes the 100% of itemsets but the average distortion is 2.1%.

Finally, the fourth row of plots of Figure 5.8 reports the worst-case distortion, i.e., the maximum distortion of the support of an itemset in $\mathcal{F}$. Note that this measure is 100% whenever the suppressive strategy makes at least one frequent itemset infrequent. While for the additive strategy the itemset maximally distorted is always the empty itemset, i.e. the bottom of the itemset lattice, which, as discussed in Section 5.7, is always present in our output $\mathcal{O}^k$, and which represents the number of transactions in the database. Since the empty itemset is subset of any set, every single transaction virtually added by the additive strategy increases its support.

Also w.r.t. this measure the additive strategy outperforms the suppressive strategy, providing a reasonable distortion. However, note that when the suppressive strategy does not make any frequent itemset become infrequent (for instance in Figure 5.8(j) for $k = 5$ and a support of 25%) the worst-case distortion is less than the distortion introduced by the additive strategy.

A major drawback of the additive approach is that the transactions virtually added are fake. The suppressive approach, on the other hand, induces a stronger distortion, but such distortion is only due to the hiding of some nuggets of information containing threats to anonymity. Therefore, in many practical situations, the suppressive strategy could be a more reasonable choice, guaranteeing a more meaningful output.

5.8.2 Run-time Analysis

Although time performance is not a central issue in our contribution, it is worth noting that the execution of both strategies was always very fast (ranging from less than 1 second to few tens of seconds). Figure 5.9 reports the time needed to detect and sanitize inference channels, i.e., Algorithm 7 followed by Algorithm 8 (Additive) or 9 (Suppressive). Note that (quite obviously) run-time generally grows for growing k and shrinking σ. However, in Figure 5.9(c) the suppressive strategy has an evident non monotonic behavior w.r.t. σ: this is due to the possibly different number of sanitization cycles required by the suppressive strategy for different σ.

5.8.3 Anonymizing Data Vs. Anonymizing Patterns

In the following we resume the thread of the discourse we began in Section 5.3. As stated before, a trivial solution to our pattern sanitization problem could be to first k-anonymize the source database $\mathcal{D}$ using some well known technique, for instance the DATAFLY algorithm introduced in [Swe02b], obtaining a k-anonymized database $\mathcal{D}'$, and then to mine the frequent itemsets from $\mathcal{D}'$, using for instance APRIORI. In fact, by mining a k-anonymized database, the set of frequent itemsets that we obtain is clearly safe, in the sense that it does not contain inference channels. The situation is described by Figure 5.10. We claim that, if the goal is to disclose the result of the mining and not the source data, such solution is overkilling.

Example 13. Consider again our running example: the database $\mathcal{D}$ in Figure 5.1(a) with $\sigma = 8$ and $k = 3$. Suppose that all items in $\mathcal{I}$ are quasi-identifier. If we try to k-anonymize the data, the sanitization process would involve also the tuples not covering any patterns. In particular, also the three singleton items which are infrequent (i.e., f, g and h) would be involved in the sanitization. Consider for instance the tuples t_9, t_{10} and t_{11}: if we consider their projection on the frequent items (i.e., a, b, c, d and e) these tuples are identical. In other words, the pattern $p = \neg a \wedge \neg b \wedge c \wedge d \wedge e$, describing the three transaction above, is k-anonymous having $sup_{\mathcal{D}}(p) = 3 = k$. Therefore, if we sanitize the patterns such tuples would not be involved. On the contrary, if we first try to sanitize the data, we must somehow sanitize these three tuples due to differences in f, g and h.

It is natural to argue that the traditional k-anonymization on data is developed for multivalued attributes (not for binary), and it is particularly effective in the cases in which there are only few quasi-identifiers among the whole set of attributes. Following this consideration, we further analyze the issue (graphically described in Figure 5.10) by means of an experimentation on the ADULT dataset from the UCI repository.

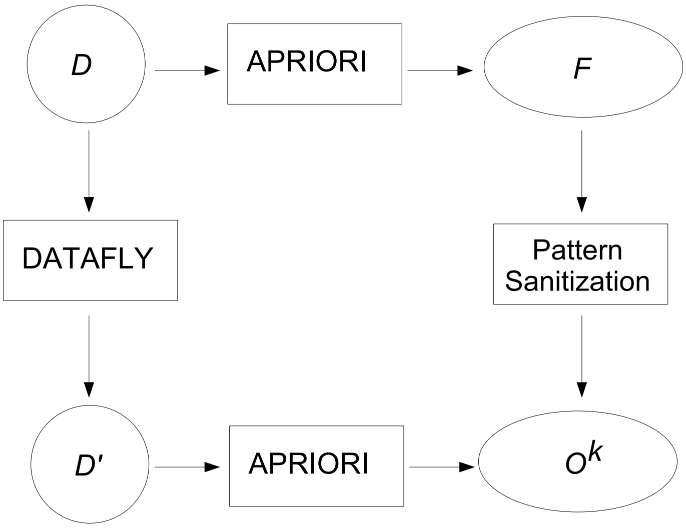


Figure 5.10: Two alternatives paths from a source database $\mathcal{D}$ to an output set of itemsets, $\mathcal{O}^k$, which can be safely disclosed. One path first k-anonymize the data, using for instance DATAFLY and obtaining a k-anonymized database $\mathcal{D}'$ from which $\mathcal{O}^k$ is mined by means of APRIORI. The second path first mine the set of frequent itemsets $\mathcal{F}$ from the original database $\mathcal{D}$, and then k-anonymize the result of the mining using, for instance our Algorithm 8 or 9.

