



Project no.: **FP6-516169**

Project acronym: **IQ**

Project title: **Inductive Queries for Mining Patterns and Models**

Instrument: **Specific Target Research Project**

Thematic Priority: **Information Society Technologies**

Deliverable D5.R/P.C

Theory of Inductive Databases

Due date of deliverable: **31 Aug 2008**

Actual submission data: **12 Sep 2008**

Start date of project: **1 Sep 2005**

Duration: **36 months**

Organization name of lead contractor for this deliverable:

(1) Katholieke Universiteit Leuven

Revision: **final**

Project co-funded by the European Commission with the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission service)	
RE	Restricted to group specified by the consortium (including the Commission service)	
CO	Confidential, only for member of the consortium (including the Commission service)	X

Authors of the deliverable: Siegfried Nijssen (KULeuven), Adriana Prado (UA), Bart Goethals (UA), Ella Bingham (UH), Sašo Džeroski (JSI), Jean-François Boulicaut (INSA)

1 Introduction

This deliverable investigates the theory of inductive queries. It lists the new results that were obtained in the third year of the project. This work represents both a continuation of research started in the first two years and new directions of research. In the third year, more attention was devoted to local pattern mining algorithms than originally planned. Given that in previous years more attention was given to model construction, this overall means that we have met the objectives of the project.

We studied the following topics:

From local patterns to global models: This topic has become one of the key topics in the IQ project. Our studies of the previous years involving pattern set selection and tiling have continued; additionally, we have obtained new results relating subgroup discovery, emerging patterns and contrast set mining to each other, and investigated the use of generator itemsets as features in classification tasks. Theoretical properties of the set cover problem, which is related to the problem of pattern selection, were investigated.

Induction of Decision Trees under Constraints: We continued our studies of our exhaustive, pattern mining based algorithm for inducing decision trees. Attention was given to adding novel constraints in this algorithm.

Generalizing Itemset Mining: An important new topic that we studied is the development of generalized systems for mining itemsets by using the principles of constraint programming.

Fault Tolerance: We continued our studies of methods to deal with fault tolerance in local pattern mining; we concentrated our efforts on fault tolerance in sequence mining.

New Patterns: We investigated novel pattern mining tasks, among which redescription mining and patterns with high visibility.

Regular Expression Constraints in String Mining: A new topic in the third year was the development of substring mining algorithms for finding substrings under frequency and regular expression constraints.

ProbLog: in the first two years, we developed a probabilistic logic called PROBLOG. We studied its application in typical data mining problems such as pattern mining in the final year.

A Data Mining Framework: the IQ project covers a broad range of topics, which makes it an important issue to study how these topics relate to each other. We studied distance-based learners in this framework in the third year, and used a functional programming language to define general distance functions between structured objects.

Distributions as Tables: the principles for the development of general inductive querying languages for probabilistic models were studied; as a possible solution the representation of distributions as tables was proposed.

2 From Local Patterns to Global Models

An important topic throughout the project has been the use of patterns in the construction of global models, both for unsupervised and supervised tasks.

We discuss our contributions involving only unsupervised modeling in Section 2.4 and Section 2.5.

Considering supervised tasks, a popular use of pattern miners is as feature generators for classification algorithms. An overview of the use of patterns to build classifiers, is provided in a chapter of the forthcoming IQ book [BNZ08]. In this chapter we distinguish several approaches.

First, there are the integrated approaches, in which pattern miners and classification algorithms are integrated with each other. DL8 (see Section 3) and Tree² (see Section 2.3) are examples of such approaches contributed by the project. Typically, in these methods the model construction algorithms guide which constraints are used by an integrated pattern miner; this pattern mining is not visible for an end-user.

Second, there are the stepwise approaches. In the stepwise approaches, we first apply a pattern miner to generate features under constraints. Feature selection is applied to these features to generate a more manageable set of patterns. The database is re-encoded using the selected features and used as input for a traditional classification algorithm.

Several aspects of this step-wise approach have been studied in detail by the project partners. Some of these results are reported in other deliverables; in particular,

- the problem of selecting a subset of patterns has a close relationship to the problem of finding a good condensed representation, which is also discussed in D4.4C, for those results that are not applicable to model construction in particular or which are not theoretical in nature;
- some applications of using patterns as features are reported in D1C.

From a theoretical aspect, the important tasks are to determine which constraints should be applied in the pattern mining process (Section 2.1) and which pattern selection strategies should be applied (Section 2.2).

References

[BNZ08] Bjorn Bringmann, Siegfried Nijssen, and Albrecht Zimmermann. From local patterns to classification models. 2008. [IQ Book].

2.1 Constraints for Classification

In its final year, the project contributed several insights to which constraints should be used in the stepwise approach.

Based on the observation that many supervised rule learning tasks and approaches are very related [ZR04], in [KLW08, ZR08] we studied a unifying framework for supervised descriptive rule discovery. In particular, correlated pattern mining (CP), contrast set mining (CSM), emerging pattern mining (EPM), and subgroup discovery (SD) were shown to be special cases of a general approach.

In [KLW08] we studied the CSM, EPM and SD problems. We showed that various rule learning heuristics used in CSM, EPM and SD algorithms are compatible with each other: the heuristics can be rewritten into each other, and given a fixed database, optimizing one heuristics is equivalent to optimizing the other heuristics. Additionally, in this work we study how the results of these approaches can be visualized.

In [ZR08], which is the journal version of [ZR04], a larger number of correlation measures was studied. Multi-target and multi-value prediction were also studied. Not all measures

proposed for these settings are equivalent to each other, but the problems were shown to be similar enough that a general algorithm is capable of dealing with all of these tasks.

To reduce the number of patterns one could argue that it is useful to push constraints aiming at condensed representations in the mining process. In [GKL08] we developed a relevancy theory for closed sets on labeled data. It was proved that closed sets characterize the space of relevant combinations of features for discriminating the target class. The practical consequences of the theory were investigated and relevant closed sets were used to efficiently learn subgroup descriptions on a high dimensional microarray data set.

Another type of condensed representation are the generator itemsets. When considering a feature construction perspective, the question whether closed sets are better or not than their generators is a bit unclear. Li et al. [LLW⁺06] use the MDL principle and suggest that free itemsets might be better than closed ones. Last but not the least, one should also consider the possibility to use a near equivalence class perspective (i.e., a δ -freeness property and thus, roughly speaking, the concept of almost-closed itemsets), which we studied in the first two years of the project (see, e.g., [SLGB06]). It turns out that feature construction approaches based on closedness properties differ in two main aspects: (i) mining can be performed on the whole database or per class, and (ii) we can mine with or without the class labels. In the delivered paper [GSB08], we discuss these issues and we contribute to difficult classification tasks by using a method based on: (1) the efficient extraction of set patterns that satisfy user-defined constraints, (2) the encoding of the original data into a new data set by using extracted patterns as new features. Clearly, one of the technical difficulties is to discuss the impact of the intrinsic properties of these patterns (i.e., closedness-related properties) on the classification process.

References

- [GKL08] Gemma C. Garriga, Petra Kralj, and Nada Lavrac. Closed sets for labeled data. *Journal of Machine Learning Research*, 9:559–580, 2008. [IQ, see D4].
- [GSB08] Dominique Gay, Nazha Selmaoui, and Jean-François Boulicaut. Feature construction based on closedness properties is not that simple. In *Proceedings Pacific-Asia Conference on Knowledge Discovery and Data Mining PaKDD'08*, volume 5012 of *LNCS*, pages 112–123. Springer, 2008. [IQ].
- [KLW08] Petra Kralj, Nada Lavrac, and Geoffrey L. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup discovery. *Journal of Machine Learning Research*, 2008. To appear. [IQ].
- [LLW⁺06] Jinyan Li, Haiquan Li, Limsoon Wong, Jian Pei, and Guozhu Dong. Minimum description length principle : generators are preferable to closed patterns. In *Proceedings AAAI'06*, pages 409–415. AAAI Press, 2006.
- [SLGB06] Nazha Selmaoui, Claire Leschi, Dominique Gay, and Jean-François Boulicaut. Feature construction and delta-free sets in 0/1 samples. In *Proceedings International Conference on Discovery Science DS'06*, volume 4265 of *LNAI*, pages 363–367. Springer, 2006.
- [ZR04] Albrecht Zimmermann and Luc De Raedt. Cluster-grouping: From subgroup discovery to clustering. In *ECML*, volume 3201 of *Lecture Notes in Computer Science*, pages 575–577. Springer, 2004.
- [ZR08] Albrecht Zimmermann and Luc De Raedt. Cluster-grouping: From subgroup discovery to clustering. *Machine Learning*, 2008. To appear. [IQ].

2.2 Mining Pattern Sets

An intermediate step in the stepwise approach for classification using patterns is the feature or pattern selection step.

The simplest setting is when we have a binary classification problem. This problem was studied in [BZ08], in which we developed our method for heuristic pattern set mining of [BZ07] further. In this approach we assume that we have pre-computed a set of patterns under a correlation constraint. We investigated a heuristic method (called BOUNCER) for determining a compact set of patterns. This method takes the following steps:

1. it mines patterns with a traditional pattern mining algorithm, such as correlated pattern miner;
2. it sorts these patterns according to size, support, correlation, or any other measure;
3. the patterns are scanned in sorting order, and for each pattern it is decided if it is added to a pattern set under construction by measuring how the *blocks* change. A block is a set of examples that for a selected set of attributes all have the same attribute values.

In [BZ08] we considered variations of this approach, including PICKER. This variation does not operate on a sorted set of patterns but repeatedly searches for a pattern in the pre-computed set that best complements the selected set of patterns. The resulting approach is computationally more intensive, but may be more accurate. More details about this work can be found in the deliverables of workpackage 4.

The problem becomes more complicated when we have classification task with many imbalanced classes. The pioneering proposal of [LHM98], but also most of its improvements (i.e., CMAR, CPAR, etc.), are based on the classical objective interestingness measures for association rules – *frequency* and *confidence* – for selecting candidate classification rules. Support-confidence-based methods show their limits on imbalanced data sets. Indeed, rules with high confidence can also be negatively correlated. [AC06, VC07] have suggested new methods based on correlation measure to overcome this weakness. However, when considering a n -class imbalanced context, even a correlation measure is not satisfactory: a rule can be positively correlated with two different classes, which leads to conflicting rules. The common problem of these approaches is that they are OVA (one-vs-all) methods, i.e., they split the classification task into n two-class classification tasks (positives vs negatives) and, for each sub-task, look for rules that are relevant in the *positive* class and irrelevant for the union of the other classes. Therefore, we have been considering an OVE (one-vs-each) method that avoids some of the problems observed with typical CBA-like methods. In [CGSB08] we formally characterize the association rules that can be used for classification purposes when considering that a rule has to be relevant for one class and irrelevant for every other class (instead of being irrelevant for their union). This provides a more effective pattern selection strategy in the case of many classes.

Additionally, we also designed a constrained hill climbing technique that automatically tunes the many parameters (frequency thresholds) that are needed when using such a step-wise approach.

A problem closely related to pattern selection is the set cover problem. It is usually desirable to select a subset of patterns such that every example in the data is covered by at least one pattern; to do so with a small set of (possibly small) patterns is a set cover problem. To further our understanding in the fundamental problem of pattern selection, we studied several variations of the set cover problem, which are reported in more detail in Section 6.3. These set cover problems are relevant for pattern selection:

- the universal set cover problem can be used as part of a pattern set selection strategy in which we wish to precompute a set of patterns such that for every subset of the data we can easily determine an expectedly cheap set of patterns to cover those examples, without having to recompute the set cover from scratch.

- the positive-negative partial set cover problem can be used to select a set of patterns in data with binary classes, such that we have a set of patterns covering (simultaneously) as many of the positive examples and as few of the negative examples as possible.

References

- [AC06] Bavani Arunasalam and Sanjay Chawla. CCCS: A Top-down Associative Classifier for Imbalanced Class Distribution. In *Proceedings ACM International Conference on Knowledge Discovery and Data Mining KDD'06*, pages 517–522, 2006.
- [BZ07] Bjorn Bringmann and Albrecht Zimmermann. The chosen few: identifying valuable patterns. In *Proceedings IEEE International Conference on Data Mining ICDM'07*, pages 63–72, 2007. [IQ, see D4].
- [BZ08] Bjorn Bringmann and Albrecht Zimmermann. One in a million: picking the right patterns. *Knowledge and Information Systems*, 2008. In Press. [IQ, see D4].
- [CGSB08] Loïc Cerf, Dominique Gay, Nazha Selmaoui, and Jean-François Boulicaut. A parameter-free associative classification method. In *Proceedings 10th International Conference on Data Warehousing and Knowledge Discovery DaWaK'08*, volume 5182 of *LNCS*, pages 293–314. Springer, 2008. [IQ].
- [LHM98] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating Classification and Association Rule Mining. In *Proceedings International Conference on Knowledge Discovery and Data Mining KDD'98*, pages 80–86. AAAI Press, 1998.
- [VC07] Florian Verhein and Sanjay Chawla. Using Significant Positively Associated and Relatively Class Correlated Rules for Associative Classification of Imbalanced Datasets. In *Proceedings IEEE International Conference on Data Mining ICDM'07*, pages 679–684, 2007.

2.3 Using Sets of Patterns in Ensemble Trees

In the first year of the project we developed a method called Tree² for classifying tree structured data. Tree² is a method in which a pattern miner is integrated in a classification algorithm. In the third year of the project we studied an extension of this method for attribute-value data.

The main idea in Tree² was to compute patterns for tree-shaped data while learning a decision tree, instead of using a precomputed set of patterns as features. The method was based on the observation that we can repeatedly search for a splitting criterion by looking for a pattern that maximizes entropy (or correlation) in a subset of the data; this is possible because entropy is a convex function for which good bounds exist.

The extension that we studied in [Zim08], is to use *ensembles* of patterns, instead of one pattern, to split the data in an internal node of the decision tree. Given an example, every pattern casts a vote for the branch of the decision tree in which the example should end up; the majority vote is chosen. The set of patterns can be determined by using top-*k* patterns, instead of one single pattern, according to a correlation measure. The approach was evaluated on attribute-value data of the UCI repository.

References

- [Zim08] Albrecht Zimmermann. Ensemble-trees: Leveraging ensemble power inside decision trees. In *Proceedings International Conference Discovery Science DS'08*, volume 5255 of *LNAI*. Springer, 2008. In Press. [IQ].

2.4 Co-Clustering using Patterns

During the first two years, we have considered the uses of local patterns, such as constrained bi-sets in 0/1 data, not only for supervised, but also for unsupervised modeling tasks. During Year 3, we have continued our work on a generic framework for co-clustering 0/1 data, which has recently led to a submission to a journal [PRB08b]. Notice also that the added-value of such a Local to Global approach has been demonstrated in the first contribution to constrained co-clustering published as [PRB08a] during Year 3 and delivered in D3C.

Related to the problem of co-clustering is the task of tiling. Tiling aims at discovering (small) collections of local patterns that can represent the encoded relation. Indeed, usually, a relation is extensionally defined as a set of tuples. Storing and retrieving an n -ary relation by listing all its tuples one by one is both time and space consuming. Hence, minimizing the expression of an arbitrary n -ary relation is an interesting problem and the *tiling task* consists of finding a collection of patterns (say tiles) that is as compact as possible but still entirely expresses the intended relation. Choosing the tiles to be closed n -sets looks like a clever idea (e.g., selecting a minimal number of formal concepts that cover all the true values of a binary relation). We started to investigate such an application in [CBRB08] (delivered in D3C) and the chapter [CRG⁺08] delivered here develops further this issue.

References

- [CBRB08] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Closed patterns meet n -ary relations. Technical report, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France, April 2008. Under minor revision for ACM Transactions on KDD. [IQ, see D4C].
- [CRG⁺08] Loïc Cerf, Céline Robardet, Bart Goethals, Jérémy Besson, and Jean-François Boulicaut. (tentative list of authors and worktitle) computing tiles and tiling from n -ary relations. Technical report, 2008. [IQ Book].
- [PRB08a] Ruggero G. Pensa, Céline Robardet, and Jean-François Boulicaut. *Constrained Clustering: Advances in Algorithms, Theory and Applications*, chapter Constraint-driven Co-Clustering of 0/1 Data, pages 145–170. Chapman & Hall/CRC Press, 2008. [IQ, see D3C].
- [PRB08b] Ruggero G. Pensa, Céline Robardet, and Jean-François Boulicaut. From local to global patterns: a co-clustering framework. Technical report, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France, August 2008. 30 pages. Submitted to Data & Knowledge Engineering. [IQ].

2.5 Using Local Patterns to Summarize Binary Data

While itemsets can be used for common machine learning tasks such as classification and clustering, we also investigated their use in other tasks. A major contribution in this is the PhD dissertation of Nikolaj Tatti [Tat08]. The dissertation deals with the following problems related to using frequent itemsets to represent the whole data set:

How to use itemsets for answering queries, that is, finding out the number of transactions satisfying some given formula? While this is a simple procedure given the original data, the task transforms into a computationally infeasible problem if we seek the solution using the itemsets. By making some assumptions of the structure of the itemsets and applying techniques from the theory of Markov Random Fields we are able to reduce the computational burden of query answering.

How to use the known itemsets to predict the unknown itemsets? The difference between the prediction and the actual value can be used for ranking itemsets. In fact, this method can be seen

as generalisation for ranking itemsets based on their deviation from the independence model, an approach commonly used in the data mining literature.

How to use itemsets to define a distance between two datasets? We achieve this by computing the difference between the frequencies of the itemsets. We take into account the fact that the itemset frequencies may be correlated and by removing the correlation we show that our distance transforms into Euclidean distance between the frequencies of parity formulae.

References

[Tat08] Nikolaj Tatti. *Advances in Mining Binary Data: Itemsets as Summaries*. 2008. Ph.D. Thesis. [IQ].

3 Mining Decision Trees under Constraints

In the previous two years we studied two types of approaches to find decision trees under constraints.

First, we developed heuristic-driven approaches, possibly upgraded by using a beam-search.

The use of these approaches has been continued in the final year; results of their application are reported in deliverables 1 and 3.

Second, we developed a complete search approach, which we called DL8. The distinguishing feature of DL8 is that it relies on itemset mining algorithms to build a lattice of itemsets, which is post-processed to select a subset of itemsets that represent a decision tree. DL8 integrates pattern mining and model construction in a single algorithm.

Examples of constraints and optimisation criteria that we studied in the second year [NF07] were:

- a maximum size constraint;
- a leaf support constraint;
- a maximum depth constraint;
- optimising accuracy;
- optimising a pruning measure;
- optimising size.

We extended this approach in a journal version [NF08]. To show the generality of our framework, we showed that we can also easily deal with the following additional constraints and optimisation criteria:

- cost-based learning, incorporating both misclassification costs and prediction costs;
- privacy preserving decision tree induction, where we showed how we can incorporate both k -anonymity and a novel ℓ -diversity constraint in the induction process;
- Bayesian decision tree learning, where we discuss how to find a maximum a posteriori (MAP) tree that optimizes a Bayesian optimisation criteria that takes into account both likelihood and prior preferences, such as that small trees should have higher probability of being correct.

Continuing our work on Bayesian decision tree learning, we presented an approach for performing Bayes optimal predictions in [Nij08]. A prediction is called Bayes optimal for decision trees if it corresponds to the prediction made by hypothetical classifier that weighs the predictions of all possible decision trees by their posterior. Starting from the observation that the itemset lattice represent a space of decision trees, we developed a method which performs such predictions from the lattice, without explicitly instantiating all the trees in the lattice. The method works in three stages:

- first, a lattice of itemsets is constructed;
- next, this lattice is traversed *two times* in linear time, to compute a weight vector for every itemset;
- finally, for every test example, we determine which itemsets it contains, and sum the weight vectors of those itemsets; the class with the highest weight is the predicted class.

Hence, we can see this method as constructing a rule-based classifier in which rules have well-chosen weights. We proved that the resulting weights yield classifications that exactly correspond to those of a Bayes-optimal classifier for decision trees.

References

- [NF07] Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In *Proceedings ACM International Conference on Knowledge Discovery and Data Mining KDD'07*, pages 530–539, 2007.
- [NF08] Siegfried Nijssen and Elisa Fromont. Optimal constraint-based decision tree learning. *Submitted*, 2008. [IQ].
- [Nij08] Siegfried Nijssen. Bayes optimal classification for decision trees. In *Proceedings International Conference on Machine Learning ICML'08*, pages 696–703. ACM, 2008. [IQ].

4 Generalizing Pattern Mining

Over the last few years many itemset mining algorithms have been developed, within the IQ project, but also outside the project. In the third year two studies were made on how to *generalize* the principles of these itemset mining algorithms; the aim of this work was to make it easier to develop algorithms for new constraints, and if possible, to even avoid implementing new algorithms for every newly proposed constraint.

The ideal in this work was to develop a more *declarative* approach to data mining, in which a user can specify *which* kind of patterns she wishes, without having to worry about *how* a new algorithm for this type of pattern should be implemented; a general solver should still find these patterns with a reasonable performance.

The class of algorithms that we chose to generalize consisted of a large number of *depth-first* itemset mining algorithms, among which Eclat [ZPOL97], MAFIA [BCF⁺05], DualMiner [BGKW03]), LCM [UKA05] and ExaMiner [BL07]. We concluded that the common feature of these algorithms is that they search for *bi-sets*, consisting of pairs of itemsets and transaction sets that satisfy certain constraints. During their search these algorithms maintain *upper* and *lower* bounds of both sets, such that if at a certain point in the depth-first search a solution can be found, this solution must be a superset of the lower bound and a subset of the upper bound. Propagation is used to ensure that these bounds are kept as tight as possible.

Incorporating this observation, we developed two approaches.

First, we developed a general itemset mining algorithm, whose feature is that it maintains upper and lower bounds, and has a plug-in interface to allow for the incorporation of propagators for constraints. This algorithm was used to search for fault tolerant formal concepts with minimal additional implementation effort.

Second, we observed that the idea of maintaining upper and lower bounds of variables during a search is very similar to what is common in the area of *constraint programming*, and we found a way to formulate common itemset mining constraints, such as frequency and closedness, in these existing systems, without having to add new propagators to these systems [DGN08]. We found that a broad range of itemset mining problems can be formulated declaratively in these systems; for appropriate formulations the search is very similar to that of well-known itemset mining systems, although the current implementations are not as fast as the systems that were designed with itemset mining in mind.

A summary of the general approach will be given in [BBD⁺08].

The usability and generality of both approaches showed that there is indeed a general principle behind existing itemset mining algorithms, which can be used in the development of declarative itemset mining systems. Our hope is that by these results, we have brought truly general and declarative inductive querying systems closer to reality.

References

- [BBD⁺08] J  r  my Besson, Jean-Fran  ois Boulicaut, Luc De Raedt, Tias Guns, and Siegfried Nijssen. (Tentative list of authors and worktitle) Generalizing itemset mining in a constraint programming setting. 2008. [IQ Book].
- [BCF⁺05] Douglas Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Mafia: A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504, 2005.
- [BGKW03] Cristian Bucila, Johannes Gehrke, Daniel Kifer, and Walker M. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Min. Knowl. Discov.*, 7(3):241–272, 2003.
- [BL07] Francesco Bonchi and Claudio Lucchese. Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.*, 60(2):377–399, 2007.
- [DGN08] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for itemset mining. In *ACM International Conference on Knowledge Discovery and Data Mining KDD’08*, 2008. In Press. [IQ].
- [UKA05] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. LCM ver.3: collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings International Workshop on Open Source Data Mining OSDM’05*, pages 77–86, 2005.
- [ZPOL97] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In *KDD*, pages 283–286, 1997.

5 Fault-Tolerance

In the first two years, we identified fault-tolerant pattern mining as an important topic of study for many types of local patterns. During Year 3, we have continued to study complex knowledge discovery scenarios that can be based on fault-tolerant string patterns (application to promoter

sequence analysis). Papers [MRS⁺08] and [RMB⁺08] are delivered respectively in D1C and D2C.

Furthermore, our contribution to fault-tolerant local patterns from 0/1 data concerns both fault-tolerant frequent itemset mining and fault-tolerant extensions to closed pattern discovery. In the invited paper [BB08] delivered in D4C, we survey some results for tackling relevancy in the popular context of closed sets from binary relations and fault-tolerance is an important issue there. In [CGGR08], we propose to counterbalance the presence of noise in many 0/1 matrices by mining fault-tolerant itemsets that are based on a relaxation of the support definition. We use the definition from [PTH01] and we design a new algorithm to efficiently compute the frequencies of such patterns. It is based on the *generalized itemsets* and the Quick Inclusion-Exclusion principle introduced in [CG05]. We also addressed the challenging new context of fault-tolerance for n -ary relation mining. [BCTB08] illustrates the crucial need for fault-tolerance when $n \geq 3$. In [CBB08], we provide the first attempt for a complete extraction of fault-tolerant closed patterns from n -ary relations. These papers are delivered in D3C.