As usually done when preprocessing data for association rule or frequent itemset mining, we have discretized the continuous-valued attributes. It is worth noting that a multivalued attribute can be transformed into a set of binary ones: one binary attribute for each possible value of the multivalued attribute. This simple transformation allows to perform frequent itemsets (and thus association rule) mining on real-world categorical data. Applying it to the ADULT dataset we obtained 341 items (or attribute-value pairs). Moreover, in the preprocessing phase, we have removed tuples containing missing values, obtaining a dataset with 30162 tuples. Let us denote this preprocessed dataset $\mathcal{D}$. In the following we describe the results of our experiments on $\mathcal{D}$ with a frequency threshold $\sigma = 18100$ (i.e., $\approx 60\%$ of the total number of tuples in $\mathcal{D}$), and an anonymity threshold $k = 50$.

As a first step we have mined frequent itemsets from $\mathcal{D}$: the result is useful for comparing the distortions introduced by the two different philosophies. We obtained 41 frequent itemsets, which are also closed itemsets ($\mathcal{F}(\mathcal{D}, 18100) = \mathcal{C}l(\mathcal{D}, 18100)$). Among these 12 are maximal, and they are reported in Table 5.2.

We have then applied the Optimized Inference Channel Detector (Algorithm 7) to $\mathcal{C}l(\mathcal{D}, 18100)$ obtaining four maximal inference channels, which are reported in Table 5.3. For instance consider the first inference channel: it indicates that in our dataset there are only 36 individuals such that: Native-Country $\neq$ United-States, Capital-Loss $\neq 0$, Workclass $\neq$ Private. The fact that our detector has found some inference channels, means that the output $\mathcal{F}(\mathcal{D}, 18100)$ can not be safely disclosed

Itemset	**Support**	**ADD**	**SUP**
{ Native-Country = United-States, Capital-Loss = 0, WorkClass = Private}	19237	19237	19237
{ Capital-Loss = 0, Sex = Male}	19290	19290	19290
{ Race = White, Capital-Gain = 0, Income = Low}	18275	18275	18249
{ Race = White, Capital-Loss = 0, Income = Low}	18489	18489	18489
{ Sex = Male, Capital-Gain = 0}	18403	18403	18263
{ Race = White, Capital-Loss = 0, WorkClass = Private}	18273	18273	18273
{ Native-Country = United-States, Sex = Male}	18572	18572	18512
{ Race = White, Native-Country = United-States, Capital-Loss = 0, Capital-Gain = 0}	20836	20836	20836
{ Native-Country = United-States, WorkClass = Private, Capital-Gain = 0}	18558	18558	18493
{ Hours-per-Week = 2}	18577	18577	18446
{ Capital-Loss = 0, WorkClass = Private, Capital-Gain = 0}	19623	19623	19573
{ Native-Country = United-States, Capital-Loss = 0, Capital-Gain = 0, Income = Low}	19009	19009	19009
Number of tuples in the database	**30162**	**30212**	**29967**

Table 5.2: The set of 12 maximal frequent itemset extracted from the database $\mathcal{D}$ with $\sigma = 18100$, and their supports. The third and the fourth columns report their supports in the output $\mathcal{O}^k$ produced by our additive strategy (ADD), and suppressive strategy (SUP). We also reports the number of tuples in the database, corresponding to the support of the empty set $sup_{\mathcal{D}}(\{\})$.

I	J	f_I^J
{ }	{ Native-Country = United-States, Capital-Loss = 0, Workclass = Private}	36
{ }	{ Race = White, Capital-Loss = 0, WorkClass = Private}	49
{ Capital-Gain = 0 }	{ Race = White, Native-Country = United-States, Capital-Loss = 0, Capital-Gain = 0}	48
{ Capital-Gain = 0 }	{ Native-Country = United-States, Capital-Loss = 0, Capital-Gain = 0, Income = Low}	48

Table 5.3: The set of maximal inference channels $\mathcal{MC}h(50, \mathcal{C}l(\mathcal{D}, 18100))$.

if we want an anonymity level larger than $k = 50$.