References

- [BB08] Jean-François Boulicaut and Jérémy Besson. Actionability and formal concepts: a data mining perspective (invited contribution). In *Proceedings 6th International Conference on Formal Concept Analysis ICFCA'08*, volume 4933 of *LNAI*, pages 14–31. Springer, 2008. [IQ, see D4C].
- [BCTB08] Jérémy Besson, Loïc Cerf, Rémi Thévenoux, and Jean-François Boulicaut. Tackling closed pattern relevancy in n -ary relations. In *Proceedings Workshop Mining Multidimensional Data MMD'08 co-located with ECML/PKDD 2008*, 2008. 15 pages. In Press. [IQ, see D3C].
- [CBB08] Loïc Cerf, Jérémy Besson, and Jean-François Boulicaut. Mining closed fault-tolerant patterns from n -ary relations. Technical report, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France, July 2008. Submitted to IEEE ICDM 2008. [IQ, see D3C].
- [CG05] Toon Calders and Bart Goethals. Quick inclusion-exclusion. In *Knowledge Discovery in Inductive Databases KDID'05 Revised Selected and Invited Papers*, volume 3933 of *LNCSE*, Springer, 2005.
- [CGGR08] Toon Calders, Calin Garboni, Bart Goethals, and Céline Robardet. Using quick inclusion-exclusion for fault-tolerant pattern mining. 2008. Research Report. Submitted to IEEE ICDM 2008, 10 pages. [IQ, see D4C].
- [MRS⁺08] Ieva Mitasiunaitė, Christophe Rigotti, Stéphane Schicklin, Laurène Meyniel, Jean-François Boulicaut, and Olivier Gandrillon. Extracting signature motifs from promoter sets of differentially expressed genes. Technical report, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France, 2008. Submitted to In Silico Biology, 31 pages. [IQ, see D1C].
- [PTH01] Jian Pei, Anthony K. H. Tung, and Jiawei Han. Fault-tolerant frequent pattern mining: Problems and challenges. In *Proceedings ACM SIGMOD Workshop DMKD*, 2001.
- [RMB⁺08] Christophe Rigotti, Ieva Mitasiunaitė, Jérémy Besson, Jean-François Boulicaut, and Olivier Gandrillon. Using a solver over the string pattern domain to analyze gene promoter sequences. Technical report, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France, 2008. [IQ Book].

6 New Types of Patterns

In the previous years we have studied two new patterns and constraints. In the last year of the project we identify some new constraints for defining interesting patterns.

6.1 Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining

Discovering interesting subgroups of observations is one of the key concepts in data mining. Subgroups can be discovered by clustering, where the dataset is typically partitioned into disjoint sets, or by pattern discovery, where each pattern defines the subgroup in which the pattern is true, and thus the patterns can be overlapping. The interestingness of a subgroup is typically measured by how likely or unlikely such a subgroup would be to arise at random.

In general, a subgroup is defined either by the explicit values of the variables of the data (subgroups defined by patterns) or by distances between rows (clusters). Here we consider only subgroups defined by using Boolean formulae on the values of the variables. Any predicate $\alpha(t)$ defined for the rows (observations) t of the dataset D defines a subgroup $\{t \in D | \alpha(t) \text{ is true}\}$. For example, if the data is 0-1, then the formula $A = 1 \wedge (B = 0 \vee C = 1)$ defines a subgroup.

In a paper by UH [GMM08] we consider the redescription mining task introduced by Ramakrishnan, Parida, and Zaki that is the task of finding subgroups having several descriptions. That is, we want to find formulae α and β such that the sets $\{t \in D | \alpha(t) \text{ is true}\}$ and $\{t \in D | \beta(t) \text{ is true}\}$ are about the same. If α and β are logically equivalent, this holds trivially, but we are interested in finding formulae that are not equivalent, but still happen to be satisfied by about the same rows of D . (One might call such formulae D -equivalent.) Another way of looking at the task is that we search for formulae α and β such that the rules $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ both hold with high accuracy.

Such pairs α and β of descriptions indicate that the subset of rows has different definitions, a fact that gives useful information about the data. We give simple algorithms for this task, and evaluate their performance. The methods are based on pruning the search space of all possible pairs of formulae by different accuracy criteria. The significance of the findings is tested by using randomization methods. Experimental results on simulated and real data show that the methods work well: on simulated data they find the planted subsets, and on real data they produce small and understandable results.

References

- [GMM08] Arianna Gallo, Pauli Miettinen, and Heikki Mannila. Finding subgroups having several descriptions: Algorithms for redescription mining. In *Proceedings SIAM Conference on Data Mining SDM'08*, pages 334–345, 2008. [IQ].

6.2 Patterns for Maximum Visibility

In a paper by UH and international collaborators [MDHM08] an interesting new problem setting is discussed: Among all attributes in the data, which ones should I choose for my document, so that as many queries as possible will return my document? — In recent years there has been significant interest in developing ranking functions and retrieval algorithms that will aid users in effective exploration of data bases. In contrast, in this paper the focus is on a complementary problem: how to guide a seller in selecting the best attributes of a new observation (a new product), such that it stands out in the crowd of existing competitive products and is widely visible to potential buyers.

The users of product databases are thus categorized into two groups: *buyers* who search such databases trying to locate objects of interest, and *sellers* who insert new objects into these

databases in the hope that they will be easily discovered by the buyers. Almost all of the prior research on effective search and retrieval techniques has been designed with the buyers in mind, instead of the sellers.

The problem is formulated in several different ways in [MDHM08]. Although these problems are NP-complete, several exact algorithms and approximation heuristics are given. The exact algorithms are based on Integer Programming formulations of the problems, as well as on adaptations of maximal frequent itemset mining algorithms. The approximation algorithms are based on greedy heuristics.

References

- [MDHM08] Muhammed Miah, Gautam Das, Vagelis Hristidis, and Heikki Mannila. Standing out in a crowd: Selecting attributes for maximum visibility. In *Proceedings IEEE International Conference on Data Engineering ICDE'08*, pages 356–365, 2008. [IQ].

6.3 Set Covering

A general problem in computer science is the set covering problem. As pointed out in Section 2.2, the set cover problem is relevant for the problem of selecting patterns. Other applications can be found in the decomposition of binary matrices. We provide more details of the set cover problems that we studied in this section.

The Positive-Negative Partial Set Cover ($\pm$ PSC) problem is a generalization of the Red-Blue Set Cover (RBSC) problem which in turn is a generalization of the classical Set Cover (SC) problem. In RBSC we are given disjoint sets R and B of red and blue elements, respectively, and a collection of sets $\{S_1, \dots, S_n\} \subseteq 2^{R \cup B}$. The goal is to find a collection $C \subseteq \{S_1, \dots, S_n\}$ that covers all blue elements, while minimizing the number of covered red elements. The RBSC problem is much harder than SC admitting the strong inapproximability property, meaning that there exist no polynomial-time approximation algorithms to approximate RBSC within some meaningful accuracy.

In $\pm$ PSC, the requirement of covering all blue elements is relaxed; instead, the goal is to find the best balance between covering the blue elements and not covering the red ones. In this context, the red and blue elements are called negative and positive elements, respectively. In the paper by Miettinen at UH [Mie08], the strong inapproximability of $\pm$ PSC is proved. Using suitable reductions, an approximation algorithm is shown to be possible. The paper also gives results about the parameterized complexity of $\pm$ PSC. The $\pm$ problem can be represented using Boolean products of 0-1 matrices, which nicely connects the problem to many other studies on 0-1 valued data that we have pursued in the recent years.

In another paper by Miettinen from UH and international collaborators [GGL⁺08], the set cover problem is extended in a very different direction compared to that in [Mie08]. Given a universe U of n elements and a weighted collection $\mathcal{S}$ of m subsets of U , the *universal set cover problem* is to a-priori map each element $u \in U$ to a set $\mathbf{S}(u) \in \mathcal{S}$ containing u , so that $X \subseteq U$ is covered by $\mathbf{S}(X) = \bigcup_{u \in X} \mathbf{S}(u)$. The aim is in finding a mapping such that the cost of $\mathbf{S}(X)$ is as close as possible to the optimal set-cover cost for X .

Universal algorithms are useful in distributed settings where decisions are taken locally to minimize the communication overhead. Similarly, in critical applications one wants to pre-compute solutions for a family of scenarios, so as to react faster when the actual input shows up. Moreover, universal mappings can be translated into online algorithms. Unfortunately, for every universal mapping, if the set X is adversarially chosen, the cost of $\mathbf{S}(X)$ can be $\Omega(\sqrt{n})$ times larger than optimal. However, in many applications, the parameter of interest is not the worst-case performance, but instead the performance on average: what can we say when X is a set of randomly chosen elements from the universe? How does the expected cost of $\mathbf{S}(X)$

compare to the expected optimal cost? In the paper [GGL⁺08] the authors present a polynomial-time $O(\log(mn))$ -competitive universal algorithm, and also give a slightly improved analysis and show that this is the best possible.

The algorithm given in [GGL⁺08] is based on interleaving two greedy algorithms: one picking the set with the best ratio of cost to number of uncovered elements, and another picking the cheapest subset that covers any uncovered element. The ideas are generalized for weighted set cover. Applications of the set-cover result to universal multi-cut and disc-covering problems are shown.

References

- [GGL⁺08] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. In *Proceedings Annual Symposium on Foundations of Computer Science FOCS'08*, 2008. To appear. [IQ].
- [Mie08] Pauli Miettinen. On the positive-negative partial set cover problem. *Information Processing Letters*, 2008. To appear. [IQ].

7 Regular Expression Constraints in String Mining

A new topic in the third year was the development of substring mining algorithms for finding substrings under frequency and regular expression constraints [BG08]. An essential contribution in this new approach is the use of *Sequence Mining Automata*, which are a general type of automaton suitable for pattern mining in string domains. The proposed automaton is based on ideas also used in Petri Nets; when a symbol is encountered, a transition labeled with the encountered symbol is allowed to fire if there is a token in its input place. Compared to the most commonly used type of Petri Nets, though, places keep their tokens even if a transition fires. The subsequences that are parsed in this way are stored. In this way, all subsequences satisfying a regular expression in a string can be retrieved, and it is possible to count their support.

A system that integrates these automata was implemented and is reported in deliverable D3C. The automaton is expected to have a wider range of applications.

References

- [BG08] F. Bonchi and B. Goethals. Sequence mining automata: a new technique for mining frequent sequences under regular expressions. 2008. Submitted to IEEE ICDM Int. Conf. on Data Mining [IQ, see D3C].

8 Probabilistic Logics

In the first year of the project the PROBLOG framework was proposed for analysing networks in which edges are labeled with probabilities. The approach was based on representing networks in a probabilistic PROBLOG program, in which each clause is labeled with the probability that it belongs to a randomly sampled program. The probability of a ProbLog query is then defined by the success probability of a query in a randomly sampled program.

In the third year, we identified that PROBLOG can serve as a general framework in which other data mining tasks can be considered within a probabilistic setting. To this aim, in the third year:

- we studied how to speed up our implementation for large datasets (see D4.3);
- we studied how to find patterns within this framework (see D3.3);

- we extended our work on compression, which was first published in [DKK⁺07], to a journal version [DKK⁺08].

The main idea which was studied in [DKK⁺07, DKK⁺08], was how to perform theory compression. Given sets of positive and negative examples of facts that should and should not be provable (for instance, nodes that should be connected and nodes that should not be connected in a network), the task is to find a subset of the clauses for which it is most likely that the connection between positive examples can be proved, while the negative connections cannot. In the extended version the details of our approach are provided, as well as more extensive experiments.

An overview of the results achieved with ProbLog is given in a chapter of the IQ book [DKKT08].

References

- [DKK⁺07] Luc De Raedt, Kristian Kersting, Angelika Kimmig, Kate Revoredo, and Hannu Toivonen. Revising probabilistic prolog programs. In *Inductive Logic Programming ILP 2006 Revised Selected Papers*, volume 4455 of *LNCS*, pages 30–33. Springer, 2007.
- [DKK⁺08] Luc De Raedt, Kristian Kersting, Angelika Kimmig, Kate Revoredo, and Hannu Toivonen. Compressing probabilistic prolog programs. *Machine Learning*, 70(2-3):151–168, 2008. [IQ].
- [DKKT08] Luc De Raedt, Kristian Kersting, Angelika Kimmig, and Hannu Toivonen. (Tentative list of authors and worktitle) Probabilistic relational learning with ProbLog. 2008. [IQ Book].

9 Data Mining Framework: Distance-Based Learning

Among the main motivations behind the IQ project is the need to develop a generally accepted framework for data mining. In year 2, Džeroski [Dze06] addressed this task directly, proposing a general framework for data mining that includes major aspects of inductive databases and inductive queries in the formulation. The framework defines clearly some basic data mining concepts and discusses some of their properties: these are relevant for the further development of both the theory and the practice of data mining, including the development of data mining languages.

A general framework for data mining should elegantly handle different types of data, different data mining tasks, and different types of patterns/models. It should also support the development of languages for data mining that facilitate the design and implementation of data mining algorithms. In this light, Džeroski [Dze06] proposes an inventory of basic data mining concepts, starting with (structured) data and generalizations (e.g., patterns and models), emphasizing the dual nature of the latter, which can be treated both as data and as functions. Four basic data mining tasks are defined (learning a predictive model, clustering, pattern discovery, and estimating the joint probability distribution). The basic components of data mining algorithms (i.e., refinement operators, distances, features and kernels) are also discussed, outlining how to use these to formulate and design generic data mining algorithms.

In year 3, the framework was elaborated upon and was exemplified on the more specific area of distance-based learning with structured data. In particular, a better understanding was developed of how distance functions for complex structured data types are constructed from distances on simpler component types. In addition to the latter, a pairing function on components of the structured objects is needed, as well as an aggregation function which combines the set of pairwise distances between the components into an overall distance value for the structured objects.

Given two structured objects $TC(a_1, a_2, \dots, a_n)$ and $TC(b_1, b_2, \dots, b_m)$, a set of pairs (a_i, b_j) of components is first constructed, based on the type constructor TC and (possibly) the distance functions on the component types. For tuples, $n = m$ and the pairing function is easy, returning pairs (a_i, b_i) . For sets, a variety of options exists and is in use, from the uninformed (a_i, b_j) for each i and j , to sophisticated matchings that use the pairwise distances between pairs to select a set of pairs where each element of the two sets is associated with at most one element of the other set. For sequences, the pairings result from alignments of the sequences.

A variety of aggregation functions exist and are in use in distance functions. The simplest ones are minimum, maximum and average. The n^{th} root of the sum of n^{th} degrees of each distance value is a more complex option, which has as special cases the Manhattan ($n = 1$), Euclidean ($n = 2$), and Chebyshev ($n = \infty$) distances. Recently, ordered weighted aggregates have been introduced that first sort the set of distance function values, then apply a set of weights thereon, dependent on the ordering of the values: minimum, maximum, median and average are all special cases of this aggregation function.

The above has led us to an understanding of the design space for distance functions on structured data and to an approach where we explicitly store, retrieve and manipulate (create/modify) distance functions for structured data. We have implemented an interface for managing a database of structured data types and distance functions thereon, as well as components (distances on primitive types, pairing functions, and aggregation functions) that can be used to create distance functions [AED08]. This has been implemented in the functional programming language Haskell, which supports the higher-order functions necessary to manipulate the distance functions. The functional programming paradigm is more widely relevant to the general framework we adopt, as generalizations (including patterns and models) can be treated as functions.

Once we have an understanding (and an implementation) of the design space for distance functions, we can search this space for appropriate distances for a given dataset. In particular, we explore distance functions with different structures composed from the building blocks available. Preliminary experiments in this direction indicate that (at least for some datasets) much more suitable distances can be found as compared to those used so far.

References

- [AED08] Darko Aleksovski, Martin Erwig, and Saso Džeroski. A functional programming approach to distance-based machine learning. In *Proceedings of the Conference on Data Mining and Data Warehouses SiKDD'08 at the 11th International Multi-conference on Information Society*, 2008. [IQ].
- [Dze06] Saso Džeroski. Towards a general framework for data mining. In Saso Džeroski and Jan Struyf, editors, *KDID*, volume 4747 of *Lecture Notes in Computer Science*, pages 259–300. Springer, 2006.

10 Distributions as Tables

Several desirable properties have been identified for inductive query languages, among which support for both models and patterns, the closure property, a high level of abstraction, and dealing with both logical and probabilistic reasoning. A language which supports all these features however still does not exist, to the best knowledge of the project. Working towards this goal, we have developed the idea of *distributions as tables*. In this approach, we conceptually treat distributions as tables (DATs), similar to the idea of mining views.

DATs can be stored *extensionally*, that is, we can list all possible combinations of attribute values, and their probability (for probabilistic models), but we can also represent them *intensionally*, by a logical or probabilistic model. Basic operations which can be applied to these tables include

enumeration (turning intensional DATs into extensional ones), turning probabilistic models into deterministic ones, turning deterministic models into probabilistic ones, and turning extensional tables into intensional ones (which are the common data mining operations). Given that the DATs correspond to joint probability distributions, operations such as marginalization and conditioning are straightforwardly supported.

In some cases, models do not specify an entire probability distribution. To deal with such models we developed the idea of partial DATs. Using partial DATs and DATs, we found that we can incorporate concept learning, local pattern mining, probabilistic modeling and clustering in a single framework.

These results were summarized in a technical report, which is attached to this deliverable [RBM08].

References

- [RBM08] Luc De Raedt, Hendrik Blockeel and Heikki Mannila. An Inductive Query Language Proposal (Extended Abstract). Technical report, Katholieke Universiteit Leuven and University of Helsinki, 2008. [IQ].

The publications of the associated deliverable D.5.R.B are:

- **(KULeuven)** Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for itemset mining. In *ACM International Conference on Knowledge Discovery and Data Mining KDD'08*. In Press.
- **(UH)** Arianna Gallo, Pauli Miettinen, and Heikki Mannila. Finding subgroups having several descriptions: Algorithms for redescription mining. In *Proceedings SIAM Conference on Data Mining SDM'08*, pages 334–345, 2008.
- **(UH)** Muhammed Miah, Gautam Das, Vagelis Hristidis, and Heikki Mannila. Standing out in a crowd: Selecting attributes for maximum visibility. In *Proceedings IEEE International Conference on Data Engineering ICDE'08*, pages 356–365, 2008.
- **(KULeuven)** A. Zimmermann. Ensemble-trees: Leveraging ensemble power inside decision trees. In *Proceedings International Conference Discovery Science DS'08*, volume 5255 of *LNAI*. Springer, 2008. In Press.
- **(KULeuven)** S. Nijssen and E. Fromont. Optimal constraint-based decision tree learning. *Technical report*, 2008.
- **(KULeuven)** S. Nijssen. Bayes optimal classification for decision trees. In *Proceedings International Conference on Machine Learning ICML'08*, pages 696–703. ACM, 2008.
- **(KULeuven+UH)** L. De Raedt, K. Kersting, A. Kimmig, K. Revoredo, and H. Toivonen. Compressing probabilistic prolog programs. *Machine Learning*, 70(2-3):151–168, 2008.
- **(INSA)** D. Gay, N. Selmaoui, and J-F. Boulicaut. Feature construction based on closedness properties is not that simple. In *Proceedings Pacific-Asia Conference on Knowledge Discovery and Data Mining PaKDD'08*, volume 5012 of *LNCS*, pages 112–123. Springer, 2008.
- **(IJS)** P. Kralj, N. Lavrač, and Geoffrey L. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup discovery. *Journal of Machine Learning Research*, 2008. To appear.
- **(KULeuven)** A. Zimmermann and L. De Raedt. Cluster-grouping: From subgroup discovery to clustering. *Machine Learning*, 2008. To appear.
- **(INSA)** L. Cerf, D. Gay, N. Selmaoui, and J-F. Boulicaut. A parameter-free associative classification method. In *Proceedings 10th International Conference on Data Warehousing and Knowledge Discovery DaWaK'08*, volume 5182 of *LNCS*, pages 293–314. Springer, 2008.
- **(INSA)** R.G. Pensa, C. Robardet, and J-F. Boulicaut. From local to global patterns: a co-clustering framework. Technical report, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France, August 2008. 30 pages. Submitted to Data & Knowledge Engineering.
- **(UH)** N. Tatti. *Advances in Mining Binary Data: Itemsets as Summaries*. 2008. Ph.D. Thesis. .
- **(UH)** F. Grandoni, A. Gupta, S. Leonardi, P. Miettinen, P. Sankowski, and M. Singh. Set covering with our eyes closed. In *Proceedings Annual Symposium on Foundations of Computer Science FOCS'08*, 2008. To appear.
- **(UH)** P. Miettinen. On the positive-negative partial set cover problem. *Information Processing Letters*, 2008. To appear.
- **(IJS)** D. Aleksovski, M. Erwig, and S. Džeroski. A functional programming approach to distance-based machine learning. *Proceedings of the Conference on Data Mining and DataWarehouses SiKDD08 at the 11th International Multi-conference on Information Society*, 2008 To appear.

- **(KULeuven+UH)** Luc De Raedt, Hendrik Blockeel and Heikki Mannila. An Inductive Query Language Proposal (Extended Abstract). Technical report, Katholieke Universiteit Leuven and University of Helsinki, 2008..

Constraint Programming for Itemset Mining

Luc De Raedt

Tias Guns

Siegfried Nijssen

Katholieke Universiteit Leuven, Department of Computer Science
Celestijnenlaan 200A, Leuven, Belgium
{luc.deraedt,tias.guns,siegfried.nijssen}@cs.kuleuven.be

ABSTRACT

The relationship between constraint-based mining and constraint programming is explored by showing how the typical constraints used in pattern mining can be formulated for use in constraint programming environments. The resulting framework is surprisingly flexible and allows us to combine a wide range of mining constraints in different ways. We implement this approach in off-the-shelf constraint programming systems and evaluate it empirically. The results show that the approach is not only very expressive, but also works well on complex benchmark problems.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database applications—*Data Mining*; F.4.1 [Mathematical Logic and Formal Languages]: Mathematical Logic—*Logic and Constraint Programming*

General Terms

Algorithms, Theory

Keywords

Itemset Mining, Constraint Programming

1. INTRODUCTION

For quite some time, the data mining community has been interested in *constraint-based mining*, that is, the use of constraints to specify the desired properties of the patterns to be mined [1, 4, 5, 6, 8, 11, 12, 15, 16, 10]. The task of the data mining system is then to generate all patterns satisfying the constraints. A wide variety of constraints for local pattern mining exist and have been implemented in an even wider range of specific data mining systems.

On the other hand, the artificial intelligence community has studied several types of constraint-satisfaction problems and contributed many *general purpose* algorithms and systems for solving them. These approaches are now gathered

in the area of *constraint programming* [2, 13]. In constraint programming, the user specifies the model, that is, the set of constraints to be satisfied, and the constraint solver generates solutions. Thus, the goals of constraint programming and constraint based mining are similar (not to say identical); it is only that constraint programming targets *any* type of constraint satisfaction problem, whereas constraint-based mining *specifically* targets data mining applications. Therefore, it is surprising that despite the similarities between these two endeavours, the two fields have evolved independently of one another, and also, that – to the best of the authors’ knowledge – constraint programming tools and techniques have not yet been applied to pattern mining, and, vice versa, that ideas and challenges from constraint-based mining have not yet been taken up by the constraint programming community.

In this paper, we bridge the gap between these two fields by investigating how standard constraint-programming techniques can be applied to a wide range of pattern mining problems. To this aim, we first formalize most well-known constraint-based mining problems in terms of constraint programming terminology. This includes constraints such as frequency, closedness and maximality, and constraints that are monotonic, anti-monotonic and convertible, as well as variations of these constraints, such as δ -closedness. We then incorporate them in off-the-shelf and state-of-the-art constraint programming tools, such as Gecode¹ [14] and ECLiPSe² [2], and run experiments. The results are surprising in that 1) using the constraint programming approach, it is natural to combine complex constraints in a flexible manner (for instance, δ -closedness in combination with monotonic and anti-monotonic constraints); unlike in the existing constraint-based mining systems, this *does not* require modifications to the underlying solvers; 2) the search strategy of constraint programming systems turns out to parallel the search strategy of existing, specialized constraint-based mining approaches among which Eclat [16], LCM [15] and DualMiner [5]; 3) even though the constraint programming methods were not meant to cope with the specifics of data mining (such as coping with large data sets, having 10 000s of constraints to solve), and even though the focus of this study is not on the development of efficient algorithms, it turns out that existing constraint programming systems already perform quite well as compared to dedicated data mining solvers in that on a number of benchmark problems their performance is similar, and in some cases even better com-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD’08, August 24–27, 2008, Las Vegas, Nevada, USA.
Copyright 2008 ACM 978-1-60558-193-4/08/08 ...\$5.00.

¹<http://www.gecode.org/>

²<http://eclipse.crosscoreop.com/>

pared to state-of-the-art itemset miners [10]. At the same time, it should be clear that – in principle – the resulting constraint programming methods can be further optimized towards data mining.

This paper is organized as follows: in Section 2 we introduce a wide variety of constraints for itemset mining problems; in Section 3 we introduce the main principles of constraint programming systems; Section 4 then shows how the constraint-based mining problems can be formulated using constraint programming principles; Section 5 then compares the operation of constraint programming systems with those of dedicated itemset mining algorithms. Section 6 reports on an experimental evaluation comparing an off-the-shelf constraint programming system with state-of-the-art itemset mining implementations, and finally, Section 7 concludes.

2. ITEMSET MINING

Let $\mathcal{I} = \{1, \dots, m\}$ be a set of items, and $\mathcal{T} = \{1, \dots, n\}$ a set of transactions. Then an itemset database $\mathcal{D}$ is a binary matrix of size $n \times m$. Furthermore, $\varphi : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{T}}$ is a function that maps an itemset I to the set of transactions from $\mathcal{T}$ in which all its items occur, that is,

$$\varphi(I) = \{t \in \mathcal{T} \mid \forall i \in I : \mathcal{D}_{ti} = 1\}$$

Dually, $\psi : 2^{\mathcal{T}} \rightarrow 2^{\mathcal{I}}$ is a function that maps a transaction-set T to the set of all items from $\mathcal{I}$ shared by all transactions in T , that is,

$$\psi(T) = \{i \in \mathcal{I} \mid \forall t \in T : \mathcal{D}_{ti} = 1\}$$

It is well-known that φ and ψ define a Galois connection between the lattices $(\mathcal{T}, \subseteq)$ and $(\mathcal{I}, \subseteq)$. This means that the following properties are satisfied:

$$\begin{aligned} \forall T_1, T_2 \subseteq \mathcal{T} & : T_1 \subseteq T_2 \rightarrow \psi(T_2) \subseteq \psi(T_1) \\ \forall I_1, I_2 \subseteq \mathcal{I} & : I_1 \subseteq I_2 \rightarrow \varphi(I_2) \subseteq \varphi(I_1) \\ \forall I \subseteq \mathcal{I} & : \varphi(I) = \bigcap_{i \in I} \varphi(\{i\}) \\ \forall T \subseteq \mathcal{T} & : \psi(T) = \bigcap_{t \in T} \psi(\{t\}) \end{aligned}$$

In the remainder of this section, we will introduce several well-known constraints in itemset mining by making use of these operators. We will show that many itemset mining problems can be formulated as a search for pairs (I, T) , where I is an itemset and T a transaction set.