We then proceed with our pattern sanitization. If we apply the Additive strategy (Algorithm 8) we must try to merge (see Definition 17) the four maximal inference channels found by the channels detector. The merging operation returns a unique inference channel $\mathcal{C}_I^J$, where I ={ Capital-Gain = 0 } and J = { Race = White, Native-Country = United-States, WorkClass = Private, Capital-Gain = 0, Capital-Loss = 0, Incomes = Low}.

Therefore, according to the additive sanitization, we increase the support of the itemset {Capital-Gain = 0} by k, which in this case is 50. We must increase the support of all its subset of the same amount: being a singleton, this corresponds to increase only the support of the empty set. As stated in Section 5.7 the support of the empty set corresponds to number of tuples in the database, and since it is a frequent itemset, it is always contained in the disclosed output.

The Suppressive strategy, instead, removed 195 transactions. It is worth noting that, in this case, the suppressive strategy has not made any frequent itemset become infrequent. In conclusion, both the additive and the suppressive strategy maintain the original set of frequent itemsets and they only slightly change the support of a few itemsets. This is not the case with the other philosophy (first k-anonymize the

Itemset	**Support**
Age = Any, Native-Country = Any, Race = Any, Hours-per-Week = 2	18577
Age = Any, Native-Country = Any, Race = Any, Sex = Male, Capital-Gain = 0	18403
Age = Any, Native-Country = Any, Race = Any, Sex = Male, Capital-Loss = 0	19290
Age = Any, Native-Country = Any, Race = Any, WorkClass = Private, Capital-Gain = 0, Capital-Loss = 0	19623
Age = Any, Native-Country = Any, Race = Any, Income = Low, Capital-Gain = 0, Capital-Loss = 0	21021

Table 5.4: The set of maximal frequent itemsets obtained by mining $\mathcal{D}'$ (the database k-anonymized by DATAFLY).

data and then mine the frequent itemsets), as described in the following.

We implemented the well known DATAFLY algorithm [Swe02b] which k-anonymizes a database by generalizing attribute which are quasi-identifier, and by removing some tuples when needed. It takes in input the database $\mathcal{D}$, the set of quasi identifiers $QI \subseteq \mathcal{I}$, and for each attribute in QI a *domain generalization hierarchy.*

In the pattern sanitization framework we consider every attribute as a possible source of threat. The same approach is inapplicable in DATAFLY (i.e., considering every attribute as a quasi-identifier): in our case, we lost all the information contained in the database, since all attributes were maximally generalized. We therefore drastically reduced the number of quasi-identifier attributes, from 15 to 5 (Age, Race, Sex, Native-Country and Income). We had that only 3170 tuples out of 30162 were non k-anonymous, leading to attributes generalization (no tuple was suppressed by DATAFLY[3]). After attribute generalization, the 341 items (or attribute-value pairs) were reduced to 292. Let us denote the resulting, anonymized database, $\mathcal{D}'$. We then mined, by means of the APRIORI algorithm, the anonymized database $\mathcal{D}'$ obtaining 128 frequent itemsets, 16 closed itemsets and 5 maximal (listed in Table 5.4).

In conclusion, by comparing Table 5.2 and Table 5.4, we can readily observe the huge loss of information obtained by mining a k-anonymized database. The four itemsets of Table 5.4 are extremely generic, if compared with the 12 patterns of Table 5.2, due to unnecessary generalization in the source data. If the goal is not to disclose data, but the result of mining, sanitizing the mined patterns yields better quality results than mining anonymized source data: the pattern sanitization process focuses only on the portions of data pertinent to the harmful patterns, instead of the whole source dataset. We might argue that in our context, contrarily to the medical wisdom, cure is better than prevention.

5.9 Conclusion and Future Work

We introduced the notion of k-anonymous patterns. Such notion serves as a basis for a formal account of the intuition that a collection of patterns, obtained by data

[3] Note that this choice is taken by the algorithm.

mining techniques and made available to the public, should not offer any possibilities to violate the privacy of the individuals whose data are stored in the source database. To the above aim, we formalized the threats to anonymity by means of inference channel through frequent itemsets, and provided practical algorithms to (i) check whether or not a collection of mined patterns exhibits threats, and (ii) eliminate such threats, if existing, by means of a controlled distortion of the pattern collection. The overall framework provides comprehensive means to reason about the desired trade off between anonymity and quality of the collection of patterns, as well as the distortion level needed to block the threatening inference channels. Concerning the blocking strategies, it is natural to confront our method with the traditional sanitization approach where the source dataset is transformed in such a way that the forbidden patterns are not extracted any longer. We, on the contrary, prefer to transform the patterns themselves, rather than the source data. In our opinion, this is preferable for two orders of reasons. First, in certain cases the input data cannot be accessed more than once: a situation that occurs increasingly often as data streams become a typical source for data mining. In this case there is no room for repeated data pre-processing, but only for pattern post-processing. Second, as a general fact the distortion of the mined patterns yields better quality results than repeating the mining task after the distortion of the source data, as thoroughly discussed in Section 5.8.3.