FREQUENT ITEMSETS. Our first example are the traditional frequent itemsets [1]. The search for these itemsets can be seen as searching for solution pairs (I, T) , such that

$$T = \varphi(I) \quad (1)$$

$$|T| \geq \theta \quad (2)$$

where θ is a frequency threshold. The first constraint specifies that T must equal the set of transactions in which I occurs; the next constraint is the well-known minimum frequency requirement: the absolute number of transactions in which I occurs must be at least θ . The properties of the φ operator imply that minimum frequency is an *anti-monotonic* constraint: every subset of an itemset that satisfies the frequency constraint, also satisfies the constraint.

ANTI-MONOTONIC CONSTRAINTS. Other examples of anti-monotonic constraints are maximum itemset size and maximum total itemset cost [5, 4]. Assume that every item

has a cost c_i (in the case of a size constraint, $c_i = 1$). Then we satisfy a maximum cost (size) constraint, for $c(I) = \sum_{i \in I} c_i$, if

$$c(I) \leq \gamma. \quad (3)$$

MONOTONIC CONSTRAINTS. The duals of anti-monotonic constraints are monotonic constraints. Maximum frequency, minimum size and minimum cost are examples of monotonic constraints [5, 4]. Maximum frequency can be formulated similar to (2) as:

$$|T| \leq \theta \quad (4)$$

while minimum cost is expressed similar to (3) by

$$c(I) \geq \gamma. \quad (5)$$

These constraints are called *monotonic* as every superset of an itemset that satisfies the constraints, also satisfies these constraints.

CONVERTIBLE ANTI-MONOTONIC CONSTRAINTS. Some constraints are neither monotonic nor anti-monotonic, but still have properties that can be exploited in mining algorithms. One such class of constraints are the convertible (anti-)monotonic constraints [12]. Let us illustrate these by the minimum average cost constraint, which can be specified as

$$c(I)/|I| \geq \gamma. \quad (6)$$

This constraint is called convertible anti-monotone as we can compute an order on the items in $\mathcal{I}$ such that for any itemset $I \subseteq \mathcal{I}$, every prefix I' of the items in I sorted in this order, also satisfies the constraint. In this case, we can order the items decreasing in cost: the average cost can only go up if we remove the item with the lowest cost.

CLOSED ITEMSETS. Closed Itemsets are a popular *condensed* representation for the set of all frequent itemsets and their frequencies [15]. Itemsets are called closed when they satisfy

$$I = \psi(T) \quad (7)$$

in addition to constraint (1); alternatively this can be formulated as $I = \psi(\varphi(I))$. A generalization are the δ -closed itemsets [7], which are itemsets that satisfy

$$\forall I' \supset I : |\varphi(I')| < (1 - \delta)|T|; \quad (8)$$

the traditional closed itemsets are a special case with $\delta = 0$.

MAXIMAL ITEMSETS. *Maximal* frequent itemsets are another condensed representation for the set of frequent itemsets [6]. In addition to the frequent itemset constraints (2) and (1) these itemsets also satisfy

$$\forall I' \supset I : |\varphi(I')| < \theta; \quad (9)$$

all itemsets that are a superset of a maximal itemset are infrequent, while all itemsets that are subsets are frequent. Maximal frequent itemsets constitute a *border* between itemsets that are frequent and not frequent.

EMERGING PATTERNS. If two databases are given, one can be interested in finding itemsets that distinguish these two databases. In other words, one is interested in finding triples $(I, T^{(1)}, T^{(2)})$, containing an itemset I and two transaction sets $T^{(1)}$ and $T^{(2)}$, such that some distinguishing property

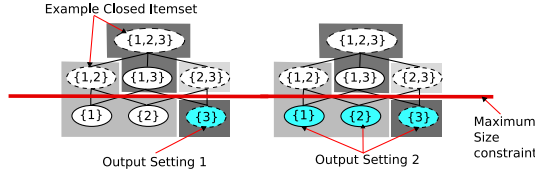


Figure 1: There are two combinations of closedness and maximum size

between $T^{(1)}$ and $T^{(2)}$ holds. Among the many ways to score a pattern’s ability to distinguish two datasets is the following constraint:

$$\begin{aligned} T^{(1)} &= \varphi_1(I) \\ T^{(2)} &= \varphi_2(I) \\ |T^{(1)}|/|T^{(2)}| &\geq \rho, \end{aligned}$$

for a given threshold ρ ; we assume that for every database we have separate φ and ψ operators. An itemset that satisfies these constraints is called *emerging* [8].

COMBINING CONSTRAINTS. As pointed out in the introduction, it can also be interesting to combine constraints. Defining combinations is not always straightforward when working with constraints such as maximality and closedness [3]. For example, assume we want to mine for δ -closed frequent itemsets that have a size lower than a threshold. Two interpretations are possible. We can define closedness with respect to the set of all frequent itemsets, which means we combine (1), (2), (3) and (8) into:

$$\begin{aligned} T &= \varphi(I) \\ |T| &\geq \theta \\ \forall I' \supset I : |\varphi(I')| &< (1 - \delta)|T| \\ c(I) &\leq \gamma \end{aligned}$$

The other interpretation is that we mine for the itemsets that are δ -closed within the set of small itemsets:

$$\begin{aligned} T &= \varphi(I) \\ |T| &\geq \theta \\ \forall I' \supset I : (|\varphi(I')| < (1 - \delta)|T| \vee c(I') > \gamma) \\ c(I) &\leq \gamma \end{aligned}$$

The difference between these two settings is illustrated in Figure 1 for $\delta = 0$ and $\gamma = 1$. Itemsets closed according to constraint (8) are dashed in this figure. In the first setting, only the itemset $\{3\}$ satisfies the constraints; in the second setting, the itemsets $\{1\}$ and $\{2\}$ are also closed considering the maximum size constraint.

Similarly, combinations of maximality and anti-monotonic constraints also have 2 interpretations. Further combinations can be obtained by combining them with emerging patterns. The challenge that we address in this paper, is how to solve such a broad range of queries and their combinations in a unified framework.

3. CONSTRAINT PROGRAMMING

Constraint programming is a declarative programming paradigm: instead of specifying how to solve a problem, the user only has to specify the problem itself. The constraint

Algorithm 1 Constraint-Search(D)

```

1:  $D := \text{propagate}(D)$ 
2: if  $D$  is a false domain then
3:   return
4: end if
5: if  $\exists x \in \mathcal{V} : |D(x)| > 1$  then
6:    $x := \arg \min_{x \in \mathcal{V}, |D(x)| > 1} f(x)$ 
7:   for all  $d \in D(x)$  do
8:     Constraint-Search( $D \cup \{x \mapsto \{d\}\}$ )
9:   end for
10: else
11:   Output solution
12: end if

```

programming system is then responsible for solving it. Constraint programming systems solve constraint satisfaction problems (CSP). A CSP $\mathcal{P} = (\mathcal{V}, D, \mathcal{C})$ is specified by

- a finite set of variables $\mathcal{V}$;
- an initial domain D , which maps every variable $v \in \mathcal{V}$ to a finite set of integers $D(v)$;
- a finite set of constraints $\mathcal{C}$.

A constraint $C(x_1, \dots, x_k) \in \mathcal{C}$ is a boolean function from variables $\{x_1, \dots, x_k\} \subseteq \mathcal{V}$. A constraint is called *unary* if it involves one variable and *binary* if it involves two. A domain D' is called *stronger* than the initial domain D if $D'(x) \subseteq D(x)$ for all $x \in \mathcal{V}$. A domain is *false* if there exists an $x \in \mathcal{V}$ such that $D(x) = \emptyset$; a variable $x \in \mathcal{V}$ is called *fixed* if $|D(x)| = 1$. A solution to a CSP is a domain D' that fixes all variables ($\forall x \in \mathcal{V} : |D'(x)| = 1$) and satisfies all constraints: abusing notation, we must have that $\forall C(x_1, \dots, x_k) \in \mathcal{C} : C(D'(x_1), \dots, D'(x_k)) = 1$; furthermore D' must be stronger than D , which guarantees that every variable has a value from its initial domain $D(x)$.

EXAMPLE 1. Assume we have four people that we want to allocate to 2 offices, and that every person has a list of other people that he does not want to share an office with. Furthermore, every person has identified rooms he does not want to occupy. We can represent an instance of this problem with four variables, which represent the persons, and inequality constraints, which encode the room-sharing constraints:

$$\begin{aligned} D(x_1) &= D(x_2) = D(x_3) = D(x_4) = \{1, 2\} \\ \mathcal{C} &= \{x_1 \neq 2, x_1 \neq x_2, x_3 \neq x_4\}. \end{aligned}$$

The simplest algorithm to solve CSPs enumerates all possible fixed domains, and evaluates all constraints on each of these domains; clearly this approach is inefficient. The outline of a general, more efficient Constraint Programming (CP) system is given in Algorithm 1 above [14]. Essentially, a CP system performs a depth-first search; in each node of the search tree the algorithm branches by assigning values to a variable that is unfixed (line 7). It backtracks when a violation of constraints is found (line 2). The search is further optimized by carefully choosing the variable that is fixed next (line 6); a function $f(x)$ ranks variables, for instance, by determining which variable is involved in most constraints.

The main concept used to speed-up the search is constraint propagation (line 1). Propagation reduces the do-

main of variables such that the domain remains *locally consistent*. One can formally define many types of local consistencies, but we skip these definitions here. In general, in a locally consistent problem a value d does not occur in the domain of a variable x if it can be determined that there is no solution D' in which $D'(x) = \{d\}$. The main motivation for maintaining local consistencies is to ensure that the backtracking search does not unnecessarily branch, thereby significantly speeding up the search.

To maintain local consistencies *propagators* or propagation rules are used. A propagator takes as input a domain and outputs a stronger, locally consistent domain. The propagation rules are derived by the system from the user specified constraints. A *checking propagator* is a propagator that produces a false domain once the original constraint is violated. Most propagators are checking propagators. The repeated application of propagators can lead to increasingly stronger domains. Propagation continues until a *fixed point* is reached in which the domain does not change any more (line 1). There are many different constraint programming systems, which differ in the type of constraints they support and the way they handle these constraints. Most systems assign priorities to constraints to ensure that propagators of lower computational complexity are evaluated first. The main challenge is to manipulate the propagators such that propagation is as cheap as possible.

EXAMPLE 2 (EXAMPLE 1 CONTINUED). *The initial domain of this problem is not consistent: the constraint $x_1 \neq 2$ cannot be satisfied when $D(x_1) = \{2\}$; consequently 2 is removed from $D(x_1)$. Subsequently, the binary constraint $x_1 \neq x_2$ cannot be satisfied while $x_2 = 1$. Therefore value 1 is removed from the domain of x_2 . The propagator for the constraint $x_1 \neq x_2$ has the following form:*

if $D(x_1) = \{d\}$ **then delete** d from $D(x_2)$.

After applying all propagators in our example, we obtain a fixed point in which $D(x_1) = \{1\}$ and $D(x_2) = \{2\}$, which means persons one and two have been allocated to an office. Two rooms are possible for person 3. The search branches therefore. For each of these branches, the second inequality constraint is propagated; a fixed point is then reached in which every variable is fixed, and a solution is found.

To formulate itemset mining problems as constraint programming models, we only use variables with binary domains, i.e. $D(x) = \{0, 1\}$ for all $x \in \mathcal{V}$. Furthermore, we make extensive use of two types of constraints. The first is a *summation* constraint, whose general form is as follows:

$$\sum_{x \in V} w_x x \geq \theta. \quad (10)$$

In this constraint, $V \subseteq \mathcal{V}$ is a set of variables and w_x is a weight for variable x and can be either positive or negative.

To make clear how this constraint can be propagated, we show a propagator here, such as implemented in most CP systems. Let us use the following notation: $x^{max} = \max_{d \in D(x)} d$ and $x^{min} = \min_{d \in D(x)} d$. Furthermore, $V^+ = \{x \in V | w_x \geq 0\}$ and $V^- = \{x \in V | w_x < 0\}$.

At any point during the search the following constraint must be satisfied in order for Equation (10) to be satisfied:

$$\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+} w_x x^{max} \geq \theta.$$

The correctness of this formula follows from the fact that the lefthand side of the equation denotes the highest value that the sum can still achieve.

A checking propagator derived from the constraint for a variable $x' \in V^+$ conceptually has the following effects:

```

1: if  $\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+} w_x x^{max} \geq \theta$  then
2:   if  $\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+ \setminus \{x'\}} w_x x^{max} < \theta$  then
3:      $D(x') = \{1\}$ 
4:   end if
5: else
6:    $D(x') = \emptyset$ 
7: end if

```

Only in line 3 an effective domain reduction takes place. A similar propagator can be derived for a variable $x' \in V^-$.

EXAMPLE 3. *Let us illustrate the application of the propagator for the summation constraint on this problem:*

$$x_1 + x_2 + x_3 \geq 2, \\ D(x_1) = \{1\}, D(x_2) = \{0, 1\}, D(x_3) = \{0, 1\};$$

In this case, we know that at least one of x_2 and x_3 must have the value 1, but we cannot conclude that either one of these variables is certainly zero or one. The propagator does not change any domains. On the other hand, if

$$x_1 + x_2 + x_3 \geq 3, \\ D(x_1) = \{1\}, D(x_2) = \{0, 1\}, D(x_3) = \{0, 1\};$$

the propagator determines that $D(x_2) = D(x_3) = \{1\}$.

Another special type of constraints that we will use are *reified constraints*:

$$C \leftrightarrow x,$$

where C is a constraint and x is a boolean variable. A reified constraint binds the value of one variable to the evaluation of a constraint. An example of such a constraint is

$$\sum_{x \in V} w_x x \geq \theta \leftrightarrow x', \quad (11)$$

which states that if the weighted sum of certain variables V is higher than θ , variable x' should be true, and vice-versa.

We can decompose a reified constraint in two directions:

$$C \rightarrow x \quad \text{and} \quad C \leftarrow x.$$

We show the propagation that can be performed for both directions, in the special case of Equation (11). For the direction $C \rightarrow x$, the propagation is:

```

1: if  $0 \in D(x)$  and  $\sum_{x \in V^-} w_x x^{max} + \sum_{x \in V^+} w_x x^{min} \geq \theta$ 
   then delete 0 from  $D(x)$ 
2: if  $D(x) = \{0\}$  then apply propagators for  $\neg C$ 

```

For the reverse direction the propagation is:

```

1: if  $1 \in D(x)$  and  $\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+} w_x x^{max} < \theta$ 
   then delete 1 from  $D(x)$ 
2: if  $D(x) = \{1\}$  then apply propagators for  $C$ 

```

It is important to note the difference between a summation constraint C and its reified version $x \rightarrow C$. As we can see from the code above, the propagator for $x \rightarrow C$ is not as

expensive to evaluate as the propagator for C as soon as $1 \notin D(x)$.

Even though the $C \leftrightarrow x$ constraint can be expressed in both ECLiPSe and Gecode, we found that the reified implications $C \leftarrow x$ and $C \rightarrow x$ are not available by default; in our implementations (see Section 6) we use additional variables to express one direction of reified constraints.

4. REFORMULATING CONSTRAINTS ON ITEMSETS

We now introduce the models of itemset mining problems that can be provided to constraint programming systems. We choose the following representation of itemsets and transactions. For every transaction we introduce a variable $T_t \in \{0, 1\}$, and for every item a variable $I_i \in \{0, 1\}$; thus, we can conceive an itemset I as a vector of length m with binary variables; a transaction set T is a vector of length n .

THEOREM 1 (FREQUENT ITEMSETS). *Frequent itemset mining is expressed by the following constraints:*

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} I_i(1 - \mathcal{D}_{ti}) = 0. \quad (12)$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} \geq \theta. \quad (13)$$

PROOF. The first constraint (12) is a reformulation of the coverage constraint (1):

$$\begin{aligned} T = \varphi(I) &= \{t \in \mathcal{T} \mid \forall i \in I : \mathcal{D}_{ti} = 1\} \\ \iff \forall t \in \mathcal{T} : t \in T \leftrightarrow \forall i \in I : \mathcal{D}_{ti} = 1 \\ \iff \forall t \in \mathcal{T} : t \in T \leftrightarrow \forall i \in I : 1 - \mathcal{D}_{ti} = 0. \\ \iff \forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} I_i(1 - \mathcal{D}_{ti}) = 0. \end{aligned}$$

The second constraint (13) is derived as follows. We can reformulate the frequency constraint as:

$$\sum_{t \in \mathcal{T}} T_t \geq \theta. \quad (14)$$

Together with the coverage constraint, this constraint defines the frequent itemset mining problem. As argued in the previous section, however, reified constraints can sometimes be desirable. To this purpose, we rewrite the frequency constraint further. First, we can observe that $\forall i \in I : |T| = |T \cap \varphi(\{i\})|$, as $T = \varphi(I) \subseteq \varphi(\{i\})$, and therefore that in a valid solution

$$\begin{aligned} \forall i \in I : |T \cap \varphi(\{i\})| &\geq \theta \\ \iff \forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} &\geq \theta. \end{aligned}$$

Please note that reification increases the number of constraints significantly; it will depend on the problem setting if this increase is still beneficial in the end. In the next section we will study how a constraint programming system operates in practice on these constraints. $\square$

Many other anti-monotonic constraints can also be specified in a straightforward way.

THEOREM 2 (ANTI-MONOTONIC CONSTRAINTS). *The maximum total cost constraint is expressed by:*

$$\sum_{i \in \mathcal{I}} c_i I_i \leq \gamma.$$

THEOREM 3 (MONOTONIC CONSTRAINTS). *A monotonic minimum cost constraint is specified by*

$$\sum_{i \in \mathcal{I}} c_i I_i \geq \gamma$$

or, equivalently, reified as

$$\forall t \in \mathcal{T} : T_t = 1 \rightarrow \sum_{i \in \mathcal{I}} c_i I_i \mathcal{D}_{ti} \geq \gamma. \quad (15)$$

PROOF. The reified version of the constraint exploits that $I \subseteq \psi(\{t\})$ for every $t \in \varphi(I)$; starting from (5):

$$\begin{aligned} \sum_{i \in I} c_i \geq \gamma &\iff \forall t \in T : \sum_{i \in (I \cap \psi(\{t\}))} c_i \geq \gamma \\ &\iff \forall t \in \mathcal{T} : T_t = 1 \rightarrow \sum_{i \in \mathcal{I}} c_i I_i \mathcal{D}_{ti} \geq \gamma. \end{aligned}$$

$\square$

THEOREM 4 (CONVERTIBLE CONSTRAINTS). *The convertible minimum average cost constraint is specified as follows:*

$$\sum_{i \in \mathcal{I}} (c_i - \gamma) I_i \geq 0$$

Equivalently, a reified version can be used similar to (15).

PROOF. This follows from rewriting constraint (6):

$$\sum_{i \in I} c_i / |I| \geq \gamma \iff \sum_{i \in \mathcal{I}} c_i I_i \geq \gamma \sum_{i \in \mathcal{I}} I_i \iff \sum_{i \in \mathcal{I}} (c_i - \gamma) I_i \geq 0.$$

$\square$

THEOREM 5 (CLOSED ITEMSETS). *The frequent closed itemset mining problem is specified by the conjunction of the coverage constraint (12), the frequency constraint (13) and*

$$\forall i \in \mathcal{I} : I_i = 1 \leftrightarrow \sum_{t \in \mathcal{T}} T_t(1 - \mathcal{D}_{ti}) = 0. \quad (16)$$

The more general δ -closed itemsets are specified by

$$\forall i \in \mathcal{I} : I_i = 1 \leftrightarrow \sum_{t \in \mathcal{T}} T_t(1 - \delta - \mathcal{D}_{ti}) \leq 0. \quad (17)$$

PROOF. Formulation (16) follows from the second Galois operator in (7) similar to the reformulation of (12) above. We skip the derivation for the δ -closed itemsets due to lack of space. $\square$

THEOREM 6 (MAXIMAL FREQUENT ITEMSET MINING). *The maximal frequent itemset mining problem is specified by the coverage constraint (12) and constraint*

$$\forall i \in \mathcal{I} : I_i = 1 \leftrightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} \geq \theta \quad (18)$$

PROOF. The maximality constraint is reformulated as follows:

$$\begin{aligned} \forall I' \supset I : |\varphi(I')| &< \theta \\ \iff \forall i \in \mathcal{I} - I : |\varphi(I \cup \{i\})| &< \theta \\ \iff \forall i \in \mathcal{I} - I : |T \cap \varphi(\{i\})| &< \theta \\ \iff \forall i \in \mathcal{I} : I_i = 0 \rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} &< \theta \end{aligned} \quad (19)$$

Together with the minimum frequency constraint (13) this becomes a two sided reified constraint. $\square$

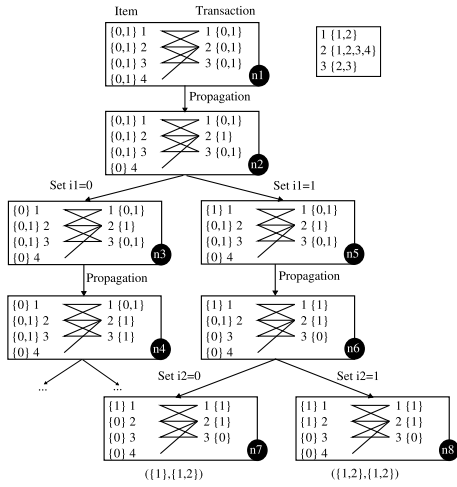


Figure 2: Search tree for the frequent itemset mining problem.

THEOREM 7 (EMERGING PATTERNS). *The problem of finding frequent emerging patterns is specified by:*

$$\forall k \in \{1, 2\} : \forall t \in \mathcal{T}^{(k)} : T^{(k)}t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} I_i (1 - D_{ti}^{(k)}) = 0.$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}_1} T_t D_{ti}^{(1)} \geq \theta.$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}^{(1)}} T_t^{(1)} D_{ti}^{(1)} - \rho \sum_{t \in \mathcal{T}^{(2)}} T_t^{(2)} D_{ti}^{(2)} \geq 0,$$

PROOF. This follows from the reification of

$$|T^{(1)}|/|T^{(2)}| \geq \rho \Leftrightarrow \sum_{t \in \mathcal{T}^{(1)}} T_t^{(1)} - \rho \sum_{t \in \mathcal{T}^{(2)}} T_t^{(2)} \geq 0;$$

furthermore, given that two datasets are given, coverage is expressed for both datasets. $\square$

5. CONSTRAINT PROGRAMMING SYSTEMS AS ITEMSET MINERS

We now investigate the behavior of constraint programming systems applied to itemset mining problems and compare this to standard constraint-based mining techniques.

Let us start with the *frequent itemset* mining problem. The search tree for an example database is illustrated in Figure 2. We use a minimum frequency threshold of $\theta = 2$. In the initial search node (n_1) the propagator for frequency constraint (13) sums for each item the number of transactions having that item, and determines that item 4 is only covered by 1 transaction, and therefore sets $D(I_4) = \{0\}$. The propagators for coverage constraint (12) determine that transaction 2 is covered by all items, and hence $D(T_2) = \{1\}$. This leads to the domain in node n_2 , where we have to branch. One of these branches leads to node n_5 , setting $D(I_1) = \{1\}$, thus including item 1 in the itemset. Coverage propagation sets transaction $D(T_3) = \{0\}$; frequency propagation determines that itemset $\{1, 3\}$ is not frequent and sets $D(I_3) = \{0\}$. Coverage propagation determines that transaction 1 is covered by all remaining items and sets $D(T_1) = \{1\}$. Propagation stops in node n_6 . Here both

possibilities in $D(I_2)$ are considered, but no further propagation is possible and we find the two frequent itemsets $\{1\}$ and $\{1, 2\}$.

In this example, the reified constraint is responsible for setting an item $D(I_i) = \{0\}$. Without reification the frequency constraint would never influence the domain of items. In our example, the system would continue branching below node n_7 to set $D(I_3) = \{1\}$, and only then find out that the resulting domain is false. By using the reified constraints, the system remembers which items were infrequent earlier, and will not try to add these deeper down the search tree. This example illustrates a key point: by using reified constraints, a CP system behaves very similar to other well-known depth-first itemset miners such as Eclat [16] and FP-Growth [11]. For a given itemset also these miners maintain which items can still be added to yield a frequent itemset (in FP-Growth, for instance, a projected database only contains such items). The transaction variables store a transaction set in a similar way as Eclat does: the variables that still have $1 \in D(T_i)$ represent transactions that are covered by an itemset during the search. A difference between well-known itemset miners and a CP system is that a CP system also maintains a set of transactions that are fixed to 1; furthermore, the CP system explicitly sets item variables to zero, while other systems usually do this implicitly by skipping them.

Let us consider the *maximal frequent itemset* mining problem next. Compared to the search tree of Figure 2, the search tree for maximal frequent itemset mining is different below node n_6 : applying the additional propagator for the maximality constraint (19), the CP system would now conclude that a sufficient number of transactions containing item 2 have been fixed to 1 and would remove 0 from the domain of item I_2 . As all variables are fixed, the search stops here and itemset $\{1, 2\}$ is found. This behavior is very similar to that of a well-known maximal frequent itemset miner: MAFIA [6] (and its generalization DualMiner [5]) uses the set of ‘unavoidable’ transactions (obtained by computing the supporting transactions of the Head-Union-Tail (HUT) itemset), and immediately adds all items if the HUT turns out to be frequent.

Likewise, we can consider *closed frequent itemset mining*: in node n_6 the CP system would conclude that all transactions with $D(T_i) = \{1\}$ support item 2, and would remove value 0 from $D(I_2)$ due to constraint (16); in general, this means that for every itemset, the items in its closure are computed; if an item is in the closure, but is already fixed to 0, the search backtracks, otherwise, the search continues with all items added. The same search strategy is employed by the well-known itemset miner LCM [15].

If we look at a *monotonic* minimum cost constraint, where we assume $c_1 = c_2 = c_4 = 1$ and $c_3 = 2$, and the cost threshold is at $\gamma = 3$, the search tree would differ already at node n_2 : if the reified constraint (15) is used, the items in transaction 1 are not expensive enough to exceed the cost threshold, and $D(T_1) = \{0\}$ is set. The effect is that item 1 does not have sufficient support any more, and is set to zero. This kind of pruning for monotonic constraints was called ExAnte pruning and was implemented in the ExaMiner [4].

Until now we did not specify how a CP system selects its next variable to fix (Algorithm 1, line 6). A careful choice can influence the efficiency of the search. This situation occurs when dealing with *convertible anti-monotonic*

	LCM [15]	MAFIA [6]	ExAMiner [4]	DualMiner [5]	CP
Constraints on data					
Minimum frequency	X	X	X	X	X
Maximum frequency				X	X
Emerging patterns					X
Condensed Representations					
Maximal	X	X		X	X
Closed	X	X			X
δ -Closed					X
Constraints on syntax					
Max/Min total cost			X	X	X
Minimum average cost			X		X
Max/Min size	X	X	X	X	X

Table 1: Comparison of Itemset Miners

constraints such as minimum average cost constraints. We can show that for a well-chosen order of variables, which reflects the cost constraint, the CP system will never encounter a false domain (similar to systems such as FP-Growth extended with convertible constraints [12]).

Summarizing our observations, it turns out that the search strategy employed by many itemset miners parallels that of standard CP systems applied to constraint-based mining problems. Furthermore, the CP approach is able to deal with *combinations* of constraints in a straight-forward manner. A comparison is given in Table 1³.

6. IMPLEMENTATION AND EXPERIMENTS

In this section we study the practical benefits and drawbacks of using state-of-the-art CP systems. In the first section, we consider this issue from a modeling perspective: how involved is it to specify an itemset mining task in CP systems? In the second section, we study the performance perspective, answering the question: how efficient are CP systems compared to existing itemset mining systems?

6.1 Modeling Efficiency

To illustrate how easy it is to specify itemset mining problems, we develop a model for standard frequent itemset mining in the ECLiPSe Constraint Programming System, a logic programming based solver with a declarative modeling language [2]. This model is given in Algorithm 2. It is almost identical to the formal notation.

This model generates all frequent itemsets, given a 2-dimensional array D , minimum frequency $Freq$, and predicate $prodlist(In1, In2, Out)$, which returns a list where each element is the multiplication of the corresponding elements in the input lists. Next to procedures for reading data all functionality is available by default in ECLiPSe.

Models for other constraints, and combinations of them, can be created in a similar way. For example, if we only want maximal itemsets, we replace line 10 by $I \# = (\text{sum}(PList) \# \geq Freq)$. If we only want itemsets of size at least 2, then we can add before line 6: $\text{sum}(Items) \# \geq 2$.

A similar model can be specified using the Gecode CP library [14]. Compared to the development of specialized algorithms, significantly less effort is needed to model the problem in CP systems

³These results are based on the parameters of the most recent implementations of the original authors, or, if not publicly available, on the original paper of these authors.

Algorithm 2 Frequent Itemset Mining in ECLiPSe

```

% Input: D: the data matrix; Freq: frequency threshold
% Output: Items: an itemset; Trans: a transaction set
1. fim_clp(D, Freq, Items, Trans) :-
2.   dim(D, [NrI, NrT]),
3.   length(Items, NrI),           % decision variables
4.   length(Trans, NrT),
5.   Items :: [0..1], Trans :: [0..1],
6.   ( foreach(I, Items), count(K, 1, _),           % model
7.     param(Trans, NrT, D, Freq)
8.   do Col is D[1..NrT, K],
9.     prodlist(Trans, Col, PList),
10.    I => (sum(PList) #>= Freq)
           %  $\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} T_t D_{ti} \geq \theta$ 
11. ),
12. ( foreach(T, Trans), count(K, 1, _),
13.   param(Items, NrI, D)
14. do Row is D[K, 1..NrI],
15.   prodlist_compl(Items, Row, PList_c),
16.   T # = (sum(PList_c) # = 0)
           %  $\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} I_i (1 - D_{ti}) = 0$ 
17. ),
18. labeling(Items).

```

6.2 Computational Efficiency

In these experiments we only use the Gecode constraint programming system [14]; we found that ECLiPSe was unable to handle the amounts of constraints that are needed to cope with larger datasets. We implemented models for frequent itemset mining, closed itemset mining, maximal itemset mining, and combinations of frequency, cost and size constraints. We compare with the most recent implementations of LCM and Mafia from the FIMI repository [10]; furthermore, we obtained the PATTERNIST system, which implements the ExAnte property [4]. All these systems are among the most efficient systems available. We use data from the UCI repository⁴. Properties of the data are listed in Table 2. To deal with missing values we preprocessed each dataset in the same way as [9]: we first eliminated all attributes having more than 10% of missing values and then removed all examples (transactions) for which the remaining attributes still had missing values. Numerical attributes were binarized by using unsupervised discretization with 4 bins. Where applicable, we generated costs per items randomly using a uniform distribution between 0 and 200. Experiments were run on PCs with Intel Core 2 Duo E6600 processors and 4GB of RAM, running Ubuntu Linux. The code of our implementation and the datasets used are available on our website⁵.

The density of a dataset is calculated by dividing the total number of 1s in the binary matrix by the size of this matrix. We encountered scalability problems for the large datasets in the FIMI challenge, and decided to restrict ourselves to smaller, but dense UCI datasets. We restrict ourselves to dense datasets as these are usually more difficult to mine. The numbers of frequent itemsets in Table 2 for a support threshold of 1% are given as an indication, and were computed using LCM. All our experiments were timed out after 30 minutes. The experiments indicate that additional constraints are needed to mine itemsets for low support values on the Segment data.

⁴<http://archive.ics.uci.edu/ml/>

⁵<http://www.cs.kuleuven.be/~dtai/CP4IM/>

	#Trans.	#Items	Density	#Patterns 1%
German Credit	1000	77	0.28	29 088 485
Letter	20000	74	0.33	1 037 221 530
Segment	2310	74	0.51	(time out)

Table 2: Description of the datasets

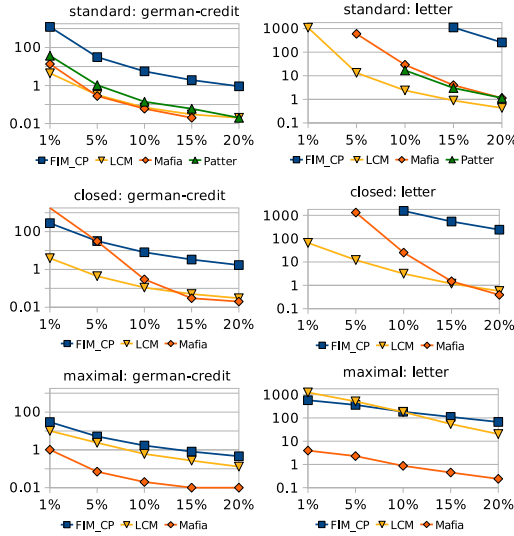


Figure 3: Runtimes of itemset miners on standard problems for different values of minimum support

Experiment 1: Standard Itemset Mining.

In our first experiment, we evaluate how the runtimes of our Gecode-based solver, denoted by *FIM_CP*, compare with those of state-of-the-art itemset miners, for the problems of frequent, maximal and closed itemset mining.

Results for two datasets are given in Figure 3. The experiments show that in most of the standard settings, which require the computation of large numbers of itemsets, *FIM_CP* is at the moment not very competitive. On the other hand, the experiments also show that the CP solver propagates all constraints as expected; for instance, maximal itemset mining is more efficient than frequent itemset mining. The system behaves very similar to other (specialized) systems from this perspective. In particular, if we compare *FIM_CP* with LCM, we see that *FIM_CP* sometimes performs better than LCM as a maximal frequent itemset miner for low support thresholds; similarly *FIM_CP* sometimes outperforms Mafia as closed itemset miner. This is an indication that further improvements in the implementations of CP solvers could lead to systems that perform satisfactory for most users.

Experiment 2: Standard Constraint-Based Mining.

In this experiment we determine how *FIM_CP* compares with other systems when additional constraints are employed.

Results for two settings are given in Figure 4. In the first experiment we employed a (monotonic) minimum size constraint in addition to a minimum frequency constraint; in the second a (convertible) maximum average cost constraint. The results are positive: even though for small minimum size constraints the brute force mining algorithms, such as LCM, outperform *FIM_CP*, *FIM_CP* does search very effectively when this constraint selects a small number of very

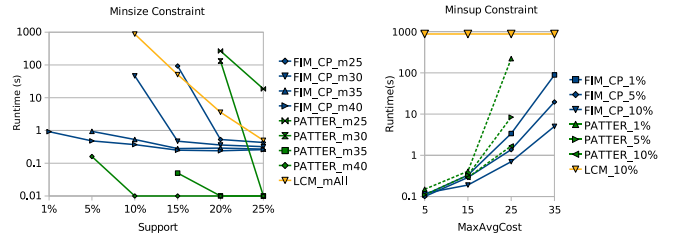


Figure 4: Runtimes of itemset miners on Segment data under constraints

large itemsets (30 items or more); in extreme cases *FIM_CP* finishes within seconds while other algorithms do not finish within our cut-off time of 30 minutes. PATTERNIST was unable to finish some of these experiments due to memory problems. This indicates that *FIM_CP* is a competitive solver when the constraints require the discovery of a small number of very large itemsets. The results for convertible constraint are particularly interesting, as we did not optimize the item order in any of our experiments.

Experiment 3: Novel Constraint-Based Mining.

In our final experiments we explore the effects of combining several types of constraints.

Our problem setting is related to the problem of finding itemsets that are predictive for one partition (or class) in the data; one way to find such itemsets is to look for itemsets that have high support in this partition and have low support in the remaining examples. Furthermore, it is desirable to condense the itemsets found; we investigate the use of δ -closedness; δ -closedness has not been studied in combination with other constraints in the literature. Finally, to reduce the complexity of the individual patterns found, we investigate the use of size constraints on the itemsets; we consider both minimum and maximum size constraints.

In our experiments, we divided the Segment dataset randomly in 2 partitions. As default parameters for the constraints we chose: a minimum support of 0.5%, a maximum support of 0.5%, a minimum size of 14, a maximum size of 16 and $\delta = 0.20$. Results are summarized in Table 3.

The issues that we study in this table are the following. First, as discussed in Section 2 and illustrated in Figure 1, there are two ways of combining maximum size and δ -closedness, referred to as Setting 1 and Setting 2. We are interested in the difference in size in the resulting sets of itemsets. In practice it turns out that Setting 1 yields significantly less itemsets than the second setting. Second, we study the influence of the minimum size constraint. The experiments show that the CP system achieves significantly lower runtimes when this additional constraint is applied, thus pushing the constraints effectively, while finding fewer patterns. Third, we study the influence of the δ -parameter. We mined for several values of δ , as can be seen in Figure 5, without size constraints. The results show for higher values of δ less patterns are returned and the algorithm runs faster.

Please note that the runtimes of the CP system are much lower than those obtained by LCM for the same support threshold; the CP approach is more efficient than running LCM and post-processing its results.

Constraint on $ I $	Setting 1		Setting 2	
	# Patterns	Time (s)	# Patterns	Time (s)
None	27860	30.27	27860	30.27
$ I \leq 16$	27756	30.20	39017	34.27
$14 \leq I $	4507	14.61	4507	14.72
$14 \leq I \leq 16$	4403	14.51	15664	19.25

Table 3: Applying size constraints on Segment data

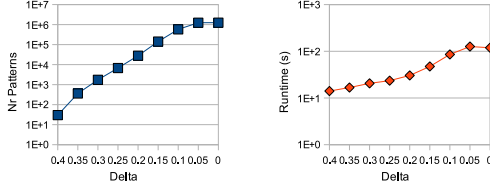


Figure 5: Mining Segment under δ -closedness

7. CONCLUSIONS

We have reformulated itemset mining problems in terms of constraint programming. This has allowed us to implement a novel mining system using standard constraint programming tools. This approach has several benefits. At a conceptual level, constraint programming offers a more uniform, extendible and declarative framework for a wide range of itemset mining problems than state-of-the-art data mining systems. At an algorithmic level, the general purpose constraint programming methods often emulate well-known itemset mining systems. Finally, from an experimental point of view, the results of the constraint programming implementation are encouraging for constraints that select many short itemsets, and are competitive or better for constraints that select few long itemsets. This despite the fact that constraint programming systems are general purpose solvers and were not developed with the large number of constraints needed for data mining in mind.

The main advantage of our method is its generality. The framework can be used to explore new constraints and combinations of constraints much more easily than is currently possible. In our approach, it is no longer necessary to develop new algorithms from scratch to deal with new types of constraints.

There are several open questions for further research. 1) How can constraint-programming algorithms be specialized and optimized for use in data mining? 2) Which other types of constraints can be used for mining with the constraint programming approach? and 3) Can the introduced framework be extended for supporting the mining of structured data, such as sequences, trees and graphs?

To summarize, we have contributed a first step towards bridging the gap between data mining and constraint programming, and have formulated a number of open questions for future research, both on the constraint programming and on the data mining side.

Acknowledgements. Siegfried Nijssen was supported by the EU FET IST project “Inductive Querying”, contract number FP6-516169. Tias Guns was supported by the Institute for the Promotion and Innovation through Science and Technology in Flanders (IWT-Vlaanderen). We are grateful to Francesco Bonchi for providing the PATTERNIST system, and to Albrecht Zimmermann and Élisabeth Fromont for discussions.

8. REFERENCES

- [1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.
- [2] K. R. Apt and M. Wallace. *Constraint Logic Programming using Eclipse*. Cambridge University Press, New York, NY, USA, 2007.
- [3] F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *ICDM*, pages 35–42. IEEE Computer Society, 2004.
- [4] F. Bonchi and C. Lucchese. Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.*, 60(2):377–399, 2007.
- [5] C. Bucila, J. Gehrke, D. Kifer, and W. M. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Min. Knowl. Discov.*, 7(3):241–272, 2003.
- [6] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu. Mafia: A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504, 2005.
- [7] J. Cheng, Y. Ke, and W. Ng. δ -tolerance closed frequent itemsets. In *ICDM ’06: Proceedings of the Sixth International Conference on Data Mining*, pages 139–148, 2006.
- [8] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *KDD*, pages 43–52, 1999.
- [9] E. Frank and I. H. Witten. Using a permutation test for attribute selection in decision trees. In *Proc. 15th International Conf. on Machine Learning*, pages 152–160, 1998.
- [10] B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on FIMI’03. In *SIGKDD Explorations Newsletter*, volume 6, pages 109–117, 2004.
- [11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2000.
- [12] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 433–442, 2001.
- [13] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., 2006.
- [14] C. Schulte and P. J. Stuckey. Efficient constraint propagation engines. *Transactions on Programming Languages and Systems*, 2008. To appear.
- [15] T. Uno, M. Kiyomi, and H. Arimura. Lcm ver.3: collaboration of array, bitmap and prefix tree for frequent itemset mining. In *OSDM ’05: Proceedings of the 1st international workshop on open source data mining*, pages 77–86, 2005.
- [16] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *KDD*, pages 283–286, 1997.

Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining

Arianna Gallo*

Pauli Miettinen†

Heikki Mannila‡

Abstract

Given a 0-1 dataset, we consider the redescription mining task introduced by Ramakrishnan, Parida, and Zaki. The problem is to find subsets of the rows that can be (approximately) defined by at least two different Boolean formulae on the attributes. That is, we search for pairs (α, β) of Boolean formulae such that the implications $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ both hold with high accuracy. We require that the two descriptions α and β are syntactically sufficiently different. Such pairs of descriptions indicate that the subset has different definitions, a fact that gives useful information about the data. We give simple algorithms for this task, and evaluate their performance. The methods are based on pruning the search space of all possible pairs of formulae by different accuracy criteria. The significance of the findings is tested by using randomization methods. Experimental results on simulated and real data show that the methods work well: on simulated data they find the planted subsets, and on real data they produce small and understandable results.

1 Introduction

Discovering interesting subgroups is one of the key concepts in data mining. Subgroups can be discovered by clustering, where the dataset is typically partitioned into disjoint sets, or by pattern discovery, where each pattern defines the subgroup in which the pattern is true, and thus the patterns can be overlapping. The interestingness of a subgroup is typically measured by how likely or unlikely such a subgroup would be to arise at random.

In general, a subgroup is defined either by the explicit values of the variables of the data (subgroups defined by patterns) or by distances between rows (clusters). Here we consider only subgroups defined by using formulae on the values of the variables. Any predicate $\alpha(t)$ defined for the rows t of the dataset D defines a subgroup $\{t \in D \mid \alpha(t) \text{ is true}\}$. For example, if the data is 0-1, then the formula $A = 1 \wedge (B = 0 \vee C = 1)$ defines a subgroup.

In this paper we consider the redescription mining task introduced by Ramakrishnan, Parida, and Zaki [16, 22, 15] that is the task of finding subgroups having several descriptions. That is, we want to find formulae α and β such that the sets $\{t \in D \mid \alpha(t) \text{ is true}\}$ and $\{t \in D \mid \beta(t) \text{ is true}\}$ are about the same. If α and β are logically equivalent, this holds trivially, but we are interested in finding formulae that are not equivalent, but still happen to be satisfied by about the same rows of D . (One might call such formulae D -equivalent.)

Another way of looking at the task is that we search for formulae α and β such that the rules $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ both hold with high accuracy.

Consider the following two transactional datasets, given here in matrix form, where 1 means the presence and 0 the absence of an item:

tid	A	B	C	D	E
1	1	0	1	0	0
2	0	0	1	0	0
3	1	1	0	0	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	1	0

tid	F	G	H	I
1	0	0	1	0
2	1	1	0	1
3	0	1	1	0
4	1	1	0	0
5	1	0	1	0
6	1	0	0	1

The datasets have the same transaction (row) identifiers but different sets of attributes. They might arise from two different studies on the same entities.

Note that the subgroup $\{1, 3, 4\}$ is defined by the formula $A = 1 \wedge (B = 0 \vee E = 1)$ in the first table. The formula $F = 0 \vee (G = 1 \wedge I = 0)$, using the attributes of the second table, is true for exactly the same set $\{1, 3, 4\}$ of rows. We say that the subset $\{1, 3, 4\}$ has *several descriptions* and that the formulae are *redescriptions* of each other.

As another example, consider the following table:

*University of Torino, Italy. Part of this work was done when the author was with Helsinki Institute for Information Technology. Email: gallo@di.unito.it

†Helsinki Institute for Information Technology, University of Helsinki, Finland. Email: Pauli.Miettinen@cs.helsinki.fi

‡Helsinki Institute for Information Technology, Helsinki University of Technology and University of Helsinki, Finland. Email: Heikki.Mannila@cs.helsinki.fi

tid	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	0	1	0	1	1	1
4	1	0	0	0	0	1
5	1	1	0	0	1	1
6	0	1	0	1	0	0
7	1	1	1	1	1	1
8	1	0	1	0	0	0
9	1	1	1	1	1	0

Here, the subset $\{1, 4, 5, 8\}$ of rows can, for example, be defined in three different ways:

$$\begin{aligned}
A &= 1 \wedge (B = 0 \vee D = 0) \\
B &= 0 \vee (A = 1 \wedge C = 0) \quad \text{and} \\
(C &= 1 \wedge E = 0) \vee (D = 0 \wedge F = 1).
\end{aligned}$$

Why search for groups with several descriptions? The fact that in D the set of rows satisfying α and β are about the same tells us something interesting about the data, provided α and β are not logically equivalent. If, for example, the set of persons whose genetic marker data satisfies a Boolean query for certain markers happens to coincide more or less exactly with the persons whose marker data satisfies another query for a completely different markers, then there is some indication that the regions of the genome might somehow be functionally connected.

In this paper we present algorithms for finding subgroups with several (typically two) descriptions. That is, we give methods that will produce pairs of formulae that are almost equivalent on the given dataset D . The methods are based on a search in the space of possible formulae; different pruning strategies are used to avoid useless paths in that space. Our goal is to produce a *small* set of pairs of formulae, where both formulae in each pair define about the same subset of rows. We define several notions of strength and interestingness for the pairs of formulae, and show how these measures can be used in pruning the search space.

Our algorithms are based on heuristic methods and our emphasis is on finding approximate redescription of arbitrary Boolean formulae. Thus our algorithms work on a different basis than many previously-proposed algorithms for redescription mining (e.g. [22, 15]), that usually concentrate on finding exact redescription of special type (e.g., monotone DNFs or only conjunctions).

If we allow arbitrarily complex formulae, then any dataset will have lots of subgroups with multiple descriptions. Thus we have to control for spurious results arising just by chance. We do this by using randomization methods, in particular swap randomization [8]. We generate a large set of randomized versions of the

input data, and run our algorithms on them. Only if the results on the real data are stronger than on, say, 99% of the randomized datasets we report a subgroup.

The rest of this paper is organized as follows. We introduce the notation and other preliminaries and define the problems in Section 2. Section 3 explains two algorithms and results from experiments with these algorithms are reported in Section 4. Some of the related work is covered in Section 5 and Section 6 contains concluding remarks.

2 The Problems

This section contains the notation used, describes the terminology used in redescription mining, and gives our definitions for the redescription mining problems. Our definitions differ slightly from the previous definitions [16, 22, 15], most notably by introducing thresholds for support and p -value of a redescription.

2.1 Notation. Given a set U of items, a transaction t is a pair (tid, X) , where tid is a transaction identifier and $X \subseteq U$ is the set of items of the transaction. A dataset D is a set of transactions: $D = \{t_1, \dots, t_n\}$. Multiple datasets are distinguished via a subscript, i.e., D_1 and D_2 are different datasets.

We consider multiple datasets $D_1, D_2, \dots$ over itemsets $U_1, U_2, \dots$. We assume that the datasets have an equal number of transactions and that the sets of their transaction ids are identical. We use n to denote the number of transactions in each dataset. The itemsets U_i and U_j , on the other hand, are supposed to be disjoint for all $i \neq j$.

We identify items as variables in Boolean queries. We let $\Gamma(U)$ denote the set of all possible Boolean formulae over the variables corresponding to items in U . Such formulae are called *descriptions*. Set $\Gamma_k(U) \subseteq \Gamma(U)$ contains all queries that have at most k variables. If dataset D is over itemset U , then by $\Gamma(D)$ we mean $\Gamma(U)$. Given a query α , we denote by $I(\alpha) \subseteq U$ the set of items appearing in α .

A set I of items can also be viewed as a truth assignment assigning the value true to the items in I and false to all other items. We say that $\alpha(I)$ is true if α is satisfied by a truth assignment corresponding to I . If $t_j = (\text{tid}_j, I_j)$, we say that $\alpha(t_j)$ is true if $\alpha(I_j)$ is true. We let $\alpha(D)$ to be the set of transaction ids from D for those transactions which satisfy α , that is, $\alpha(D) = \{\text{tid}_j : (\text{tid}_j, I_j) \in D \text{ and } \alpha(I_j) \text{ is true}\}$. The frequency $\text{freq}(\alpha)$ is defined as $|\alpha(D)|$. Furthermore, if $\alpha_i \in \Gamma(D_i)$, then by $\text{freq}(\alpha_1, \alpha_2, \dots, \alpha_n)$ we mean $|\bigcap_{i=1}^n \alpha_i(D_i)|$, i.e., the size of the set of all transaction ids such that those transactions satisfy each of the formulae α_i .

2.2 Jaccard similarity and redescrptions. If X and Y are two sets, the Jaccard similarity [7] between them is defined to be

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}.$$

It is a commonly used measure of the similarity of X and Y : if $X = Y$, $J(X, Y) = 1$ and if $X \cap Y = \emptyset$, $J(X, Y) = 0$.

If $\alpha \in \Gamma(D_1)$ and $\beta \in \Gamma(D_2)$ are Boolean formulae over the items in two datasets, the Jaccard similarity for the queries is defined as

$$J(\alpha, \beta) = J(\alpha(D_1), \beta(D_2)).$$

As the transaction ids of D_1 are identical to those of D_2 , the value $J(\alpha(D_1), \beta(D_2))$ lies in the interval between $[0, 1]$. If $J(\alpha(D_1), \beta(D_2)) = 1$, α and β are called *exact redescrptions* of each other, and if $J(\alpha(D_1), \beta(D_2)) \geq J_{\min}$ for some $J_{\min} \in (0, 1)$, they are called *approximate redescrptions* (with respect to $J_{\min}$). In the following, we refer to tuples of Boolean formula simply as redescrptions.

For multiple databases D_i and queries $\alpha_i \in \Gamma(D_i)$ for $i = 1, \dots, m$, we define

$$\begin{aligned} J(\alpha_1, \dots, \alpha_m) &= J(\alpha_1(D_1), \dots, \alpha_m(D_m)) \\ &= \frac{|\bigcap_{i=1}^m \alpha_i(D_i)|}{|\bigcup_{i=1}^m \alpha_i(D_i)|}. \end{aligned}$$

2.3 The p -values. Two queries α and β can be true for exactly the same set of transactions, but the pair (α, β) can still be uninteresting for our purposes. The first case in which this happens is when α and β are logically equivalent; we avoid this possibility by requiring that the set of variables (items) used in α and β are disjoint.