The first objective of our on-going investigation is the characterization of *border crossing inference channels*; i.e., inference channels also made of itemsets which are not frequent. Recall that in Section 5.5 we defined a subset of all possible patterns, namely σ-vulnerable patterns (Definition 10), and proved that our detection and sanitization methodology produces an output which is k-anonymous w.r.t. σ-vulnerable patterns.

Example 14. Consider again our running example of Figure 5.1 with $\sigma = 8$ and $k = 3$. As shown in Example 11, the set of closed frequent itemsets resulting from the additive sanitization is $\mathcal{O}^3 = \{\langle\emptyset, 21\rangle, \langle a, 12\rangle, \langle e, 17\rangle, \langle ab, 8\rangle, \langle ae, 8\rangle, \langle de, 13\rangle, \langle cde, 9\rangle\}$. From Theorem 2 we know that all σ-vulnerable patterns such as $(a \wedge \neg b) \vee (c \wedge \neg e)$ are k-anonymous, i.e. there exists at least a database, compatible with $\mathcal{O}^3$, in which such pattern does not appear or it appears at least k times. Therefore, a malicious attacker can not infer, in any possible way, that the cardinality of the group of individuals described by such pattern is for sure lower than k.

Unfortunately, Theorem 2 applies to σ-vulnerable pattern such as $(a \wedge \neg b) \vee (c \wedge \neg e)$, i.e., both itemsets $\{ab\}$ and $\{ce\}$ are frequent; but it does not apply, for instance, to the pattern $a \wedge b \wedge \neg c$, since the itemset $\{abc\}$ is not frequent.

In the particular case of our running example, all non σ-vulnerable patterns are also k-anonymous, in other terms no *border crossing inference channels* can be found. This means that $\mathcal{O}^3$ is completely safe. To prove this, we show that exists a 3-anonymous database compatible with $\mathcal{O}^3$ (w.r.t. $\sigma = 8$).

a	b	c	d	e	
1	1	1	1	1	x4
0	0	1	1	1	x5
0	0	0	1	1	x4
1	0	0	0	1	x4
1	1	0	0	0	x4

Similarly, we can show that also the $\mathcal{O}^3$ produced by the suppressive strategy is completely safe.

Unfortunately this ad-hoc proofs can not be applied in general. We are provable protected under attacks to σ-vulnerable patterns, but, although hardly, it is possible to find set of sanitized itemsets that allow the attacker to discover non σ-vulnerable patterns that are non k-anonymous.

Example 15. Suppose that for a given database, and with thresholds $\sigma = 5$ and $k = 3$, after our pattern sanitization we got the following set of itemsets: $\mathcal{O}^3 = \{\langle ab, 9\rangle, \langle ac, 8\rangle, \langle bc, 5\rangle, \langle a, 14\rangle, \langle b, 14\rangle, \langle c, 13\rangle, \langle \emptyset, 22\rangle\}$.

By applying deduction rules [CG02], we can infer some information on the support of the infrequent itemsets $\{abc\}$. In fact we got that:
$sup(abc) \geq sup(ab) + sup(ac) - sup(a) = 3$, and
$sup(abc) \leq sup(\emptyset) - sup(a) - sup(b) - sup(c) + sup(ab) + sup(ac) + sup(bc) = 22 - 14 - 14 - 13 + 9 + 8 + 5 = 3$. Therefore, although it has not been disclosed, we can conclude that the support of the infrequent itemset $\{abc\}$ is exactly 3. From this we can discover a *border crossing inference channels*: $\mathcal{C}_{bc}^{abc}$ which as support $2 < k$. We have inferred that the non σ-vulnerable patterns $b \wedge c \wedge \neg a$ is also non k-anonymous.

This kind of attacks are very difficult but, in some cases, possible. We are actually characterizing them, and studying ad hoc detection and sanitization techniques.

Other issues, emerging from our approach, are worth a deeper investigation and are left to future research.

One path of research regards the mapping of our theoretical framework to the more concrete case of categorical data originating from relational tables. In this context, we could exploit the semantics of the attributes in order to apply generalization techniques similar to what done by classical k-anonymization [Swe02b]. Moreover, we could introduce in our framework the concept of quasi-identifiers and focus our pattern sanitization techniques only to the projection of patterns on the quasi-identifiers. After this concretization step, we should be able to provide a more comprehensive empirical evaluation of our approach: to this purpose we intend to conduct a large-scale experiment with real life bio-medical data about patients to assess both applicability and scalability of the approach in a realistic, challenging domain.

Finally, we plan to investigate whether the proposed notion of anonymity preserving pattern discovery may be applied to other forms of patterns and models, for instance, classification or clustering models. In those scenarios, we should study how to produce a model that, while maintaining accuracy, it provably does not allow an attacker to infer information regarding less than k individuals in the original training set.