The second case is when the equality of the queries is a straightforward consequence of the marginal probabilities of the items. Suppose for example that two items A and B are almost always 0. Then the queries $A = 0$ and $B = 0$ have high support, their Jaccard similarity is large, but there is no new information in the pair.

To avoid this type of results we prune our result set using p -values. For calculating the p -value we consider a *null model* for the database, representing the “uninteresting” situation in which the items occur independently from each other in the database. The p -value of an itemset I represents the surprisingness of I under this null model. The smaller the p -value, the more “significant” the itemset is considered to be. More details on the different ways for computing the p -value of a given pattern are given in [4] and [5]; we summarize

the approach here briefly. See [19, 20, 10] for other approaches.

Let p_i be the marginal probability of an item i , i.e., the fraction of rows in the database that contain item i . Then the probability that an itemset I is contained in a transaction is $p_I = \prod_{i \in I} p_i$. The probability that an itemset I occurs in the dataset at least $\text{freq}(I)$ times is then obtained by using the binomial distribution:

$$(2.1) \quad p\text{-value}(I) = \sum_{s=\text{supp}(I)}^n \binom{n}{s} p_I^s (1 - p_I)^{n-s}.$$

Equation (2.1) encodes the situation where the items occurring in a transaction are independent from each other, and gives the “strength” of the surprisingness of an itemset I under this assumption.

Note that we could consider also other p -values. For instance, another p -value could measure the significance of a set of transactions containing a certain item, this time with the assumption that the transactions contain that item independently of each other. The p -value of an itemset can now be computed as the maximum of these two p -values [4].

As we already pointed out, an itemset I can be interpreted as a truth assignment assigning the value true to all the items in I and false to all other items. For instance, if the itemset I is defined as the set of items $\{A, B, C\}$, the rule $\alpha(I)$ corresponds to $A = 1 \wedge B = 1 \wedge C = 1$. Such formula is satisfied in a given transaction only if all the three literals are present in the transaction. For a certain conjunctive query α , the probability p_α that α is true for a transaction is computed under the assumption of independence of variables occurring in α . The probability can be defined recursively:

$$p_\alpha = \begin{cases} p_\beta p_\gamma & \text{if } \alpha = \beta \wedge \gamma \\ 1 - p_\beta & \text{if } \alpha = \neg \beta \\ p_\beta + p_\gamma - p_\beta p_\gamma & \text{if } \alpha = \beta \vee \gamma. \end{cases}$$

We can now generalize (2.1) for formulae:

$$(2.2) \quad p\text{-value}(\alpha) = \sum_{s=\text{freq}(\alpha)}^n \binom{n}{s} p_\alpha^s (1 - p_\alpha)^{n-s}.$$

2.4 Problem definitions. In this subsection we give the definition of redescription mining that is used throughout this paper. Our definition has minor differences to previous definitions [16, 22, 15].

We look for redescrptions that are true for a reasonable large set of transactions, whose Jaccard

similarity is high enough, and the p -value is small enough. We will also use a threshold for maximum frequency; the purpose of this threshold partly coincides with that of p -value, as it is used to prune results that are not significant because they cover most of the data. Apart the Jaccard similarity, these thresholds were not presented in previous definitions of redescription mining [16, 22, 15]. Thus, our definitions can be seen as generalizations of those definitions.

The first problem version is formulated for the case where there are several databases with the same set of transactions, and the second version is for the case of a single database. In the single database case, we search for queries that do not share items.

PROBLEM 2.1. *Given $m \geq 2$ datasets $D_1, D_2, \dots, D_m$, each containing the same set of transaction ids and disjoint set of items, and thresholds σ_{min} , σ_{max} , J_{min} , and P_{max} for minimum and maximum support, Jaccard similarity, and p -value, respectively. Find a set of m -tuples of Boolean formulae, $(\alpha_1, \alpha_2, \dots, \alpha_m) \in \Gamma(D_1) \times \Gamma(D_2) \times \dots \times \Gamma(D_m)$, so that*

$$(2.3) \quad freq(\alpha_1, \dots, \alpha_m)/n \in [\sigma_{min}, \sigma_{max}]$$

$$(2.4) \quad J(\alpha_1, \dots, \alpha_m) \geq J_{min}$$

$$(2.5) \quad p\text{-value}(\alpha_1, \dots, \alpha_m) \leq P_{max}.$$

A slightly different version of this problem arises when we are given only one dataset. In that case we search for Boolean formulae satisfied by approximately the same set of transactions, but being defined by using disjoint sets of items.

PROBLEM 2.2. *We are given a dataset D , an integer $m \geq 2$, and the thresholds as in Problem 2.1. The task is to find a set of m -tuples of Boolean formulae, $(\alpha_1, \dots, \alpha_m)$, $\alpha_i \in \Gamma(D)$, so that the conditions (2.3)–(2.5) hold and that for $i \neq j$*

$$(2.6) \quad \alpha_i(D) \cap \alpha_j(D) = \emptyset.$$

Notice that the condition (2.6) is implied in Problem 2.1 by the assumption of mutual disjointness of U_i 's.

3 Algorithms

In this section we present two algorithms for the redescription mining, namely the GREEDY algorithm and the MID algorithm. The algorithms are heuristic search methods that try to form useful pairs or tuples of queries.

3.1 The Greedy Algorithm. The GREEDY algorithm performs a level-wise search using a greedy heuristic to prune the search space. In addition to the threshold values, it also uses two other parameters to control

ALGORITHM 3.1 The GREEDY algorithm for Problem 2.1.

Input: Datasets $D_1, \dots, D_m$, thresholds σ_{min} , σ_{max} , J_{min} , and P_{max} and a maximum number of formula tuples N and a maximum number of variables per query K .
Output: A set $\mathcal{F}$ of m -tuples of Boolean queries with at most K variables.

```

1:  $\mathcal{F} \leftarrow \{(\alpha_1, \dots, \alpha_m) \in \Gamma_1(D_1) \times \dots \times \Gamma_1(D_m) : (\alpha_1, \dots, \alpha_m) \text{ satisfies (2.3)–(2.5)}\}$ 
2: Order  $\mathcal{F}$  using  $J(\alpha_1, \dots, \alpha_m)$  and keep top  $N$  redescrptions.
3: for  $(\alpha_1, \dots, \alpha_m) \in \mathcal{F}$  do
4:    $V_i \leftarrow U_i \setminus I(\alpha_i), i = 1, \dots, m$ 
5:   for  $i \leftarrow 1, \dots, m$  do
6:      $W_i \leftarrow V_i$ 
7:     for  $i \in V_i$  do
8:       Try to expand  $\alpha_i$  to  $\alpha'_i$  using  $i$  with  $\wedge$ ,  $\vee$ , and  $\neg$ 
9:       if  $J(\alpha_1, \dots, \alpha'_i, \dots, \alpha_m) \geq J(\alpha_1, \dots, \alpha_i, \dots, \alpha_m)$  and
          $freq(\alpha_1, \dots, \alpha'_i, \dots, \alpha_m)/n \in [\sigma_{min}, \sigma_{max}]$  then
10:         $W_i \leftarrow W_i \setminus \{i\}$  and save  $\alpha'_i$ 
11:      Select the  $\alpha'_i$  with highest  $J(\alpha_1, \dots, \alpha'_i, \dots, \alpha_m)$ 
12:       $\alpha_i \leftarrow \alpha'_i$  and  $V_i \leftarrow V_i \setminus (W_i \cup I(\alpha'_i))$ 
13:      if  $\max\{I(\alpha_i)\} < K$  and there is  $V_i \neq \emptyset$  then
14:        goto line 5
15: return  $\mathcal{F}$ 

```

the result, namely the maximum number of initial redescrptions to process and the maximum number of variables per Boolean formula. The pseudo-code of the algorithm is given as Algorithm 3.1.

The GREEDY algorithm works as follows. First (in lines 1–2) it finds the initial redescrptions, that is, tuples of formulae with each formula having only one variable. This is done via an exhaustive search. The redescrptions are ordered according to their Jaccard similarity, and only the best ones are considered in the remaining of the algorithm.

In the next step, the algorithm considers each initial redescription separately. In each redescription, it starts by considering formula α_1 which it tries to expand using the items from set V_1 (line 8). So if $\alpha_1 = (A = 0)$, the algorithm creates formulae $(A = 0 \vee B = 1)$, $(A = 0 \wedge B = 1)$, $(A = 0 \vee B = 0)$, and $(A = 0 \wedge B = 0)$. If any of the created formulae does not decrease the Jaccard similarity of the redescription and keeps the support within the limits, then the item i is removed from the set W_1 of “useless” items and the formula is saved with the new Jaccard similarity. When all items and formulae are tried, the algorithm selects the one that improves the Jaccard similarity most and replaces the old formula with that. It also removes from the set V_1 of available items the item added to α_1 and the items from set W_1 , i.e., the items that only decreased the Jaccard similarity (lines 11–12). Algorithm then proceeds to formula α_2 , and so on.

When all formulae are considered, algorithm checks if the maximum number of variables per formula is reached. If not, and if there still are items that can be added to the formulae, algorithm starts again from formula α_1 . Otherwise algorithm returns all redescrptions

that are in $\mathcal{F}$. If wanted, the redescrptions in $\mathcal{F}$ can once more be pruned according to the $P_{\max}$ threshold.

The way the GREEDY algorithm prunes its search space can lead to sub-optimal results in many ways. First, of course, the algorithm can prune initial pairs that could be used to create good descriptions. Second, the algorithm prunes the level-wise search tree in a way that can yield to pruning of items that could be used later. The pruning is similar to that of many frequent itemset mining algorithm's (e.g., APRIORI) but, unlike in frequent itemset mining, the anti-monotonicity property does not hold here because of the disjunctions.

The time complexity of the algorithm depends heavily on the efficiency of the pruning phase. If K , the maximum number of variables per formula, is not limited, then the algorithm can take exponential running time with respect to the maximum number of items in datasets. Another variable that has exponential effect to the running time of GREEDY is the number of datasets, m . Assuming all datasets have equal number k of items, just creating the initial pairs takes $O(2^m n^m)$ time. However, in practice m is usually small and does not cause serious slowdown of the algorithm.

If m is very small, say $m = 2$, a simple improvement is applicable to the GREEDY algorithm: instead of only iterating over formulae α_i in fixed order, try all permutations of the ordering and select the best. We employ this improvement in our experiments, where m is always 2. With larger values of m the number of permutations, of course, becomes quickly infeasible.

Modifications for Single Dataset. We can modify the GREEDY algorithm to work with the setting of Problem 2.2, i.e., when only one dataset D is given. First, the initial m -tuples are computed over the $2^m \binom{|U|}{m}$ possible Boolean formulae in $\Gamma_1(D)$. The sets V_i and W_i in Algorithm 3.1 are replaced with two sets, V and W , which are used with all formulae. In that way the algorithm ensures that the formula tuples satisfy the disjointness condition (2.6). Also in this case, if m is small, we can, and in the experiments will, apply the aforementioned improvement of iterating over all permutations of the ordering of the formulae.

3.2 The MID Algorithm. The GREEDY algorithm starts from singleton queries, i.e., queries with only one variable, and iteratively adds new variables using Boolean operators.

The approach described in this section starts instead from a set of itemsets, i.e., a set of positive conjunctive queries, and iteratively adds new itemsets using Boolean operators. At each step, the p -value of the query is used for pruning non-interesting queries. Since in real-world tasks the number of all the possible item-

sets is often massive, we only extract a condensed and lossless representation of them, i.e., a set of closed itemsets [14]. A closed itemset is an itemset with no frequent superset having same support. We can focus only on closed itemsets since for any non-closed itemset there exists a closed itemset with lower p -value [4].

In this section we describe MID, an algorithm for Mining Interesting Descriptors of subgroups. The sketch of the MID algorithm is given in Algorithm 3.2. The algorithm starts by mining the set of frequent closed itemsets from the given dataset(s) and sorting them by p -value (line 1). In this step we compute the p -value of an itemset I as the maximum of the two p -values computed under the assumptions of independence of items and transactions, as explained in Section 2.3.

The set R_i^0 contains the closed itemsets mined from the dataset D_i . For each query $\alpha \in R_i^0$, the query $(\neg\alpha)$ is added to R_i^0 if $p\text{-value}(\neg\alpha) < P_{\max}$ (lines 2–4). The p -values in line 3 are computed as in the previous step. Only the first N queries constitute the input for the next steps (line 5). At each step $j = 1, \dots, K-1$, (line 6) new queries are added to the set R_i^{j+1} of “surprising” queries using Boolean operators $\wedge$ and $\vee$. Such queries have p -value higher than the given threshold and are built by using $j+1$ closed itemsets. That is, at each step j , the set R_i^{j+1} is built in a level-wise fashion, starting from the set R_i^j (lines 6–13). In these steps, when a positive disjunctive query is extended by adding to its formula another itemset, the sharing of any items between the two itemsets is avoided. For this reason, the p -values in lines 9 and 11 are computed without taking into account the second p -value described in Section 2.3, i.e., the one that considers the transactions to be independent, but only that computed under the assumption that items are independent.

In the last step (lines 15–22), the subgroups having redescrptions are mined. If only one dataset is given as input, i.e. $m = 1$, the descriptions are computed using the queries (α_1, α_2) found in the previous steps (lines 20–22). Instead, m descriptions, $\alpha_1, \alpha_2, \dots, \alpha_{K-1}$, will be sought in lines 16–18 taking into account the set of queries R_i^{K-1} found in the previous steps from each dataset D_i , respectively.

The final result R is sorted by Jaccard similarity. However, this measure could suggest misleading results. Indeed, a set of queries with high Jaccard similarity is not necessarily an “interesting” result, as mentioned in Section 2.3. Thus, the MID algorithm uses both the Jaccard similarity and p -value to find accurate and interesting redescrptions.

Let us note that the result of the MID algorithm is slightly different to that of GREEDY. While MID tries to find the descriptions starting from itemsets, the

Table 1: The position of the planted redescription in the results ordered according to Jaccard similarity. “–” denotes that the algorithm was unable to find the redescription.

Algorithm	Noise level										
	0	0.02	0.04	0.06	0.08	0.1	0.12	0.14	0.16	0.18	0.2
GREEDY	1	1	1	1	–	1	3	1	–	67	–
MID	1	1	1	331	–	–	–	–	–	–	–

Table 2: The results with completely random data and varying $P_{\max}$.

Algorithm	$P_{\max}$				
	1	0.5	0.25	0.125	0.0625
GREEDY	158	64	23	5	1
MID	0	0	0	0	0

Table 3: The results with completely random data and varying $J_{\min}$.

Algorithm	$J_{\min}$				
	0.4	0.5	0.6	0.7	0.8
GREEDY	158	4	0	0	0
MID	0	0	0	0	0

performance can be attributed to MID’s stronger use of p -values in pruning the search space: as the noise level increases, the findings start more and more look like the ones obtained by a mere change.

The other synthetic data we used consisted of two matrices with fully random data. The idea of this data is to show that from a random data one cannot find any significant descriptions. The support thresholds for this experiment were set to $\sigma_{\min} = 20\%$ and $\sigma_{\max} = 80\%$. Thresholds $P_{\max}$ and $J_{\min}$ were varied: first, $J_{\min}$ was fixed to 0.4, and $P_{\max}$ varied from 1 to 0.0625, halving at every step; then $P_{\max}$ was fixed to 1 and $J_{\min}$ varied from 0.4 to 0.8. The goal of this experiment was not to find anything, and the MID algorithm performed perfectly in this sense: it never found any descriptions. The GREEDY algorithm did find some descriptions; its results are given in Tables 2 and 3.

Tables 2 and 3 show that while GREEDY does find descriptors with the minimum values of the thresholds, the number of descriptors it reports decreases rapidly when the thresholds are increased. The $J_{\min}$ threshold is especially effective, as there are no descriptions reported after $J_{\min} \geq 0.6$.

4.2 Real Data. We tested the methods on real data. The first test is whether the methods find intuitively interesting results from real data. That is, do the algorithms output something that passes the pruning criteria and seems interesting for a user who understands (at least part of) the application domain.

As this test can be somewhat subjective, we also wanted to verify that the results are not just due to the presence of sufficient amounts of data about phenomena that we are familiar with. To test this, we used randomization methods to check that on the real data there are clearly more pairs of formulae passing the pruning criteria than on randomized versions of the data. For the second type of test, we used swap randomization [3, 8], a method that generates matrices having the same row and column sums as the original 0-1 data set.

The Data. We used three real data sets, called **Courses**, **Web**, and **DBLP**. **Courses** data contains students’ course enrollment data of CS students at University of Helsinki. The data contains a single dataset of 106 courses (items) and 2401 students (transactions). Thus it is applicable for testing our algorithms in the framework of Problem 2.2.

The **Web** data¹ is used in [2] and is collected from web pages of US universities’ CS departments. It contains two datasets. The first dataset is about the terms appearing in the web pages. The transactions (rows) are the web pages (1051 pages) and the items are the terms occurring in the pages (1798 terms). The other dataset contains the terms from hyperlinks pointing to the web pages: the transactions are again the pages, and the items are the terms occurring in the hyperlinks (438 terms). The **Web** data thus gives two parallel views to a single web page: the terms used in the page and the terms used to describe the page in hyperlinks. All terms were stemmed and the most frequent and infrequent terms were removed.

The **DBLP** data was collected from the DBLP Bibliography database². It also contains two datasets. The

¹<http://www.cs.cmu.edu/afs/cs/project/theo-11/www/wkbb/>

²<http://www.informatik.uni-trier.de/~ley/db/>

transactions (rows) in the first dataset are authors, and the items in the first dataset are CS conferences: the presence of an item in a transaction means that the author has published in that conference. The other dataset indicates the co-authorship relations: transactions are again authors and as items we also have authors. The dataset is symmetric (i.e., the matrix representation of it is symmetric). As the whole dataset is huge, we selected only 19 conferences in our sample. The conferences were WWW, SIGMOD, VLDB, ICDE, KDD, SDM, PKDD, ICDM, EDBT, PODS, SODA, FOCIS, STOC, STACS, ICML, ECML, COLT, UAI, and ICDT. We then removed all authors that had published less than 5 times in these conferences, resulting in a dataset containing 2345 authors. Thus the dimensions of the two tables are 2345×19 and 2345×2345 . The properties of the real data are summarized in Table 4.

The Results. With the **Courses** data, the GREEDY algorithm reports a lot of descriptions. However, as discussed earlier, many of these results are in fact redundant, and can thus be pruned using the redundancy reduction method from Section 3.3. Another way to prune the results is to select the accuracy threshold $J_{\min}$ to a high enough value. Our experiments with swap randomized data suggest setting it to as high as 0.9, as with lower values of $J_{\min}$ the results are not found significant, but while there still are more than 50 redescrptions having Jaccard similarity at least 0.9, none of such redescrptions were found from randomized data, as can be seen from Table 5. After applying these two pruning criteria, we were left with 5 redescrptions. The results are given in Table 6.

Both formulae in the first pair shown in Table 6, for example, describe a set of students that have studied their first-year courses according to the old curriculum. The second pair of formulae is about the students that have studied their first-year courses according to the new curriculum. The fifth pair of formulae is also describing the similar set of students, but this time with somewhat different set of attributes. The results with MID were similar in fashion, and they are omitted here. From the randomized data MID did find many redescrptions but, as all of those redescrptions had p -value higher than the highest p -value of the redescrptions from the original data, they were pruned and thus all results by MID can be considered significant.

While the results from the GREEDY algorithm with the **Courses** data originally contained lots of redundancy, the results from the **Web** data were free of redundancy. The similarity of the results was also lower than in the results from **Courses**, hinting that the data does not contain subgroups that are easy to describe.

However, the results can be considered significant, as neither of the algorithms found any results from the randomized data (Table 7). The GREEDY algorithm gave 9 redescrptions, some of which are reported in Table 8.

Due to the nature of the data, the redescrptions in Table 8 mostly refer to courses' home pages: course pages constitute a substantial part of the data and they tend to be very homogeneous, all containing similar words. Describing, for example, people's personal home pages is much harder due to the heterogeneity of them. All of the examples in Table 8 describe some set of course home pages: the first pair, for example, is about courses held in autumn or winter and not in Seattle.

The MID algorithm reported 56 redescrptions having similarity at least 0.4. Some of the redescrptions are given in Table 9. The first two redescrptions, both having very high similarity, exploit the usual structure in web pages: universities' and people's home pages tend to mention some address. The third redescription is similar to those reported by GREEDY, though with smaller similarity.

The third data, the DBLP data, has a different structure. A redescription in this data describes a group of computer scientists via their publication patterns and co-authorships. As a such, it is easier to describe what a certain computer scientist is *not* than to say what she is. However, saying that those who do not publish in data mining conferences have not published with certain data miners is not as interesting as it is to say the opposite. Therefore, for this data, we applied an extra restriction to the descriptions: For the GREEDY algorithm the formulae must be monotone, i.e., they can only test if variable equals to 1, and for the MID algorithm the itemsets can be joined using only conjunctions and negations.

The GREEDY algorithm again reported very small set of redescrptions, only 8, and the similarities were quite low, varying between 0.35 to 0.25 (0.2 being the minimum allowed). The results were, again, very significant, as nothing could be found from the randomized data even with as low of a minimum similarity as 0.1 (Table 10). Some of the results by GREEDY are reported in Table 11.

The results in Table 11 show three groups of conferences and authors: machine learning (redescription 1), theoretical computer science (redescription 2), and data mining (redescription 3).

The results from the MID algorithm are similar to those of GREEDY, but with somewhat more structure and less similarity. As with GREEDY all results can be considered significant, as no results with accuracy higher than 0.1 were found from the randomized data (Table 10). Some example results of MID are given in

Table 4: Properties of the real datasets.

Data	Description	Rows	Columns	Density
Courses	students $\times$ courses	2401	106	0.1223
Web	pages $\times$ terms in pages	1051	1798	0.0378
	pages $\times$ terms in links	1051	438	0.0060
DBLP	authors $\times$ conferences	2345	19	0.1938
	authors $\times$ authors	2345	2345	0.0049

Table 5: Number of redescription found from original and randomized **Courses** data; $N = 200$.

Algorithm	Data	$J_{\min}$					
		0.4	0.5	0.6	0.7	0.8	0.9
GREEDY	Orig.	200	200	200	200	200	56
	Rand.	200	200	200	200	167	0
MID	Orig.	150	119	53	19	0	0
	Rand.	0	0	0	0	0	0

Table 12. To improve the readability, we write formulae of type $\neg(A = 1 \wedge B = 1)$ as $A = 0 \vee B = 0$; thus the results contain disjunctions even if the algorithm did not use them. The redescription given in Table 12 describe a subgroup of theoretical computer scientists with the special emphasis on avoiding those with research also in the field of data mining.

5 Related Work

Redescription rule mining was introduced by Ramakrishnan et al. [16], and they also presented an algorithm for it. Their algorithm, CARTwheels, is based on learning decision trees and, like our algorithms, it does not put any restriction to the type of the formulae in redescription, but rather limits the number of variables in formulae. Since that, other algorithms have been proposed, most of which concentrate only on special type of Boolean formulae. For example, Zaki and Ramakrishnan [22] give an algorithm to find exact minimal conjunctive redescription while Parida and Ramakrishnan [15] give algorithms for finding exact and approximate monotone redescription in CNF or DNF. Parida and Ramakrishnan also study several theoretical properties of redescription.

Recently, Kumar et al. [11] extended redescription mining framework to the fascinating task of *storytelling*, where the goal is to find consecutive redescription that relate completely disjoint elements. Notice that this is different to our goal of finding m -tuples of formulae, as we require the mutual similarity of the formulae to be high, while in storytelling, the similarities between

descriptions next to each other is required to be high, but the similarity between first and last formulae must be zero.

Mining association rules and their variants has been, and still is, an active area of research. See [9] for a thorough survey. Redescription mining can be seen as a one generalization of association rule mining. Other generalizations include, for example, the negative association rules [17], i.e., rules of type $A \rightarrow \neg B$, $\neg A \rightarrow B$, and $\neg A \rightarrow \neg B$.

An adaptation of the standard classification rule learning methods to subgroups discovery is proposed in [12]. The method proposed in [12] seeks for the best subgroups in terms of rule coverage and distributional usualness. A weighted relative accuracy heuristic is used to trade off generality and accuracy of a certain rule $\alpha \rightarrow A$, where A is a single target attribute. For other methods for subgroup discovery, see, e.g., [18].

Recently, Wu et al. [21] proposed an algorithm to mine positive and negative association rules. Their approach has some similarities with ours, as they define an interestingness measure and a confidence measure and use these and the frequency to prune the search space. However, excluding the frequency, their measures are different to ours. Also, they only consider itemsets and their negations, not general Boolean formulae, as we do. Negative association rules have also been applied with success for example to bioinformatics [1].

Another generalization to association rules are the disjunctive association rules that can contain disjunctions of itemsets. Nanavati et al. [13] propose an algorithm for mining them.

Both negative and disjunctive association rules still have restriction in the types of the Boolean expressions they consider, and they do not allow arbitrary Boolean expressions. Recently Zhao et al. [23] have proposed a framework to mine arbitrary Boolean expressions from binary data. While their algorithm is able to find all frequent expressions, and in theory could be used in redescription mining, its time complexity makes it difficult to apply for our purposes.

The idea of studying two parallel datasets is present also in co-training [2]. This technique has some similar-

Table 6: Results from the **Courses** dataset with GREEDY algorithm.