In any case, the importance of the advocated form of privacy-preserving pattern discovery is evident: demonstrably trustworthy data mining techniques may open up tremendous opportunities for new knowledge-based applications of public utility and large societal and economic impact.

Conclusions

The main purpose of this thesis is two-fold:

- to show how abductive reasoning can be profitably used in order to enhance data mining results;
- to study and define anonymity breaches in the data mining results;

Our results showed that Abductive frameworks can be seen as a layer able to accept user knowledge and requests in an high level language and then to compute the solution. We started from the idea that framing the data mining step into a logic framework provides us with the possibility of exploiting also domain knowledge in the knowledge extraction and exploitation process, then we showed that the result of a classification tree can be improved if the tree is not simply traversed, but it is visited in abductive mode.

Indeed, the basic observation is that the application of a classification tree to a new observation can be viewed as a straightforward abductive computation, as soon as the tree is represented as a collection of rules in the obvious way. This observation opens up a world of possibilities, since the abductive computations can become very sophisticated, and they can take into account several types of knowledge. First, we considered domain knowledge represented as simple constraints involving equalities, conjunction and negation: even with this limited power, we could show examples in which the classification task was radically improved. Then we extended the system to include probabilities as costs and rules mined from the input dataset. The result is a framework able to give an interpretation of the data mining results with almost no loss of prediction accuracy. We are confident that more sophisticated use of knowledge can lead us to much better improvements.

With respect to our second purpose, we developed a theory on anonymity-preserving pattern discovery based on k-anonymity, showing that data mining results can violate the anonymity of individuals.

We developed efficient algorithms to find and also remove such anonymity breaches, obtaining patterns that are provably anonymous. The threats to anonymity are formalized by means of inference channel through frequent itemsets, and provided practical algorithms to:

- check whether or not a collection of mined patterns exhibits threats;

- eliminate such threats, if existing, by means of a controlled distortion of the pattern collection.

The overall framework provides comprehensive means to reason about the desired trade off between anonymity and quality of the collection of patterns, as well as the distortion level needed to block the threatening inference channels.

We believe that this kind of privacy preserving data mining can open up several new applications, from medical to spatio-temporal data mining.

Bibliography

[AA01] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Symposium on Principles of Database Systems*, 2001.

[Abd04] Ashraf M. Abdelbar. Approximating cost-based abduction is np-hard. *Artificial Intelligence*, 159(1-2):231–239, 2004.

[ABE+99] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules, 1999.

[ABGP05a] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Anonymity and data mining. *Computer Systems: Science & Engineering*, 20(5):369–376, 2005.

[ABGP05b] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *5th IEEE International Conference on Data Mining (Houston, Texas, USA)*. IEEE, 2005.

[ABGP05c] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. k-anonymous patterns. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases (Porto, Portugal, 3-7 October, 2005)*, volume 3721 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[ABGP06] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Towards low-perturbation anonymity preserving pattern discovery. In *SAC 2006*. ACM, 2006. to appear.

[AEI+99] M. Atallah, A. Elmagarmid, M. Ibrahim, E. Bertino, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*, page 45. IEEE Computer Society, 1999.

[Agg05] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, pages 901–909, 2005.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, pages 207–216, 1993.

[AKSX02] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Hippocratic databases. In *28th Int'l Conf. on Very Large Databases (VLDB), Hong Kong*, 2002.

[AMT03] Maurizio Atzori, Paolo Mancarella, and Franco Turini. Abduction in classification tasks. In A.Cappelli and F.Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence*, volume 2829 of *LNAI*, pages 213–224. SV, 2003.

[AMT05] Maurizio Atzori, Paolo Mancarella, and Franco Turini. Memory-aware frequent k-itemset mining. In Francesco Bonchi and Jean Francois Boulicaut, editors, *KDID 2005, Knowledge Discovery in Inductive Databases, Proceedings of the Fourth International Workshop on Knowledge Discovery in Inductive Databases, Revised Selected Papers*, volume 3933 of *Lecture Notes in Computer Science*. Springer, 2005.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[AS00] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD on Management of Data*, pages 439–450, 2000.

[AVNDB01] Ofer Arieli, Bert Van Nuffelen, Marc Denecker, and Maurice Bruynooghe. Coherent composition of distributed knowledge-bases through abduction. In *Logic for Programming, Artificial Intelligence, and Reasoning, 8th International Conference, LPAR 2001, Havana, Cuba, December 3-7, 2001, Proceedings*, volume 2250 of *Lecture Notes in Computer Science*, pages 620–635. Springer, 2001.

[AW91] Nabil R. Adam and John C. Wortmann. Security-control methods for statistical databases: A comparative study. In *ACM Computing Surveys*, number 4 in 21, pages 515–556, 1991.

[BATJ91] Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49(1-3):25–60, 1991.

[BC02] P. Scheuermann B. Chen, P.J. Haas. Fast: A new sampling-based algorithm for discovering association rules. In *18th International Conference on Data Engineering*, 2002.