	Left formula	Right formula	J
1.	“Introduction to UNIX”=1 $\vee$ “Programming in Java”=0 $\vee$ “Introduction to Databases”=0 $\vee$ “Programming (Pascal)”=1	“Introduction to Programming”=0 $\vee$ “Introduction to Application Design”=0 $\vee$ “Programming Project”=0 $\vee$ “The Principles of ADP”=1	0.91
2.	“Database Systems I”=0 $\vee$ “Concurrent Systems (Fin)”=1 $\vee$ “Data Structures Project”=0 $\vee$ “Operating Systems I”=1	“Introduction to Application Design”=1 $\vee$ “Concurrent Systems (Eng)”=0 $\vee$ “Database Management”=1 $\vee$ “Data Structures”=1	0.91
3.	“Information Systems Project”=1 $\wedge$ “Data Structures Project”=1 $\wedge$ “Database Management”=0 $\wedge$ “Elementary Studies in Computer Science”=0	“Database Systems I”=1 $\wedge$ “Information Systems”=1 $\wedge$ “Database Application Project”=0 $\wedge$ “Operating Systems I”=0	0.90
4.	“Information Systems”=1 $\vee$ “Introduction to Databases”=0 $\vee$ “Managing Software Projects”=1 $\vee$ “Corba Architecture”=1	“Introduction to Application Design”=0 $\vee$ “Database Systems I”=1 $\vee$ “Languages for AI”=1 $\vee$ “Unix Softwareplatform”=1	0.90
5.	(“Programming in Java”=1 $\wedge$ “Programming (Pascal)”=0 $\vee$ “Introduction to Application Design”=1) $\wedge$ “Software Engineering”=0	(“Operating Systems I”=1 $\vee$ “Introduction to Programming”=1 $\vee$ “Introduction to Databases”=1) $\wedge$ “Corba Architecture”=0	0.90

Table 7: Number of redescrptions found from original and randomized **Web** data.

Algorithm	Data	$J_{\min}$					
		0.4	0.5	0.6	0.7	0.8	0.9
GREEDY	Orig.	52	50	13	5	0	0
	Rand.	0	0	0	0	0	0
MID	Orig.	56	56	56	56	55	0
	Rand.	0	0	0	0	0	0

ities with redescription mining. In the problem setting one is given two parallel datasets with some, but not all, transactions being labeled. The goal is to build two classifiers to classify the data. The building of these classifiers is done iteratively, using the labeling given by one of the classifier as a training data to the other, after which the roles are switched. This approach has some similarities with the CARTwheels algorithm by Ramakrishnan et al. [16] and indeed, in redescription mining, if we are given a formula $\alpha \in \Gamma(D_1)$, finding the formula $\beta \in \Gamma(D_2)$ can be considered as a machine learning task.

The concept of cross-mining binary and numerical attributes was studied recently in [6]. In short, in the cross-mining problem we are given a set of transactions, each of which contains two types of attributes, Boolean and real-valued, and the goal is to segment the real-valued data so that the segments correspond to itemsets in Boolean data and segment the real-valued data well.

In a Boolean data table, a pattern is defined as a set of rows that share the same values in two or more columns. Similarly, it can be identified as a set of items occurring in the same transactions. Unfortunately, in real-world problems the number of all the possible patterns is prohibitively large. Quantifying the interestingness of patterns found is thus often mandatory in order to prune the search space. Several methods were defined in the last decade to define the “interestingness” of a pattern: conciseness, generality, reliability, peculiarity, diversity, novelty, surprisingness, utility, applicability. A survey on the most prominent interestingness measures is given in [7].

In [4] a scalable method to mine interesting and non-redundant patterns from a dataset is proposed. A new statistical measure of interestingness is devised in order to measure the “surprisingness” of a pattern under two different null models. An extension of the method for parallel datasets is given in [5].

6 Concluding remarks

We have considered a version of the redescription mining problem, i.e., finding Boolean formulae α and β such that the rules $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ both hold with a reasonable accuracy. The problem is motivated by the general goal of data mining: finding unexpected viewpoints to the data.

We gave two simple heuristic algorithms for the task. The results are pruned by requiring that the subgroup is large enough, the Jaccard similarity between

Table 8: Results from the **Web** data with the GREEDY algorithm. Left formulae are over the terms in web pages and right formulae are over terms in links.

	Left formula	Right formula	J
1.	$(\text{autumn}=1 \vee \text{cse}=1) \wedge \text{seattl}=0 \wedge \text{intern}=0$	$(\text{cse}=1 \vee \text{autumn}=1) \wedge \text{other}=0 \vee \text{winter}=1$	0.75
2.	$(\text{grade}=1 \wedge \text{section}=1 \vee \text{wendt}=0) \wedge \text{hour}=1$	$(\text{section}=1 \vee \text{lectur}=1) \wedge \text{system}=0 \wedge \text{program}=0$	0.63
3.	$(\text{lectur}=1 \vee \text{instructor}=1) \wedge \text{fax}=0 \vee \text{exam}=1$	$\text{cs}=1 \vee \text{cse}=1 \vee \text{report}=1 \vee \text{introduc}=1$	0.55

Table 9: Results from the **Web** data with the MID algorithm. Left formulae are over the terms in web pages and right formulae are over terms in links.

	Left formula	Right formula	J
1.	$\text{dayton}=0 \vee \text{wi}=0 \vee \text{madison}=0$	$\text{back}=0 \vee \text{home}=0 \vee \text{page}=0 \vee \text{to}=0$	0.88
2.	$\text{texa}=0 \vee \text{austin}=0$	$\text{back}=0 \vee \text{home}=0 \vee \text{page}=0 \vee \text{to}=0$	0.83
3.	$\text{instructor}=1 \wedge \text{assign}=1 \wedge \text{hour}=1$	$\text{section}=1 \wedge \text{cs}=1 \vee \text{cse}=1$	0.34

Table 10: Number of descriptions found from original and randomized DBLP data.

Algorithm	Data	$J_{\min}$						
		0.1	0.15	0.2	0.25	0.3	0.35	0.4
GREEDY	Orig.	8	8	8	8	5	1	0
	Rand.	0	0	0	0	0	0	0
MID	Orig.	2440	0	0	0	0	0	0
	Rand.	0	0	0	0	0	0	0

Table 11: Results from the DBLP data with the GREEDY algorithm. Left formulae are over the conferences and right formulae are over the co-authors.

	Left formula	Right formula	J
1.	$\text{COLT}=1 \wedge \text{ICML}=1$	$\text{"Michael J. Kearns"}=1 \vee \text{"Peter L. Bartlett"}=1 \vee \text{"Robert E. Schapire"}=1 \vee \text{"Phillip M. Long"}=1$	0.35
2.	$\text{SODA}=1 \wedge \text{FOCS}=1 \wedge \text{STOC}=1$	$\text{"Noga Alon"}=1 \vee \text{"Frank Thomson Leighton"}=1 \vee \text{"Leonidas J. Guibas"}=1 \vee \text{"Sampath Kannan"}=1$	0.32
3.	$\text{SDM}=1 \wedge \text{ICDE}=1$	$\text{"Philip S. Yu"}=1 \vee \text{"Matthias Schubert"}=1 \vee \text{"Jessica Lin"}=1 \vee \text{"Yiming Ma"}=1$	0.30

Table 12: Results from the DBLP data with the MID algorithm. Left formulae are over the conferences and right formulae are over the co-authors.

	Left formula	Right formula	J
1.	$\text{SODA}=1 \wedge \text{FOCS}=1 \wedge \text{STOC}=1 \wedge (\text{KDD}=0 \vee \text{PODS}=0)$	$\text{"Noga Alon"}=1 \wedge \text{"Hector Garcia-Molina"}=0$	0.13
2.	$\text{SODA}=1 \wedge \text{FOCS}=1 \wedge \text{STOC}=1 \wedge (\text{PKDD}=0 \vee \text{ICML}=0)$	$\text{"Sanjeev Khanna"}=1 \wedge \text{"Rakesh Agrawal"}=0$	0.09

the formulae is large enough, and the p -value of the formulae being approximately equivalent is small enough. The significance of the results was evaluated by using swap randomization.

We tested the algorithms on synthetic data, showing that the methods can find planted subgroup descriptions, and that on completely random data the methods do not find many descriptions. On real datasets the algorithms find small sets of results, and the results are easy to interpret. Swap randomization shows that on randomized versions of the datasets the algorithms do not find anything: this implies that the results are not due to chance.

There are obviously several open problems. Our algorithms are quite simple; it would be interesting to know whether more complex methods would be useful. Also, evaluation of the methods on, say, genetic marker data is being planned.

Generalizing the approach to other types of queries is – at least in principle – straightforward. For example, if the data is real-valued, we can ask whether there are subgroups that have at least two different definitions by sets of inequalities. Algorithmically, such variants seem challenging.

Acknowledgments

The authors are grateful to Gemma Garriga, Aristides Gionis, and Evimaria Terzi for their helpful comments.

References

- [1] I. I. Artamonova, G. Frishman, and D. Frishman. Applying negative rule mining to improve genome annotation. *BMC Bioinformatics*, 8(261), 2007. <http://www.biomedcentral.com/1471-2105/8/261>.
- [2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.
- [3] G. W. Cobb and Y.-P. Chen. An application of Markov chain Monte Carlo to community ecology. *American Mathematical Monthly*, 110:264–288, 2003.
- [4] A. Gallo, T. De Bie, and N. Cristianini. MINI: Mining informative non-redundant itemsets. In *PKDD*, pages 438–445, 2007.
- [5] A. Gallo, T. De Bie, and N. Cristianini. Mining informative patterns in large databases. Technical report, University of Bristol, UK, September 2007. <http://patterns.enm.bris.ac.uk/node/161>.
- [6] G. C. Garriga, H. Heikinheimo, and J. K. Seppänen. Cross-mining binary and numerical attributes. In *ICDM*, 2007. To appear.
- [7] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), 2006. Article 9.
- [8] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data*, 1(3), 2007.
- [9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, second edition, 2006.
- [10] E. Keogh, S. Lonardi, and B. Chiu. Finding surprising patterns in a time series database in linear time and space. In *KDD*, pages 550–556, 2002.
- [11] D. Kumar et al. Algorithms for storytelling. In *KDD*, pages 604–610, 2006.
- [12] N. Lavrac, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *The Journal of Machine Learning Research*, 5:153–188, 2004.
- [13] A. A. Nanavati, K. P. Chitrapura, S. Joshi, and R. Krishnapuram. Mining generalised disjunctive association rules. In *CIKM*, pages 482–489, 2001.
- [14] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.
- [15] L. Parida and N. Ramakrishnan. Redescription mining: Structure theory and algorithms. In *AAAI*, pages 837–844, 2005.
- [16] N. Ramakrishnan et al. Turning CARTwheels: An alternating algorithm for mining redescriptions. In *KDD*, pages 266–275, 2004.
- [17] A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. In *ICDE*, pages 494–505, 1998.
- [18] M. Sholz. Sampling-based sequential subgroup mining. In *KDD*, pages 265–274, 2005.
- [19] G. Webb. Discovering significant rules. In *KDD*, pages 434–443, 2006.
- [20] G. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.
- [21] X. Wu, C. Zhang, and S. Zhang. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems*, 22(3):381–405, 2004.
- [22] M. J. Zaki and N. Ramakrishnan. Reasoning about sets using redescription mining. In *KDD*, pages 364–373, 2005.
- [23] L. Zhao, M. Zaki, and N. Ramakrishnan. BLOSUM: a framework for mining arbitrary Boolean expressions. In *KDD*, pages 827–832, 2006.

Standing Out in a Crowd: Selecting Attributes for Maximum Visibility

Muhammed Miah^{#1}, Gautam Das^{#2}, Vagelis Hristidis^{*}, Heikki Mannila⁺

[#]*Department of Computer Science and Engineering, University of Texas at Arlington*

416 Yates Street, Arlington, TX 76019, USA

¹*mzmiah@uta.edu*

²*gdas@uta.edu*

^{*}*School of Computing and Information Sciences, Florida International University*

11200 S.W. 8th Street, Miami, FL 33199, USA

vagelis@cis.fiu.edu

⁺*HIIT, Helsinki University of Technology and University of Helsinki*

P.O Box 68, FI 00014, University of Helsinki, Helsinki, Finland

Heikki.Mannila@tkk.fi

Abstract— In recent years, there has been significant interest in development of ranking functions and efficient top- k retrieval algorithms to help users in ad-hoc search and retrieval in databases (e.g., buyers searching for products in a catalog). In this paper we focus on a novel and complementary problem: how to guide a seller in selecting the best attributes of a new tuple (e.g., new product) to highlight such that it stands out in the crowd of existing competitive products and is widely visible to the pool of potential buyers. We develop several interesting formulations of this problem. Although these problems are NP-complete, we can give several exact algorithms as well as approximation heuristics that work well in practice. Our exact algorithms are based on Integer Programming (IP) formulations of the problems, as well as on adaptations of maximal frequent itemset mining algorithms, while our approximation algorithms are based on greedy heuristics. We conduct a performance study illustrating the benefits of our methods on real as well as synthetic data.

I. INTRODUCTION

In recent years, there has been significant interest in developing effective techniques for ad-hoc search and retrieval in unstructured as well as structured data repositories, such as text collections and relational databases. In particular, a large number of emerging applications require exploratory querying on such databases; examples include users wishing to search databases and catalogs of products such as homes, cars, cameras, restaurants, or articles such as news and job ads. Users browsing these databases typically execute search queries via public front-end interfaces to these databases. Typical queries may specify sets of keywords in case of text databases, or the desired values of certain attributes in case of structured relational databases. The query-answering system answers such queries by either returning all data objects that satisfy the query conditions (*Boolean retrieval*), or may rank and return the top- k data objects using suitable scoring functions (*Top- k retrieval*).

However, the focus of this paper is *not* on new search and retrieval techniques that will aid users in effective exploration of such databases. Rather, the focus is on a complementary yet novel problem of data exploration, as described below.

Selecting Attributes for Maximum Visibility. We distinguish between two types of users of these databases: users who search such databases trying to locate objects of interest, *as well as* users who insert new objects into these databases in the hope that they will be easily discovered by the first types of users. For example, in a database representing an e-marketplace (such as Craigslist.com, or the classified ads section of newspapers), the former type of users are potential *buyers* of products, while the latter type of users are *sellers* or *manufacturers* of products – where products could range from automobiles to phones to rental apartments to job advertisements. We note that almost all of the prior research efforts on effective search and retrieval techniques – such as new top- k algorithms, new ranking functions, and so on – have been designed with the first kind of user in mind (i.e., the buyer). In contrast, relatively less research has been expended towards developing techniques to help a seller/manufacture insert a new product for sale in such databases that markets it in the best possible manner – i.e., such that it *stands out in a crowd* of competitive products and is *widely visible* to the pool of potential buyers.

It is this latter problem that is the main focus of this paper. To understand it a little better, consider the following real-world scenario: assume that we wish to insert a classified ad in an online newspaper to advertise an apartment for rent. Our apartment may have numerous attributes (it has two bedrooms, electricity will be paid by the owner, it is near a train station, etc). However, due to the ad costs involved, it is not possible for us to describe all attributes in the ad. So we have to select, say the ten best attributes. Which ones should we select? Thus, one may view our effort as an attempt to build a recommendation system for *sellers*, unlike the more

traditional recommendation systems for buyers. It may also be viewed as *inverting* a ranking function, i.e., determining the argument of a ranking function that will lead to high ranking scores.

This general problem also arises in domains beyond e-commerce applications. For example, in the design of a new product, a manufacturer may be interested in selecting the ten best features from a large wish-list of possible features – e.g., a homebuilder can find out that adding a swimming pool really increases visibility of a new home in a certain neighborhood. Likewise, we may be interested in developing a catchy title, or selecting a few important indexing keywords, for a scientific article.

To define our problem more formally, we need to develop a few abstractions. Let D be the database of products already being advertised in the marketplace (i.e., the “competition”). In addition, let Q be the set of search queries that have been executed against this database in the recent past – thus Q is the “workload” or “query log”. The query log is our primary model of what past potential buyers have been interested in. Consider a new product t that needs to be inserted into this database. While the product has numerous attributes, due to budget constraints there is a limit, say m , on the number of attributes that can be selected for entry into the database. Our problem can now be defined as follows:

PROBLEM: Given a database D , a query log Q , a new tuple t , and an integer m , determine the best (i.e., top- m) attributes of t to retain such that if the shortened version of t is inserted into the database, the number of queries of Q that retrieve t is maximized.

In this paper we initiate a thorough investigation of this novel optimization problem. We mainly focus on an important variant where the data is Boolean and the queries follow conjunctive retrieval semantics. We also briefly consider several important variants, including Boolean data with other types of retrieval semantics (disjunctive as well as top- k retrieval), as well as categorical, numeric and text data. Our main contributions are summarized below:

1. We introduce the problem of *selecting attributes of a tuple for maximum visibility* as a new data exploration problem that benefits a certain class of users interested in designing and marketing their products. We consider several interesting variants of the problem as well as diverse application scenarios.
2. We show that the problem is NP-complete.
3. We give exact Integer Programming (IP) and Integer Linear Programming (ILP)-based algorithms to solve the problem. These algorithms are effective for moderate-sized problem instances.

4. We also develop more scalable optimal solutions based on novel adaptations of maximal frequent itemset algorithms. Furthermore, in contrast to ILP-based solutions, we can leverage preprocessing opportunities in these approaches.
5. We also present fast greedy approximation algorithms that work well in practice.
6. We perform detailed performance evaluations on both real as well as synthetic data to demonstrate the effectiveness of our developed algorithms.

The rest of the paper is organized as follows. In Section II we give a formal definition of our problem, including several interesting problem variants. Section III analyzes the computational complexity of the problem, and Section IV presents various optimal algorithms as well as heuristics. In Section V we extend these techniques to also work for other problem variants. In Section VI we discuss related work, and present the result of extensive experiments in Section VII. Section VIII is a short conclusion.

II. PROBLEM FRAMEWORK

In this section we first develop, with a detailed example, a formal definition of the problem for the case of Boolean data and a query log of conjunctive Boolean queries. We then briefly define other Boolean variants, as well as extensions for other kinds of data such as categorical, text and numeric data. Due to space constraints, throughout the paper we primarily focus on the first Boolean variant, and restrict our discussion of other variants to briefly outlining the extensions necessary.

A. A Boolean Problem Variant.

Some useful definitions and notations are given here.

Database: Let $D = \{t_1 \dots t_N\}$ be a collection of Boolean tuples over the attribute set $A = \{a_1 \dots a_M\}$, where each tuple t is a bit-vector where a 0 implies the absence of a feature and a 1 implies the presence of a feature. A tuple t may also be considered as a subset of A , where an attribute belongs to t if its value in the bit-vector is 1.

Tuple Domination: Let t_1 and t_2 be two tuples such that for all attributes for which tuple t_1 has value 1, tuple t_2 also has value 1. In this case we *say* that t_2 *dominates* t_1 .

Tuple Compression: Let t be a tuple and let t' be a subset of t with m attributes. Thus t' represents a compressed representation of t . Equivalently, in the bit-vector *representation* of t , we retain only m 1's and convert the rest to 0's.

Query: We view each query as a subset of attributes. The retrieval semantics is *Conjunctive Boolean Retrieval*, e.g., a query such as $\{a_1, a_3\}$ is equivalent to “return all tuples such that $a_1 = 1$ and $a_3 = 1$ ”. Alternatively, if we view q as a special

type of “tuple”, then t dominates q . The set of returned tuples is denoted as $R(q)$.

Query Log: Let $Q = \{q_1 \dots q_s\}$ be collection of queries where each query q defines a subset of attributes.

We are now ready to formally define the main problem variant considered in the paper.

PROBLEM SOC-CB-QL (or “Stand Out in a Crowd-Conjunctive Boolean-Query Log”): Given a query log Q with Conjunctive Boolean Retrieval semantics, a new tuple t , and an integer m , compute a compressed tuple t' by retaining m attributes such that the number of queries that retrieve t' is maximized.

Intuitively, for buyers interested in browsing products of interest, we wish to ensure that the compressed version of the new product is visible to as many buyers as possible. The following running example will be used to illustrate Problem SOC-CB-QL (as well as other variants later in the paper).

Car ID	AC	Four Door	Turbo	Power Doors	Auto Trans	Power Brakes
t_1	0	1	0	1	0	0
t_2	0	1	1	0	0	0
t_3	1	0	0	1	1	1
t_4	1	1	0	1	0	1
t_5	1	1	0	0	0	0
t_6	0	1	0	1	0	0
t_7	0	0	1	1	0	0

Database D

Query ID	AC	Four Door	Turbo	Power Doors	Auto Trans	Power Brakes
q_1	1	1	0	0	0	0
q_2	1	0	0	1	0	0
q_3	0	1	0	1	0	0
q_4	0	0	0	1	0	1
q_5	0	0	1	0	1	0

Query Log Q

New Car	AC	Four Door	Turbo	Power Doors	Auto Trans	Power Brakes
t	1	1	0	1	1	1

New tuple t to be inserted

Fig 1 Illustrating EXAMPLE 1

EXAMPLE 1: Consider an inventory database of an auto dealer, which contains a single database table D with $N=7$ rows and $M=6$ attributes where each tuple represents a car for sale. The table has numerous attributes that describe

details of the car: Boolean attributes such as AC, Four Door, etc; categorical attributes such as Make, Color, etc; numeric attributes such as Price, Age, etc; and text attributes such as Reviews, Accident History, and so on. Fig 1 illustrates such a database (where only the Boolean attributes are shown) of seven cars already advertised for sale. The figure also illustrates a query log of five queries, and a new car t that needs to be advertised, i.e., inserted into this database. $\square$

Suppose we are required to retain $m = 3$ attributes of the new tuple. It is not hard to see that if we retain the attributes AC, Four Door, and Power Doors (i.e., $t' = [1, 1, 0, 1, 0, 0]$), we can satisfy a maximum of three queries (q_1 , q_2 , and q_3). No other selection of three attributes of the new tuple will satisfy more queries. Notice that in SOC-CB-QL, it is the query log Q that needs to be analyzed in solving the problem; the actual database D (i.e., the “competing products”) is irrelevant. That is, there is no need of access to the database.

B. Other Problem Variants.

Here we define several other variants of the problem that are useful in different applications. Problem SOC-CB-QL has a per-attribute version where m is not specified; in this case we may wish to determine t' such that the number of satisfied queries divided by $|t'|$ is maximized. Intuitively, if the number of attributes retained is a measure of the cost of advertising/manufacturing the new product, this problem maximizes the number of potential buyers per unit cost.

A complementary variant to SOC-CB-QL is SOC-CB-D (or “Stand Out in a Crowd-Conjunctive Boolean-Data”): given a database D , a new tuple t , and an integer m , we wish to compute a compressed tuple t' by retaining m attributes such that the number of tuples in D dominated by t' is maximized. This is useful in scenarios where we have access to the database, but do not have access to the query log. To illustrate this variant, consider the example in Fig 1 again. Suppose we are required to retain $m = 4$ attributes of the new tuple t . It is not hard to see that if we retain the four attributes AC, Four Door, Power Doors and Power Brakes (i.e., $t' = [1, 1, 0, 1, 0, 1]$), we dominate four tuples (t_1 , t_4 , t_5 and t_6). No other selection of four attributes of the new tuple will dominate more tuples. It is also easy to see that any algorithm that solves SOC-CB-QL can be also used to solve SOC-CB-D, by replacing the query log with the database as input. SOC-CB-D also has a natural per-attribute version.

A more general problem variant arises when the retrieval semantics is not simply conjunctive Boolean; e.g., the retrieval semantics could be *disjunctive Boolean* or even *Top-k retrieval*. To understand the latter, let $score(q, t)$ be a scoring function that returns a real-valued score for any tuple t . Let k be a small integer associated with a query q . Then $R(q)$ is defined as the set of top- k tuples in the database with the highest scores. The problem variant SOC-Topk is defined as: given a database D , a query log Q with top- k retrieval semantics via a scoring function, a new tuple t , and an integer m , compute a compressed tuple t' by retaining m attributes

such that the number of queries that retrieve t' is maximized. Solving this problem requires access to both the query log as well as the database.

The above problems are not restricted only to Boolean databases. *Categorical databases* are natural extensions of Boolean databases where each attribute a_i can take one of several values from a multi-valued categorical domain Dom_i . Problem variants similar to Boolean can be defined for categorical databases. Furthermore, *text databases* consist of a collection of documents, where each document is modeled as a bag of words as is common in Information Retrieval. Queries are sets of keywords, with top- k retrieval via scoring functions such as the tf-idf-based BM25 scoring function [19]. *SOC-Topk* can be directly mapped to a corresponding problem for text data if we view a text database as a Boolean database with each distinct keyword considered as a Boolean attribute. This problem arises in several applications, e.g. when we wish to post a classified ad in an online newspaper and need to specify important keywords that will enable the ad to be visible to the maximum number of potential buyers. Finally, the above problems can be extended to *numeric databases*, i.e., databases with numeric attributes, where queries specify ranges over a subset of attributes. For example, users browsing a database for digital cameras may specify desired ranges on price, weight, resolution, etc, and the returned results may be ranked by price.

In the rest of the paper we primarily focus on analyzing the complexity and developing solutions for our main variant *SOC-CB-QL* defined in Section II.A; due to lack of space our discussion of the other variants is restricted to brief outlining the extensions necessary of our proposed approaches.

III. COMPLEXITY RESULTS

In this section we show that our main problem variant is NP-complete.

Theorem 1: *SOC-CB-QL is NP-complete.*

Proof (sketch): The membership of the decision version of the problem in NP is obvious. To see NP-hardness, we reduce the Clique problem to *SOC-CB-QL*. Given a graph $G = (V, E)$ and an integer r , the task in the Clique problem is to check if there is a clique of size r in G . We transform this to an instance of *SOC-CB-QL* as follows. The attribute set A will be V , and the query log will contain one row for each edge. If $e = (u, v)$ is an edge, then the query log Q contains the conjunctive query $\{u, v\}$, i.e., the query retrieving all tuples with $u=1$ and $v=1$. The new tuple t has all the attributes in V set to 1. Let $m = r$. It is straightforward to verify that t has a compressed representation with m attributes that satisfies $m(m-1)/2$ queries if and only if the graph G has a clique of size r . $\square$

It is not hard to show that the same proof can be extended to show that all the other variants described in Section II.B are also NP-hard. We omit further details due to lack of space.