[BD99] Ivan Bratko and Saso Dzeroski, editors. *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999.* Morgan Kaufmann, 1999.

[Ber02] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[BGMP03] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Examiner: Optimized level-wise frequent pattern mining with monotone constraint. In *ICDM*, pages 11–18, 2003.

[Bor05] Christian Borgelt. Keeping things simple: Finding frequent item sets by recursive elimination. In *Workshop Open Software for Data Mining (OSDM05)*, 2005.

[Cal04] T. Calders. Computational complexity of itemset frequency satisfiability. In *Proc. PODS Int. Conf. Princ. of Database Systems*, 2004.

[CG02] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proceedings of the 6th PKDD*, 2002.

[CH91] Eugene Charniak and Saadia Husain. A new admissible heuristic for minimal-cost proofs. In *AAAI*, pages 446–451, 1991.

[CHN$^+$96] D.W. Cheung, Jiawei Han, V.T. Ng, A.W. Fu, and Yongjian Fu. A fast distributed algorithm for mining association rules. In *l4th International Conference on Parallel and Distributed Information Systems (PDIS '96)*, 1996.

[Chr00] H. Christiansen. Abduction and induction combined in a metalogic framework. In A. Kakas P. Flach, editor, *Abductive and Inductive Reasoning: Essays on their Relation and Integration*, 2000.

[CHS02] Bin Chen, Peter Haas, and Peter Scheuermann. A new two-phase sampling based algorithm for discovering association rules. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–468. ACM Press, 2002.

[CKV02a] Chris Clifton, Murat Kantarcioglu, and Jaideep Vaidya. Defining privacy for data mining. In *Natural Science Foundation Workshop on Next Generation Data Mining*, 2002.

[CKV+02b] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, 2002.

[CM00] LiWu Chang and Ira S. Moskowitz. An integrated framework for database inference and privacy protection. In *Data and Applications Security*, 2000.

[Coh95] William W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.

[Com03] SIGKDD Executive Committee. Data mining is not against civil liberties (open letter). `http://www.acm.org/sigs/sigkdd/`, 2003.

[CS94] Eugene Charniak and Solomon Eyal Shimony. Cost-based abduction and map explanation. *Artificial Intelligence*, 66(2):345–374, 1994.

[DA01] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, 2001.

[DAR03] DARPA. Total information awareness project. `http://www.eff.org/Privacy/TIA/overview.php`, 2003.

[DR98] L. De Raedt. Attribute value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *ILP98*, volume 1446 of *LNAI*, pages 1–8. Springer, 1998.

[DS98] Marc Denecker and Danny De Schreye. Sldnfa: An abductive procedure for abductive logic programs. *J. Log. Program.*, 34(2):111–167, 1998.

[DVEB01] Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. In *Proceedings of the 4th International Workshop on Information Hiding*, 2001.

[DZ02] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE international conference on Privacy, security and data mining*, 2002.

[ECB99] Vladimir Estivill-Castro and Ljiljana Brankovic. Data swapping: Balancing privacy against precision in mining for logic rules. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*, 1999.

[EG95] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.

[EGS03] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2003.

[EK89] K. Eshgi and R.A. Kowalski. Abduction compared with negation by failure. In G. Levi and M. Martelli, editors, *Proc. of the 1989 International Conference on Logic Programming*, pages 234–254. MIT Press, 1989.

[ESAG02] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[Evf02] Alexandre Evfimievski. Randomization in privacy preserving data mining. *SIGKDD Explor. Newsl.*, 4(2), 2002.

[Fab73] C.M. Fabiani. *Organon: Categorie, Dell'espressione, Primi Analitici, Secondi Analitici (Italian).* 1973.

[FK97] Tzee Ho Fung and Robert A. Kowalski. The iff proof procedure for abductive logic programming. *J. Log. Program.*, 33(2):151–165, 1997.

[FK00] Peter A. Flach and Antonis C. Kakas, editors. *Abduction and Induction: Essays on their relation and integration.* Kluwer Academic Publishers, April 2000.

[FR04] Peter Fule and John F. Roddick. Detecting privacy and ethical sensitivity in data mining results. In *Proc. of the 27th conference on Australasian computer science*, 2004.

[Fre00] Alex Alves Freitas. Understanding the crucial differences between classification and discovery of association rules - a position paper. *SIGKDD Explorations*, 2(1):65–69, 2000.

[Fun04] Electronic Frontier Fundation. Rejection of requests to track cell phones without a search warrant. `http://www.eff.org/Privacy/TIA/overview.php`, 2004.

[Goe] Bart Goethals. Frequent itemset mining dataset repository. `http://fimi.cs.helsinki.fi/data/`.

[Goe03] B. Goethals. Survey on frequent pattern mining, 2003.

[Goe04] B. Goethals. Memory issues in frequent itemset mining. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC'04)*. ACM, 2004.

[GSG99] G. Gupta, A. Strehl, and J. Ghosh. Distance based clustering of association rules, 1999.

[GZ03] B. Goethals and Mohammed J. Zaki. Advances in frequent itemset mining implementations: Introduction to fimi'03. In *Proceedings of the FIMI'03 Workshop on Frequent Itemset Mining Implementations.*, 2003.

[HB99] S. Hettich and S. D. Bay. The uci kdd archive, irvine, ca: University of california. `http://kdd.ics.uci.edu/`, 1999.

[HGN00] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[HK00] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[HMP$^+$00] Sanda M. Harabagiu, Dan I. Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan C. Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. Falcon: Boosting knowledge for answer engines. In *TREC*, 2000.

[HMS01] David Hand, Heikki Mannila, and Padhraic Smyh. *Principles of Data Mining.* The MIT Press, 2001.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 1–12. ACM, 2000.

[HSAM93] Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul Martin. Interpretation as abduction. *Artificial Intelligence*, 63(1-2):69–142, 1993.

[IB04] Md. Zahidul Islam and Ljiljana Brankovic. A framework for privacy preserving classification in data mining. In *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 163–168, 2004.

[IGA02] Ioannis Ioannidis, Ananth Grama, and Mikhail Atallah. A secure protocol for computing dot-products in clustered and distributed environments. In *Proceedings of the International Conference on Parallel Processing (ICPP'02)*, 2002.

[IM02] Mitsuru Ishizuka and Yutaka Matsuo. Sl method for computing a near-optimal solution using linear and non-linear programming in cost-based hypothetical reasoning. *Knowl.-Based Syst.*, 15(7):369–376, 2002.

[Jr.94] Eugene Santos Jr. A linear constraint satisfaction approach to cost-based abduction. *Artificial Intelligence*, 65(1):1–28, 1994.

[JWBV] Davy Janssens, Geert Wets, Tom Brijs, and Koen Vanhoof. Integrating classification and association rules by proposing adaptations to the cba algorithm.

[KC01] J. Kacprzyk and K.J. Cios, editors. *Medical Data Mining and Knowledge Discovery.* Physica-Verlag, 2001.

[KC02] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *In The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, 2002.

[KDWS03] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining*, 2003.

[KDWS05] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowl. Inf. Syst.*, 7(4):387–414, 2005.

[KJC04] Murat Kantarcioglu, Jiashun Jin, and Chris Clifton. When do data mining results violate privacy? In *Proceedings of the tenth ACM SIGKDD*, 2004.

[KKT93] A.C. Kakas, R.A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[KM90] A.C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In *Proc. of 9th European Conference on Artificial Intelligence*, pages 385–391. Pitman, 1990.

[KM98] Antonis C. Kakas and A. Michael. Applications of abductive logic programming. In *IJCSLP '98*, pages 343–344, 1998.

[KM99] Antonis C. Kakas and A. Michael. Air-crew scheduling through abduction. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 600–611, 1999.

[KMM00] A.C. Kakas, A. Michael, and C. Mourlas. Aclp: Abductive constraint logic programming. *Journal of Logic Programming*, 44(1-3):129–177, July-August 2000.

[Knu97] D.E. Knuth. *Fundamental Algorithms.* Addison-Wesley, Reading, Massachusetts, 1997.

[KPR99] Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. A microeconomic view of data mining. *J. Data Mining and Knowledge Discovery*, 1999.

[KR00] A. Kakas and F. Riguzzi. Abductive concept learning. *New Generation Computing*, 18(3), 2000.

[KSI97] Shohei Kato, Hirohisa Seki, and Hidenori Itoh. A parallel implementation of cost-based abductive reasoning. In *PASCO '97: Proceedings of the second international symposium on Parallel symbolic computation*, pages 111–118, New York, NY, USA, 1997. ACM Press.

[KSP03] Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. In *Proceedings of the ACM PODS 2003*, volume 28. ACM, 2003.

[KVND01] C. Kakas, Antonis, Bert Van Nuffelen, and Marc Denecker. A-system: Problem solving through abduction. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, volume 1, pages 591–596. IJCAI, inc and AAAI, Morgan Kaufmann Publishers, Inc, 2001.

[LHP01] Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *ICDM*, pages 369–376, 2001.

[LMMR99] E. Lamma, P. Mello, M. Milano, and F. Riguzzi. Integrating induction and abduction in logic programming. *Information Science*, 116(1):25–54, 1999.

[LSW97] Brian Lent, Arun N. Swami, and Jennifer Widom. Clustering association rules. In *ICDE*, pages 220–231, 1997.

[Men96] T. Menzies. Applications of abduction: Knowledge level modeling, 1996.

[MI02] Yutaka Matsuo and Mitsuru Ishizuka. Two transformations of clauses into constraints and their properties for cost-based hypothetical reasoning. In *PRICAI '02: Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence*, pages 118–127, London, UK, 2002. Springer-Verlag.