IV. ALGORITHMS FOR SOC-CB-QL

In this section we discuss our main algorithmic results. We restrict our discussion to Problem *SOC-CB-QL*, and defer discussion of other problem variants to Section V.

A. Optimal Brute Force Algorithm.

Clearly, since *SOC-CB-QL* is NP-hard, it is unlikely that any optimal algorithm will run in polynomial time in the worst case. The problem can be obviously solved by a simple brute force algorithm (henceforth called *BruteForce-SOC-CB-QL*), which simply considers all combinations of m -attributes of the new tuple t and determines the combination that will satisfy the maximum number of queries in the query log Q . However, we are interested in developing optimal algorithms that work much better for typical problem instances. We discuss such algorithms next.

B. Optimal Algorithm Based on ILP.

We next show how *SOC-CB-QL* can be described in an integer programming (ILP) framework. Let the new tuple be the Boolean vector $t = \{a_1(t), \dots, a_M(t)\}$, and let $x_1, \dots, x_M$ be integer variables such that if $a_i(t) = 1$ then $x_i \in \{0, 1\}$, else $x_i = 0$. Consider the task:

$$\text{Maximize } \sum_{i=1}^S \prod_{a_j \in q_i} x_j \text{ subject to } \sum_{j=1}^M x_j \leq m$$

It is easy to see that the maximum gives exactly the solution to *SOC-CB-QL*. The objective function is not linear, however, and thus we next show how this can be achieved.

We introduce additional 0-1 integer variables $y_1, \dots, y_S$, i.e., one variable for each query in Q .

$$\text{Maximize } \sum_{i=1}^S y_i \text{ subject to}$$

$$\sum_{j=1}^M x_j \leq m \text{ and } y_i \leq x_j \text{ for each } j \text{ and } i \text{ such that } a_j \in q_i$$

Thus, the variable y_i corresponding to a query can be 1 only if all the variables x_j corresponding to the attributes in the query are 1. This implies that the maximum remains the same. We refer to the above algorithm as *ILP-SOC-CB-QL*. The integer *linear* formulation is particularly attractive as unlike more general IP solvers, ILP solvers are usually more efficient in practice.

C. Optimal Algorithm Based on Maximal Frequent Itemsets.

The algorithm based on Integer Linear Programming described in the previous subsection has certain limitations; it is impractical for problem instances beyond a few hundred queries in the query log. The reason is that it is a very generic method for solving arbitrary integer linear programming formulations, and consequently fails to leverage the specific nature of our problem. In this subsection we develop an alternate approach that scales very well to large query logs. This algorithm, called *MaxFreqItemSets-SOC-CB-QL*, is based on an interesting adaptation of an algorithm for mining

Maximal Frequent Itemsets [11]. We first define the frequent itemset problem:

The Frequent Itemset Problem. Let R be a N -row M -column Boolean table, and let $r > 0$ be an integer known as the threshold. Given an itemset I (i.e., a subset of attributes), let $\text{freq}(I)$ be defined as the number of rows in R that “support” I (i.e., the set of attributes corresponding to the 1’s in the row is a superset of I). Compute all itemsets I such that $\text{freq}(I) > r$.

Computing frequent itemsets is a well studied problem and there are several scalable algorithms that work well when R is sparse and the threshold is suitably large. Examples of such algorithms include [2, 14]. In our case, given a new tuple t , recall that our task is to compute t' , a compression of t by retaining only m attributes, such that the number of queries that satisfy t' is maximized. This suggests that we may be able to leverage algorithms for frequent itemsets mining over Q for this purpose. However, there are several important complications that need to be overcome.

Complementing the Query Log. Firstly, in itemset mining, a row of the Boolean table is said to support an itemset if the row is a superset of the itemset. In our case, a query satisfies a tuple if it is a subset of the tuple. To overcome this conflict, our first task is to *complement* our problem instance, i.e., convert 1’s to 0’s and vice versa. Let $\sim t$ ($\sim q$) denote the complement of a tuple t (query q), i.e., where the 1’s and 0’s have been interchanged. Likewise let $\sim Q$ denote the complement of a query log Q where each query has been complemented. Now, $\text{freq}(\sim t)$ can be defined as the number of rows in $\sim Q$ that support $\sim t$.

The rest of the approach is now seemingly clear: compute all frequent itemsets of $\sim Q$ (using an appropriate threshold to be discussed later), and from among all frequent itemsets of size $M - m$, determine the itemset I that is a superset of $\sim t$ with the highest frequency. The optimal compressed tuple t' is therefore the complement of I , i.e., $\sim I$.

However, the problem is that Q is itself a sparse table, as the queries in most search applications involve the specification of just a few attributes. Consequently, the complement $\sim Q$ is an extremely dense table, and this prevents most frequent itemset algorithms from being directly applicable to $\sim Q$. For example, most “level-wise algorithms” (such as Apriori [2], which operates level by level of the Boolean lattice over the attributes set by first computing the single itemsets, then itemsets of size 2, and so on) will only progress past just a few initial levels before being overcome by an intractable explosion in the size of candidate sets. To see this, consider a table with $M=50$ attributes, and let $m = 10$. To determine a compressed tuple t' with 10 attributes, we need to know the itemset of $\sim Q$ of size 40 with maximum frequency. Due to the dense nature of $\sim Q$, algorithms such as Apriori will not be able to compute frequent itemsets beyond a size of 5-10 at the most. Likewise, the sheer number of frequent itemsets will also prevent other algorithms such as FP-Tree [14] from being effective.

Setting of the Threshold Parameter. Let us assume we can solve the itemset mining problem of extremely dense datasets. What should be the setting of the threshold? Clearly setting the threshold $r=1$ will solve *SOC-CB-QL* optimally. But this is likely to make any itemset mining algorithm impractically slow.

There are two alternate approaches to setting the threshold. One approach is essentially a heuristic, where we set the threshold to a reasonable fixed value dictated by the practicalities of the application. For example, setting the threshold as 1% of the query log size implies that we are attempting to compress t such that at least 1% of the queries are still able to retrieve the tuple. For a fixed threshold setting such as this, one of two possible outcomes can occur. If the optimal compression t' satisfies more than 1% of the queries, the algorithm will discover it. Otherwise, the algorithm will return empty.

The alternate adaptive procedure of setting the threshold is, first initialize the threshold to a high value and compute the frequent itemsets of $\sim Q$. If there are no frequent itemsets of size at least $M - m$ that are supersets of $\sim t$, repeat the process with a smaller threshold (say half of the previous threshold). This process is guaranteed to discover the optimal t' .

Random Walk to Compute Maximal Frequent Itemsets.

We now return back to the task of how to compute frequent itemsets of the dense Boolean table $\sim Q$. In fact, we do not compute all frequent itemsets of the dense table $\sim Q$, as we have already argued earlier that there will be prohibitively too many of them. Instead, our approach is to compute the *maximal frequent itemsets* of $\sim Q$. A maximal frequent itemset is a frequent itemset such that none of its supersets are frequent. The set of maximal frequent itemsets are much smaller than the set of all frequent itemsets. For example, if we have a dense table with M attributes, then it is quite likely that most of the maximal frequent itemsets will exist very high up in the Boolean lattice over the attributes, very close to the highest possible level M . Fig 2 shows a conceptual diagram of a Boolean lattice over a dense Boolean table $\sim Q$. The shaded region depicts the frequent itemsets and the maximal frequent itemsets are located at the highest positions of the border between the frequent and infrequent itemsets.

There exist several algorithms for computing maximal frequent itemsets, e.g. [3, 4, 13, 11]. Let us consider the *random walk* based algorithm in [11], which starts from a random singleton itemset I at the bottom of the lattice, and at each iteration, adds a random item to I (from among all items $A - I$ such that I remains frequent), until no further additions are possible. At this point a maximal frequent itemset I have been discovered. If the number of maximal frequent itemsets is relatively small, this is a practical algorithm: repeating this random walk a reasonable number of times will discover all maximal frequent itemsets with high probability. However, since this algorithm is based on traversing the lattice from bottom to top, the random walk will have to traverse a lot of

levels before it reaches a maximal frequent itemset of a dense table.

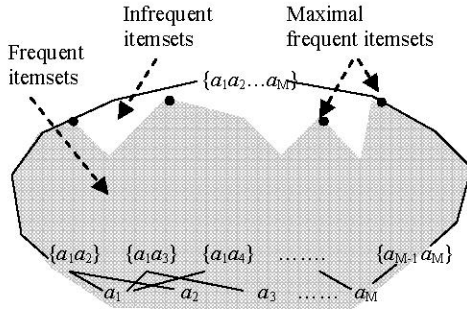


Fig 2 Maximal frequent itemsets in a Boolean Lattice

Instead, we propose an alternate approach which starts from the top of the lattice and traverses down. Our random walk can be divided into two phases: (a) *Down Phase*: starting from the top of the lattice ($I = \{a_1 a_2 \dots a_M\}$), walk down the lattice by removing random items from I until I becomes frequent, and (b) *Up Phase*: starting from I , walk up the lattice by adding random items to I (from among all items $A - I$ such that I remains frequent), until the no further additions are possible, resulting in a maximal frequent itemset I .

Fig 3 shows an example of the two phases of the random walk. What is important to note is that this process is much more efficient than a bottom-up traversal, as our walks are always confined to the top region of the lattice and we never have to traverse too many levels.

Number of Iterations. Repeating this two phase random walk several times will discover, with high probability, all the maximal frequent itemsets. The actual number of such iterations can be monitored adaptively; our approach is to stop the algorithm if each discovered maximal frequent itemset has been discovered at least twice (or a maximum number of iterations have been reached). This stopping heuristic is motivated by the *Good-Turing estimate* for computing the number of different objects via sampling [8].

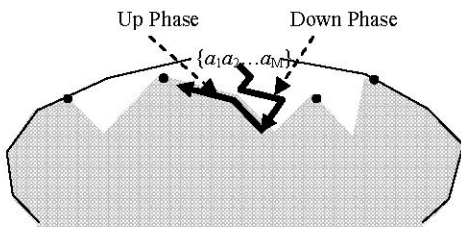


Fig 3 Two phase random walk

Frequent itemsets at level $M - m$. Finally, once all maximal frequent itemsets have been computed, we have to check which ones are supersets of $\sim t$. Then, for all possible subsets (of size $M - m$) of each such maximal frequent itemset (see Fig 4), we can determine that subset I that is (a) a superset of

$\sim t$, and (b) has the highest frequency. The optimal compressed tuple t' is therefore the complement of I , i.e., $\sim I$.

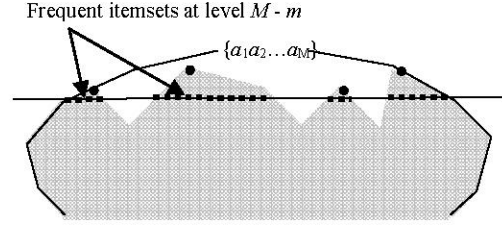


Fig 4 Checking frequent itemsets at level $M - m$

In summary, the pseudo-code of our algorithm *MaxFreqItemSets-SOC-CB-QL* is shown in Fig 5 (details of how certain parameters such as the threshold are set, are omitted from the pseudo-code).

Preprocessing Opportunities. Note that the algorithm also allows for certain operations to be performed in a preprocessing step. For example, all the maximal itemsets can be pre-computed, and the only task that needs to be done at runtime would be to determine, for a new tuple t , those itemsets that are supersets of $\sim t$ and have size $M - m$. If we know the range of m that is usually requested for compression in new tuples, we can even precompute all frequent itemsets for those values of m , and simply lookup the itemset with the highest frequency at runtime.

D. Greedy Algorithms.

While the maximal frequent itemset based algorithm has much better scalability properties than the ILP based algorithm, it too becomes prohibitive for really large datasets (query logs). Consequently, we also developed three (suboptimal) greedy heuristics for solving *SOC-CB-QL*, some of whom worked very well in our experiments.

The greedy algorithm *ConsumeAttr-SOC-CB-QL* first computes the number of times each individual attribute appear in the query log. It then selects the top- m attributes of the new tuple that have the highest frequencies. The greedy algorithm *ConsumeAttrCumul-SOC-CB-QL* is a cumulative version of the previous algorithm. It first selects the attribute with the highest individual frequency in the query log. It then selects the second attribute that co-occurs most frequently with the first attribute in the query log, and so on. Finally, instead of consuming attributes greedily, an alternative approach is to consume queries greedily. The greedy algorithm *ConsumeQueries-SOC-CB-QL* operates as follows. It first picks the query with minimum number of attributes, and selects all attributes specified in the query. It then picks the query with minimum number of new attributes (i.e., not already specified in the first query), and adds these new attributes to the selected list. This process is continued until m attributes have been selected.

```

Q: Query Log
t: new tuple
m: num attributes of t to retain
r: threshold  $\leftarrow$  suitable value
MaxFreqItemsets  $\leftarrow \{\}$ 
MaxNumIter  $\leftarrow$  suitable value

Algorithm TwoPhase-Random-Walk( $\sim Q, r$ )
  execute Down Phase random walk
  execute Up Phase random walk
  return itemset reached after Up Phase

Algorithm ComputeMaxFreqItemsets( $\sim Q, r$ )
  while
    ( $i++ \leq \text{MaxNumIter}$ ) and
    ( $\exists J$  in MaxFreqItemsets s.t.
       $\text{timesDiscovered}(J) = 1$ )
     $I \leftarrow \text{TwoPhase-Random-Walk}(\sim Q, r)$ 
     $\text{timesDiscovered}(I)++$ 
     $\text{MaxFreqItemsets} \leftarrow \text{MaxFreqItemsets} \cup \{I\}$ 

Algorithm MaxFreqItemSets-SOC-CB-QL( $\sim Q, r$ )
  ComputeMaxFreqItemsets( $\sim Q, r$ )
  let Itemsets(t)  $\leftarrow \{I \mid I \subseteq \text{MaxFreqItemsets},$ 
     $|I| = M - m, \text{ and } I \supseteq \sim t\}$ 
  let I be the itemset in Itemsets(t) with highest
    frequency
  return  $\sim I$ 

```

Fig 5 Algorithm MaxFreqItemSets-SOC-CB-QL

V. ALGORITHMS FOR OTHER PROBLEM VARIANTS

In this section we briefly outline how the algorithms developed in Section IV can be extended to solve the problem variants defined in Section II.B.

The per-attribute variant of *SOC-CB-QL* can be simply solved by trying out values of *m* between 1 and *M* (recall that *M* is the total number of attributes of the database) and making *M* calls to any of the algorithms discussed in Section IV, and selecting the solution that maximizes our objective.

SOC-CB-D can be solved using any algorithm for *SOC-CB-QL* by replacing the query log with the database.

For the *SOC-Topk* problem, since the retrieval semantics is not always conjunctive Boolean, the frequent itemset approach is not always applicable. In general, the problem can be formulated as a non-linear integer program. However, ILP formulations and frequent itemset mining approaches are possible in the case of *global* scoring functions – i.e., functions of the form $\text{score}(t)$ which are dependent on the tuple, but not on the query. For example, a user may be interested in getting the top-10 cars with AC and Turbo, ordered by decreasing number of available features – here the scoring function is the number of attributes in each tuple that have value 1. Another example of a global scoring function is ordering by a numeric attribute such as Price. We omit further details of these formulations from this version of the paper. Alternatively, the greedy algorithms of Section IV.D can be extended to the *SOC-Topk* problem.

The case of categorical data is a straightforward generalization of Boolean data. Likewise, as discussed in Section II.B, text data can be treated as Boolean data, and in principle all the algorithms developed for Boolean data can be used for text data. However, if we view each distinct keyword in the text corpus (or query log) as a distinct Boolean attribute, the dimension of the Boolean database is enormous. Consequently, the greedy approaches are the only ones feasible in this scenario.

Finally, problems involving numeric databases and query logs of range queries can be reduced to Boolean problem instances as follows: We first execute each query in the query log, and for each numeric attribute a_i in *Q*, we replace it by a Boolean attribute b_i as follows: if the *i*th range condition of query *q* contains the *i*th value of tuple *t*, then assign 1 to b_i for query *q*, else assign 0 to b_i for query *q*. I.e., each query has effectively been reduced to a Boolean row in a Boolean query log *Q*. The tuple *t* can be converted to a Boolean tuple consisting of all 1's. It is not hard to see that we have created an instance of *SOC-CB-QL*, whose solution will solve the corresponding problem for numeric data.

VI. RELATED WORK

A large corpus of work has tackled the problem of ranking the results of a query. In the documents world, the most popular techniques are tf-idf based [21] ranking functions, like BM25 [19], as well as link-structure-based techniques like PageRank [5] if such links are present (e.g., the Web). In the database world, automatic ranking techniques for the results of structured queries have been recently proposed [1, 6, 22]. In addition to ranking the results of a query, there has been recent work [7] on ordering the displayed attributes of query results.

Both of these tuple and the attribute ranking techniques are inapplicable to our problem. The former inputs a database and a query, and outputs a list of database tuples according to a ranking function, and the latter inputs the list of database results and selects a set of attributes that “explain” these results. In contrast, our problem inputs a database, a query log, and a new tuple, and computes a set of attributes that will rank the tuple high for as many queries in the query log as possible.

Although the problem of choosing attributes is seemingly related to the area of *feature selection* [9], our work differs from the extensive body of work on feature selection because our goal is very specific – to enable a tuple to be highly visible to the users of the database – and not to reduce the cost of building a mining model such as classification or clustering.

Kleinberg et al. [15] present a set of microeconomic problems suitable for data mining techniques; however no specific solutions are presented. Their problem closer to our work is identifying the best parameters for a marketing strategy in order to maximize the attracted customers, given

that the competitor independently also prepares a similar strategy. Our problem is different since we know the competition (other data items). Another area where boosting an item's rank has received attention is Web search, where the most popular techniques involve manipulating the link-structure of the Web to achieve higher visibility [12].

Integer and linear programming optimization problems are extremely well studied problems in operations research, management science and many other areas of applicability (see recent book on this subject [20]). Integer programming is well-known to be NP-hard [10]; however carefully designed branch and bound algorithms can efficiently solve problems of moderate size. In our own experiments, we use an of-the-shelf ILP solver available from <http://lpsolve.sourceforge.net/5.5/download.htm>.

Computing frequent itemsets is a popular area of research in data mining and some of the best known algorithms include Apriori ([2]) and FP-Tree [14]. Several papers have also investigated the problem of computing maximal frequent itemsets [3, 4, 11, 13].

The recent works on dominant relationship analysis [16] and dominating neighborhood profitably [17] are related to our work. The former tries to find out the dominant relationship between products and potential buyers where by analyzing such relationships, companies can position their products more effectively while remaining profitable, and the latter introduces skyline query types taking into account not only min/max attributes (e.g., price, weight) but also spatial attributes (e.g., location attributes) and the relationships between these different attribute types. Their work aims at helping manufacturers choose the right specs for a new product, whereas our work aims at choosing the attributes subset of an existing product for advertising purposes.

VII. EXPERIMENTS

In this section we measure (a) the time cost of the proposed optimal and greedy algorithms, and (b) the approximation quality of the greedy algorithms. Due to lack of space we have chosen to present the results for the most representative problem variant, *SOC-CB-QL*. Due to space constraints we omit experiments on text, categorical and numeric data problem variants as they are adaptations of the algorithms used for *SOC-CB-QL*, as described in Section V.

System Configuration: We used Microsoft SQL Server 2000 RDBMS on a P4 3.2-GHZ PC with 1 GB of RAM and 100 GB HDD for our experiments. We implemented all algorithms in C#, and connected to the RDBMS through ADO.

Dataset: We use an *online used-cars dataset* consisting of 15,211 cars for sale in the Dallas area extracted from autos.yahoo.com. There are 32 Boolean attributes, such as AC, Power Locks, etc. We used a real workload of 185 queries created by users at UT Arlington, as well as a

synthetic workload of 2000 queries. In the synthetic workload, each query specifies 1 to 5 attributes chosen randomly distributed as follows: 1 attribute – 20%, 2 attributes – 30%, 3 attributes – 30%, 4 attributes – 10%, 5 attributes – 10%. That is, we assume that most of the users specify two or three attributes.

The top- m attributes selected by our algorithms seem promising. For example, even with a small real query log of 185 queries, our optimal algorithms could select top features specific to the car, e.g., sporty features are selected for sports cars, safety features are selected for passenger sedans, and so on.

We first compare the execution times of the optimal and greedy algorithms that solve *SOC-CB-QL*. These are (Section IV.A): *ILP-SOC-CB-QL*, *MaxFreqItemSets-SOC-CB-QL*, which produce optimal results, and *ConsumeAttr-SOC-CB-QL*, *ConsumeAttrCumul-SOC-CB-QL*, and *ConsumeQueries-SOC-CB-QL*, which are greedy approximations. The *SOC-CB-QL* suffix is skipped in the graphs for clarity.

Fig 6 shows how the execution times vary with m for the real query workload, averaged over 100 randomly selected to-be-advertised cars from the dataset. Note that different y-axis scales are used for the two optimal and the three greedy algorithms to better display the differences among themselves. We note that the *MaxFreqItemSets* algorithm consistently performs better than the *ILP* algorithm. Another interesting observation is that the cost of *ILP* does not always increase with m . The reason seems to be that the *ILP* solver is based on branch and bound, and for some instances the pruning of the search space is more efficient than for others.

The times in Fig 6 for *MaxFreqItemSets* also include the preprocessing stage, which can be performed once in advance regardless of the new tuple (user car), as explained in Section IV.C. If the pre-processing time is ignored, then *MaxFreqItemSets* takes only approximately 0.015 seconds to execute for any m value.

Fig 7 shows the quality, that is, the numbers of satisfied queries for the greedy algorithms along with the optimal numbers, for varying m . The numbers of queries are averaged over 100 randomly selected to-be-advertised cars from the dataset. Note that no query is satisfied for $m = 3$ because all queries specify more than 3 attributes. We see that *ConsumeAttr* and *ConsumeAttrCumul* produce near-optimal results. In contrast, *ConsumeQueries* has low quality, since it is often the case that the attributes of the queries with few attributes (which are selected first) are not common in the workload.

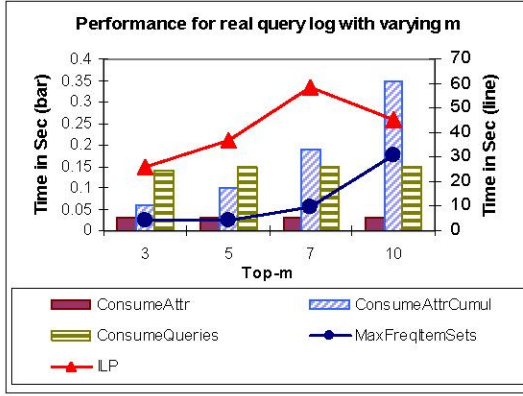


Fig 6 Execution times for SOC-CB-QL for varying m , for real workload.

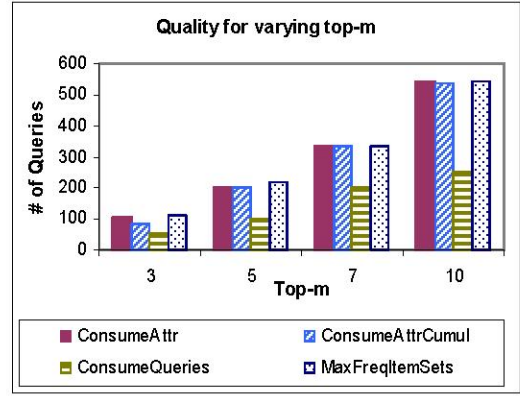


Fig 9 Satisfied queries for SOC-CB-QL for varying m , for synthetic workload of 2000 queries.

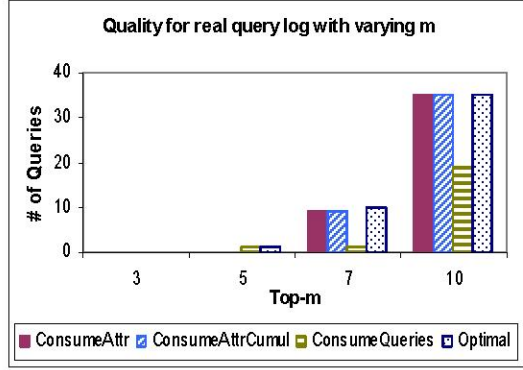


Fig 7 Satisfied queries for SOC-CB-QL for varying m , for real workload.

Fig 8 and Fig 9 repeat the same experiments for the synthetic query workload. In Fig 8, we do not include the *ILP* algorithm, because it is very slow for more than 1000 queries (as also shown in Fig 10).

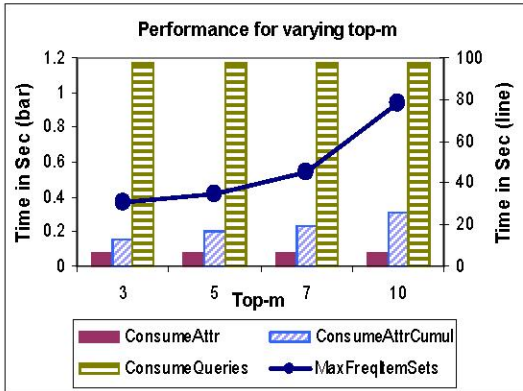


Fig 8 Execution times for SOC-CB-QL for varying m , for the synthetic workload of 2000 queries

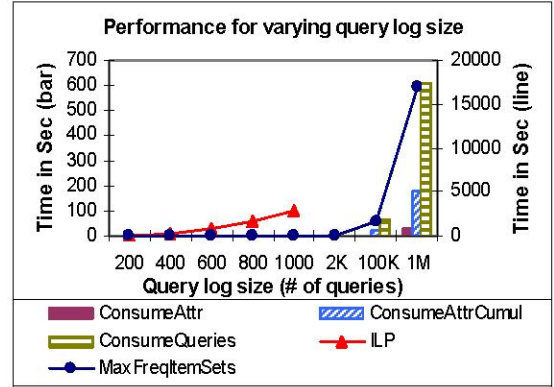


Fig 10 Execution times for SOC-CB-QL for varying query log size, varying synthetic workload size, $m = 5$

Fig 11 focuses on the two optimal algorithms, and measures the execution times of the algorithms, averaged over 100 randomly selected to-be-advertised cars from the dataset, for varying number M of total attributes of the dataset and queries, for a synthetic query log of 200 queries. We observe that *ILP* is faster than *MaxFreqItemSets* for more than 32 total attributes. For 32 total attributes *MaxFreqItemSets* is faster as also shown in Fig 6. However, note that *ILP* is only feasible for very small query logs. For larger query logs, *ILP* is very slow or infeasible, as is also shown by the missing values in Fig 10. To summarize, *ILP* is better for small query logs and

many total attributes (i.e. short and wide query log), whereas *MaxFreqItemSets* is better for larger query logs with fewer total attributes (i.e. long and narrow query log). However for query logs those are long as well as wide, the problem becomes truly intractable, and approximation methods such as our greedy algorithms are perhaps the only feasible approaches.