[Mie03] Taneli Mielikäinen. On inverse frequent set mining. In Wenliang Du and Christopher W. Clifton, editors, *2nd Workshop on Privacy Preserving Data Mining (PPDM).* IEEE Computer Society, 2003.

[Mit97] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[MS99] Krishnamurty Muralidhar and Rathindra Sarathy. Security of random data perturbation methods. *ACM Trans. Database Syst.*, 24(4), 1999.

[MT03] P. Mancarella and G. Terreni. An abductive proof procedure handling active rules. In A.Cappelli and F.Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence*, volume 2829 of *LNAI*, pages 105–117. SV, 2003.

[OI97] Yukio Ohsawa and Mitsuru Ishizuka. Networked bubble propagation: A polynomial-time hypothetical reasoning method for computing near-optimal solutions. *Artificial Intelligence*, 91(1):131–154, 1997.

[OY97] Yukio Ohsawa and Masahiko Yachida. An extended polynomial solvability of cost-based abduction (poster session abstracts). *International Joint Conference of Artificial Intelligence (IJCAI'97)*, 1997.

[OZ02] Stanley R. M. Oliveira and Osmar R. Zaiane. Privacy preserving frequent itemset mining. In *Proceedings of the IEEE international conference on Privacy, security and data mining*, 2002.

[OZ03] Stanley R. M. Oliveira and Osmar R. Zaiane. Protecting sensitive knowledge by data sanitization. In *Third IEEE International Conference on Data Mining (ICDM'03)*, 2003.

[OZS04] Stanley R. M. Oliveira, Osmar R. Zaiane, and Yucel Saygin. Secure association rule sharing. In *Proc.of the 8th PAKDD*, 2004.

[PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT '99*, 1999.

[Pei] C. S. Peirce. Collected papers of Charles Sanders Peirce. Vol. 2, Hartshorn et al. eds., Harvard Univesity Press.

[PHW03] Jian Pei, Jiawei Han, and Jianyong Wang. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *SIGKDD '03*, 2003.

[Pin02] Benny Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explor. Newsl.*, 4(2), 2002.

[Poo93] David Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[Qui89] J. R. Quinlan. Unknown attribute values in induction. In *Proc. of the Sixth International Machine Learning Workshop*, pages 164–168. Morgan Kaufmann, 1989.

[Qui93] J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann, 1993.

[RH02] Shariq Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB Conference*, 2002.

[SIC] `http://www.sics.se/sicstus/`.

[SON95] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *The VLDB Journal*, pages 432–444, 1995.

[SS96] Eugene Jr. Santos and Eugene S. Santos. Polynomial solvability of cost-based abduction. *Artificial Intelligence*, 86(1):157–170, 1996.

[SS98] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington*, 1998.

[Sti91] Mark E. Stickel. A prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. *Ann. Math. Artificial Intelligence*, 4:89–105, 1991.

[SVC01] Yucel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Rec.*, 30(4), 2001.

[Swe00] L. Sweeney. Uniqueness of simple demographics in the u.s. population. Technical report, CMU, 2000.

[Swe02a] Latanya Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty Fuzziness and Knowledge-based Systems*, 10(5), 2002.

[Swe02b] Latanya Sweeney. k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty Fuzziness and Knowledge-based Systems*, 10(5), 2002.

[TKS02] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41. ACM Press, 2002.

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *In Proc. 1996 Int. Conf. Very Large Data Bases*, pages 134–145. Morgan Kaufman, 09 1996.

[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 *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[VNK01] Bert Van Nuffelen and C. Kakas, Antonis. A-System : Programming with abduction. In *Logic Programming and Nonmonotonic Reasoning, LPNMR 2001, Proceedings*, volume 2173 of *LNAI*, pages 393–396. Springer Verlag, 2001.

[WWWL05] Xintao Wu, Ying Wu, Yongge Wang, and Yingjiu Li. Privacy aware market basket data set generation: A feasible approach for inverse frequent set mining. In *Proc. 2005 SIAM Int. Conf. on Data Mining*, 2005.

[Zak00] M. J. Zaki. Scalable algorithms for association mining. In *IEEE Transactions on Knowledge and Data Engineering*, pages 372–390. ACM Press, 2000.

[ZG98] Jean-Daniel Zucker and Jean-Gabriel Ganascia. Learning structurally indeterminate clauses. In *International Workshop on Inductive Logic Programming*, pages 235–244, 1998.

[ZH02] Mohammed J. Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemsets mining. In *2nd SIAM International Conference on Data Mining*, 2002.

[ZO98] Mohammed J. Zaki and Mitsunori Ogihara. Theoretical foundations of association rules. In *In Proceedings of 3 rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, 1998.

“Before I came here I was confused about this subject.
Having listened to your lecture I am still confused. But on a higher level.”
Enrico Fermi