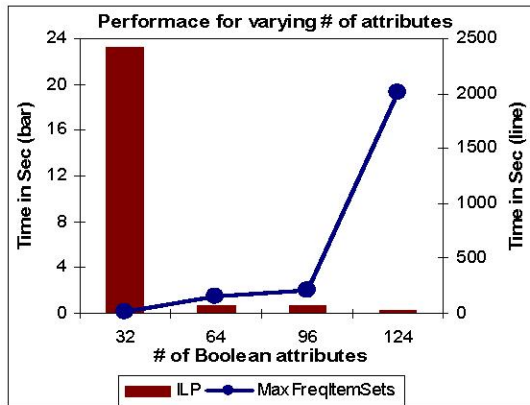


Fig 11 Execution times for SOC-CB-QL for varying M, synthetic workload of 200 queries, $m = 5$

VIII. CONCLUSIONS

In this work we introduced the problem of selecting the best attributes of a new tuple, such that this tuple will be ranked highly, given a dataset, a query log, or both, i.e., the tuple “stands out in the crowd”. We presented variants of the problem, and showed that even though the problem is NP-complete, optimal algorithms are feasible for small inputs. Furthermore, we present greedy algorithms, which are experimentally shown to produce good approximation ratios.

While the problems considered in this paper are novel and important to the area of ad-hoc data exploration and retrieval, we observe that our specific problem definition does have limitations. After all, a query log is only an approximate surrogate of real user preferences, and moreover in some applications neither the database, nor the query log may be available for analysis; thus we have to make assumptions about the nature of the competition as well as about the user preferences. Finally, in all these problems our focus is on deciding *what subset* of attributes to retain of a product. We do not attempt to suggest *what values* to set for specific attributes, which is a problem tackled in marketing research, e.g., [18].

However, while we acknowledge that the scope of our problem definition is indeed limited in several ways, we do feel that our work takes an important first step towards developing principled approaches for attribute selection in a data exploration environment. Overcoming the limitations mentioned above is subject of future work.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers, Stefan Schoenauer, and Valentin Polishchuk for useful comments. The work of Vagelis Hristidis was partially supported by National Science Foundation Grant IIS-0534530. The work of Gautam Das and Muhammed Miah was partially supported by unrestricted gifts from Microsoft Research and start-up funds from the University of Texas, Arlington. The work of Heikki Mannila was partially supported by the Academy of Finland.

REFERENCES

- [1] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, Aristides Gionis: Automated Ranking of Database Query Results. CIDR 2003.
- [2] R. Agrawal, and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. VLDB, 1994.
- [3] Roberto J. Bayardo Jr.: Efficiently Mining Long Patterns from Databases. SIGMOD Conference 1998: 85-93. 1
- [4] Douglas Burdick, Manuel Calimlim, Johannes Gehrke: MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. ICDE 2001
- [5] S. Brin and L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW Conference, 1998
- [6] S. Chaudhuri, G. Das, V. Hristidis, G. Weikum: Probabilistic Ranking of Database Query Results. VLDB, 2004
- [7] Gautam Das, Vagelis Hristidis, Nishant Kapoor, S. Sudarshan. Ordering the Attributes of Query Results. SIGMOD, 2006.
- [8] Good, I., The population frequencies of species and the estimation of population parameters, Biometrika, v. 40, 1953, pp. 237-264.
- [9] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. Journal of Machine Learning Research, 3(mar):1157-1182, 2003.
- [10] Michael R. Garey and David S. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman. ISBN 0-7167-1045-5.
- [11] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, R. S. Sharm: Discovering all most specific sentences. ACM TODS. 28(2): 140-174 (2003)
- [12] M. Gori and I. Witten. The bubble of web visibility. Commun. ACM 48, 3 (Mar. 2005), 115-117
- [13] Karam Gouda, Mohammed J. Zaki: Efficiently Mining Maximal Frequent Itemsets, ICDM 2001
- [14] Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. SIGMOD 2000: 1-12.
- [15] J. Kleinberg, C. Papadimitriou and P. Raghavan. A Microeconomic View of Data Mining. Data Min. Knowl. Discov. 2, 4 (Dec. 1998), 311-324
- [16] Cuiping Li, Beng Chin Ooi, Anthony K. H. Tung, Shan Wang: DADA: a Data Cube for Dominant Relationship Analysis. SIGMOD Conference 2006: 659-670
- [17] Cuiping Li, Anthony K. H. Tung, Wen Jin, Martin Ester: On Dominating Your Neighborhood Profitably. VLDB 2007: 818-829
- [18] Thomas T. Nagle, John Hogan. The Strategy and Tactics of Pricing: A Guide to Growing More Profitably (4th Edition), Prentice Hall, 2005
- [19] S E Robertson and S Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. SIGIR 1994
- [20] Alexander Schrijver: Theory of Linear and Integer Programming. John Wiley and Sons. 1998
- [21] G. Salton. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison Wesley, 1989
- [22] W. Su, J. Wang, Q. Huang, F. Lochovsky. Query Result Ranking over E-commerce Web Databases. ACM CIKM 2006

Ensemble-Trees: Leveraging Ensemble Power inside Decision Trees

Albrecht Zimmermann

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan
200A, 3001 Leuven, Belgium
`Albrecht.Zimmermann@cs.kuleuven.be`

Abstract. Decision trees are among the most effective and interpretable classification algorithms while ensembles techniques have been proven to alleviate problems regarding over-fitting and variance. On the other hand, decision trees show a tendency to lack stability given small changes in the data, whereas interpreting an ensemble of trees is challenging. We propose the technique of *Ensemble-Trees* which uses ensembles of rules *within* the test nodes to reduce over-fitting and variance effects. Validating the technique experimentally, we find that improvements in performance compared to ensembles of pruned trees exist, but also that the technique does less to reduce structural instability than could be expected.

1 Introduction

Decision trees have been a staple of classification learning in the machine learning field for a long time [1–3]. They are efficiently inducible, relatively straightforward to interpret, generalize well, and show good predictive performance. All of these benefits, however, are diminished by the fact that small changes in the data can lead to very different trees, over-fitting effects, and varying accuracies when employed to classify unseen examples.

To deal with problems in classification learning such as over-fitting, ensemble methods have been proposed. While the details vary, the main mechanism remains the same: a so-called “weak” learner, such as the afore-mentioned decision trees, is trained on a particular sample distribution of the available training data. The subset is then altered, possibly according to the performance of the learned classifier, and another classifier trained. In the end, an ensemble of weak classifiers is combined to classify unseen examples. The use of ensembles results in more stability w.r.t. predictive accuracy since over-fitting effects of several classifiers are balanced against each other. This advantage is however traded off both against difficulties with interpreting the complete classifier, since it can easily number tens of weak classifiers, and of course against increased training times.

To improve on the weaknesses of both decision trees and ensembles, we propose to essentially invert the ensemble construction process: Instead of inducing

complete classical decision trees on subsets of the data and combining those into a classifier, our solution – *Ensemble-Trees* – induces small ensembles of rules as tests in each inner node. The expected benefits would be relatively small and interpretable trees compared to ensembles of decision trees, and better structural stability w.r.t. changes in the data and higher predictive accuracy compared to classical decision trees.

The paper is structured as follows: in the following section, we will develop our approach, going over the main details of decision tree induction, k -best rule mining, and combination of rule predictions in the final tree. In Section 3, the connection to existing works will be made, followed by an experimental evaluation of our hypotheses regarding the behavior of *Ensemble-Trees* in Section 4. Finally, we discuss the results of this evaluation, and conclude, in Section 5.

2 From Decision Trees to *Ensemble-Trees*

Decision trees in different forms [1–3] have been one of the biggest success stories of classification learning, with Quinlan’s C4.5 one of the most effective symbolic classification algorithms to date. Even when adapted to allow for regression or clustering, the main algorithm for inducing a decision tree stays more or less the same: For a given subset of the data, a “best” test according to an interestingness measure is selected, the data split according to this test, and the process repeated on the two subsets thus created. The interestingness measures used (e.g. information gain, intra-class variance) typically reward tests that make the subsets more homogeneous w.r.t. a target variable (e.g. the class label, or a numerical value). In general, a test can have any number of outcomes but for reasons that will be discussed later we limit ourselves to binary tests. The pseudo-code of a decision tree induction algorithm is shown as Algorithm 1.

Algorithm 1 The General Decision Tree Algorithm

Input: a data set $\mathcal{D}$, interestingness measure σ , minimum leaf size m

Output: a decision tree T

$T = \text{make\_node}(\mathcal{D}, \sigma, m)$

return T

$\text{make\_node}(\mathcal{D}, \sigma, m)$

Find best test t on $\mathcal{D}$ according to σ

$\mathcal{D}_{yes} = \{d \in \mathcal{D} \mid t \text{ matches } d\}$

$\mathcal{D}_{no} = \{d \in \mathcal{D} \mid t \text{ does not match } d\}$

if $|\mathcal{D}_{yes}| \geq m \wedge |\mathcal{D}_{no}| \geq m$ **then**

$n_{yes} = \text{make\_node}(\mathcal{D}_{yes}, \sigma, m)$

$n_{no} = \text{make\_node}(\mathcal{D}_{no}, \sigma, m)$

return $\text{node}(t, n_{yes}, n_{no})$

else

return $\text{leaf}(\text{prediction}(\mathcal{D}))$

end if

Note that the best test as well as the prediction given in a leaf is not specifically instantiated in this pseudo-code and depends on the data mining task at hand. In the popular C4.5 approach, the leaf prediction is the majority class in $\mathcal{D}$, and a binary test takes the form of the attribute-value pair that leads to the largest increase in homogeneity, measured by information gain. This means, however, that small changes in the data can have a strong effect on the composition of the tree. Consider σ to be information gain, and a data set with the following characteristics:

index	A_1	A_2	...	class
1	v_2	v_1	...	+
2	v_1	v_2	...	-
3	v_2	v_1	...	+
4	v_1	v_1	...	+
5	v_1	v_2	...	-
6	v_1	v_2	...	-
7	v_1	v_1	...	-
8	v_2	v_1	...	+
...	...	...	...	...

Obviously, a change in either the value of A_1 in instance 4, or the value of A_2 in instance 7 would improve the strength of these attributes as a test, respectively. Similarly, removal of one of those two instances, e.g. as part of a cross-validation, changes which attribute is chosen. Since the choice of test affects how the data is split, this effect ripples down through the rest of the tree. Assuming that A_1 and A_2 perform equally well on the rest of the data, the decision between them comes down to an arbitrary tie-breaker, giving the tree potentially rather different composition – and performance.

While attempts have been made to alleviate this problem, using linear combinations of attributes-value pairs in nodes [4], or *option trees* [5], which in case of tests with very similar performance keep *all* of them and sort the data down all paths, these techniques in fact strongly decrease comprehensibility of the tree, trading it off against better performance.

2.1 Ensembles of Rules and Efficient k -best induction

Ensembles of conjunctive rules, used together with conflict-resolution techniques such as (weighted) voting [6], or double induction [7] have been empirically shown to perform better than decision lists of ordered rules, balancing the bias/overfitting of single rules.

If this stabilizing effect of using a (small) ensemble of rules is brought to bear for tests inside a decision tree node, we expect that splits effected by those ensembles are less susceptible to changes in the data than splits effected by individual tests. This in turn should lead to trees having rather similar composition and structure, and ideally also similar rule ensembles in the test nodes.

Additionally, overfitting effects of individual tests that affect the accuracy can be expected to be reduced, potentially removing the need for post-pruning

of the induced tree. Using conjunctive rules as tests always leads to binary splits, since a conjunction can be either *true* or *false*, thus the formulation of a test as binary in the general algorithm.

Using a branch-and-bound search based on the pruning techniques for convex interestingness measures (such as information gain, and intra-class variance) pioneered by Morishita and Sese [8], it is possible to efficiently induce the k best conjunctive rules w.r.t. the measure used. The branch-and-bound technique is based on calculating an upper bound for each rule encountered during enumeration. Only rules whose upper bound exceeds the k th-best score at the time of potential specialization *are* actually specialized. For further details, we refer the reader to Morishita and Sese’s work.

Thus, the generic “Find best test” can be instantiated by “Find the k best rules”. Since conjunctive rules induced in this way are quantified w.r.t. a *target value* such as the class label, the split becomes more complex than the straight-forward “match” used in the general algorithm. We will discuss the majority voting technique used in our approach in the next section.

2.2 Majority Splitting

Splitting the data according to a single attribute-value pair is straight-forward in that the pair either matches an instance, or does not match it, and the class prediction is implicit in how the distribution of class labels changes in the subsets thus produced. We adopt the convention that the left branch of a test node is the “yes” branch, i.e. instances that are matched by a test are sorted to the left, unmatched instances to the right.

Once several tests are used to split the data, the situation becomes a bit more complex however. Consider the following three rules for a binary prediction problem:

1. $A_1 = v_1 \wedge A_2 = v_1 \Rightarrow +$
2. $A_3 = v_1 \wedge A_4 = v_1 \Rightarrow -$
3. $A_1 = v_1 \wedge A_5 = v_2 \Rightarrow -$

and a subset of instances:

index	A_1	A_2	A_3	A_4	A_5	class
1	v_1	v_1	v_2	v_1	v_1	+
2	v_1	v_1	v_1	v_1	v_1	+
3	v_1	v_2	v_1	v_1	v_2	−
4	v_1	v_1	v_1	v_2	v_2	−
5	v_2	v_1	v_2	v_1	v_1	+

Considering the first instance, we see that rule 1 matches, therefore advocating that the instance be sorted to the *left*. Rules 2 and 3 do not match, thus advocating to sort the instance to the *right*. Since they form the majority, the instance would be sorted to the right. However, their prediction is *opposite* to the prediction of the first rule. Therefore, the match/non-match of rules 1, and

rules 2 and 3 have a different semantic interpretation. In the interest of making class distribution more homogeneous to facilitate classification, it is intuitive to interpret the non-match of rules 2 and 3 as in effect advocating that instance 1 be sorted to the *left*. All three rules are then in agreement and instance 1 is sorted to the left.

The convention that matched instances are sorted to the left therefore is transformed into the convention that instance that are matched by rules *that predict the first class value* (or not matched by rules predicting the second class label) are sorted to the left, while in the opposite case being sorted to the right.

Looking at instance 2, we observe that it is matched by both rules 1 and 2, garnering one vote for being sorted into either direction. Rule 3 acts as a tie-breaker in this case, leading to instance 2 being sorted left. This example illustrates that the k that governs the amount of rules used in a test node should be odd so that ties do not have to be broken arbitrarily.

2.3 Inducing Ensemble Trees

Having covered the basic ingredients for inducing *Ensemble-Trees* in the preceding sections, Algorithm 2 shows the pseudo-code of the complete algorithm.

Algorithm 2 The Ensemble Tree Algorithm

Input: data set $\mathcal{D}$, interestingness measure σ , minimum leaf size m , number of rules k

Output: an Ensemble Tree T

$T = \text{make\_node}(\mathcal{D}, \sigma, m, k)$

return T

make_node($\mathcal{D}, \sigma, m, k$)

Attempt to find k best conjunctive rules $\mathcal{R} := \{r_i\}$ on $\mathcal{D}$ according to σ

if $|\mathcal{R}| < k$ **then**

return majority class in $\mathcal{D}$

end if

$\mathcal{D}_{\text{left}} = \{d \in \mathcal{D} \mid \text{at least } \lceil k/2 \rceil \text{ rules vote } d \text{ is sorted left}\}$

$\mathcal{D}_{\text{right}} = \{d \in \mathcal{D} \mid \text{at least } \lceil k/2 \rceil \text{ rules vote } d \text{ is sorted right}\}$

if $|\mathcal{D}_{\text{left}}| \geq m \wedge |\mathcal{D}_{\text{right}}| \geq m$ **then**

$n_{\text{left}} = \text{make\_node}(\mathcal{D}_{\text{left}}, \sigma, m, k)$

$n_{\text{right}} = \text{make\_node}(\mathcal{D}_{\text{right}}, \sigma, m, k)$

return $\text{node}(\mathcal{R}, n_{\text{left}}, n_{\text{right}})$

else

return $\text{leaf}(\text{majority\_class}(\mathcal{D}))$

end if

Note, that the Ensemble Tree algorithm includes another pre-pruning/stopping criterion: If it is not possible to find the user-specified amount of rules, not enough

different splitting tests improving homogeneity can be induced, and the node is turned into a leaf. This can occur since typically k -best interestingness mining rejects rules which have the same score as one of their generalizations, since this translates into conjuncts that do not add any information. Similarly, and additionally, to the minimum size of a leaf m , this criterion pre-prunes test nodes that are not supported by enough data to be expected to generalize well.

3 Related Approaches

As mentioned above, *Ensemble-Trees* can be considered as extensions of classical decision trees such as the trees induced by the C4.5 algorithm, in that the main difference lies in the replacement of the attribute-value pairs in test nodes by small ensembles of conjunctive rules. Our approach is not the first to attempt and modify the tests in nodes however.

3.1 Decision Trees With Complex Tests

To our knowledge the first that recognized the problem of similarly performing tests, and proposed a solution in the form of *option trees* were Kohavi and Kunz [5]. Their solution consists of not choosing one out of several tests arbitrarily but instead creating “option nodes” having children nodes including all of these tests, and splitting the subset at each node. For classification, unseen examples are propagated downwards through the tree, and in case of an option node, the label predicted by the majority of child nodes chosen. While their approach improved significantly on the accuracy of C4.5 and an ensemble of bagged trees, the option trees themselves turned out to easily consist of one thousand nodes or more, comparable to the accumulated size of 50 bagged trees.

In [4], the authors suggest an algorithm consisting of a mixture of uni-variate (classical) decision tree tests, linear and non-linear combinations of attributes, using neural networks. The resulting trees are reported to be smaller than purely uni-variate trees and to improve on the accuracy in several cases. It seems obvious however, that such combinations of attributes will be rather hard to interpret for end users.

In [9] and [10], data mining techniques were employed to mine complex fragments in graph- or tree-structured class-labeled data, which are then used as tests in a decision tree classifier. The motivation in these works is similar to ours in that the need for complex tests is acknowledged. Both techniques limit themselves to a single test in each node, however, not attempting to correct over-fitting effects during the induction of the trees.

3.2 Ensembles of Trees

Ensemble methods have been used with decision trees before, with the trees acting as the weak classifiers in the ensemble scheme. One of the best-known ensemble techniques, BAGGING [11], was proposed specifically with decision trees

as weak classifiers. In bagging, repeated boot-strap sampling of a data set is performed, decision trees learned on each of these samples, and their predictions combined using a majority vote. While BAGGING helps in reducing over-fitting effects, the fact that instead of one tree several are created can be an impediment to interpreting the resulting classifiers.

BOOSTING, as embodied for instance by the well-known ADABOOST system [12] on the other hand, iteratively induces weak classifiers on data sets whose distribution is altered according to the performance of preceding classifiers. Mostly, this takes the form of resampling misclassified instances, or re-weighting them. While boosting technique can be proven to approximate the actual underlying classification function to an arbitrary degree, the resulting ensemble is again rather difficult to interpret, especially given the changes in the underlying distribution that are effected during the learning process.

4 Experimental Evaluation

An experimental evaluation is used to answer

Q1 How well *Ensemble-Trees* are suited to the classification task, especially in the absence of post-pruning.

Q2 Whether the use of ensembles of rules as tests in the inner nodes leads to more stable trees in the presence of changes in the underlying data, and whether the resulting trees are smaller (and better interpretable) than ensembles of trees.

We use several UCI data sets to compare *Ensemble-Trees* to the WEKA [13] implementations of C4.5, BAGGING, and ADABOOST, each with C4.5 as the weak classifier.

Since we limit ourselves to nominal attribute values in this work, numerical attributes were discretized, using ten bins of equal width. With the branch-and-bound technique used for inducing the rule ensembles being limited to binary classes, we used only data sets with binary class labels. However, given that every classification problem can be translated into a number of *one-against-one*, or *one-against-all* problems, we do not see this as a significant drawback of our approach.

We performed experiments with the minimum leaf size parameter m set to 2, 3, 4, 5, 10 and in case of large data sets, 10% of the training data. As the interestingness measure σ , information gain was used. BAGGING and ADABOOST were set to 10 iterations of inducing weak classifiers, and we built *Ensemble-Trees* with $k = 3$, and $k = 5$, respectively. ADABOOST was used both in the resampling and the re-weighting mode. Decision tree and ensemble method results are reported on pruned trees while *Ensemble-Trees* are always unpruned.

4.1 Predictive Accuracy

Predictive accuracy was evaluated using a class-validated 10-fold cross-validation. Table 1 reports mean and standard deviation for a minimum leaf size $m = 5$, except for the Trains data set, where $m = 2$. While details vary, the main trends can be observed for all minimum leaf sizes evaluated in the experiments.

Table 1. Predictive accuracies for decision trees/*Ensemble-Trees* with minimum leaf size of 5

Data set	C4.5	ET _{k=3}	ET _{k=5}	BAGGING	ADABOOST _{RS}	ADABOOST _{RW}
Breast-Cancer	73.42 ± 5.44●	78.69 ± 4.34	80.14 ± 6.16	73.77 ± 6.98	67.09 ± 10.10●	66.77 ± 6.81●
Breast-Wisconsin	94.56 ± 2.93	95.28 ± 1.35	95.14 ± 1.80	95.42 ± 2.76	96.28 ± 1.81	95.99 ± 1.14
Credit-A	84.20 ± 2.93	85.51 ± 2.56	85.51 ± 2.56	85.22 ± 2.35	82.75 ± 3.45	82.03 ± 4.44●
Credit-G	71.90 ± 3.96●	80.33 ± 2.00	79.10 ± 5.09	74.40 ± 4.06●	72.60 ± 3.24●	70.30 ± 4.00●
Heart-Statlog	82.96 ± 8.04	81.85 ± 5.08	79.63 ± 6.82	80.74 ± 8.52	80.37 ± 7.82	78.52 ± 7.57
Hepatitis	84.50 ± 6.22	89.58 ± 5.61	90.92 ± 5.54	83.25 ± 5.35●	83.17 ± 7.07●	82.71 ± 8.27●
Ionosphere	88.60 ± 5.88	91.44 ± 3.82	88.92 ± 5.80	91.15 ± 4.37	90.87 ± 5.69	91.15 ± 6.25
Molec. Biol. Prom.	78.09 ± 14.57	83.73 ± 9.28	83.64 ± 11.36	88.00 ± 13.00	85.73 ± 10.96	88.64 ± 5.94
Mushroom	100 ± 0○	99.95 ± 0.06	99.95 ± 0.06	96.31 ± 5.94	100 ± 0○	100 ± 0○
Tic-Tac-Toe	91.76 ± 3.81○	76.20 ± 1.47	72.86 ± 4.59	96.87 ± 1.55○	98.02 ± 1.59○	96.97 ± 2.62○
Trains	60.00 ± 51.64	50.00 ± 52.70	50.00 ± 52.70	40.00 ± 51.64	80.00 ± 42.16	50.00 ± 52.70
Voting-Record	95.85 ± 2.83●	98.39 ± 1.11	98.62 ± 1.19	95.62 ± 2.29●	94.94 ± 3.04●	96.08 ± 3.09●

The table shows that *Ensemble-Trees* perform mostly well w.r.t. classification. In several cases, *Ensemble-Trees* are significantly better than ensemble methods (● denotes a significant loss of a technique at the 5%-level), while being outperformed only on Mushroom (barely), and Tic-Tac-Toe (○ denotes a significant win at the 5%-level).

Inspection of the standard deviations, however, gives a first indication that *Ensemble-Trees* do not turn out to be more stable performance-wise when the data composition changes. In fact, *Ensemble-Trees* using only three rules per test-node ensemble show *less* standard deviation, i.e. less variance, w.r.t. accuracy than *Ensemble-Trees* with larger ensembles.

4.2 Size and Stability of Induced Trees

The second question we evaluated was concerned with stability of induced trees w.r.t. changes in the data, and the comprehensibility of *Ensemble-Trees* and ensembles of trees, respectively. We used the 10-fold cross-validation mechanism already employed in the accuracy estimation to simulate changes in the underlying data, and report on size characteristics of the trees in Tables 2 and 3. For C4.5 decision trees and *Ensemble-Trees*, we report the mean and standard deviation of sizes (number of nodes) and maximal depths, i. e. length of the longest branch, of the different trees. Since the ensemble methods induce a different tree in each iteration, averaging over the iterations *and* the folds becomes a difficult endeavor, and we report on the accumulated number of nodes (among all trees per fold).

Inspection of Table 2 shows that *Ensemble-Trees* always have a lot less nodes than ensembles of trees. Even if the accumulated sizes are normalized by dividing by the number of trees, this difference is still pronounced in most cases. Given that the ensemble methods do not improve markedly on the accuracy of *Ensemble-Trees* in most cases, quite a few more bags/iterations would be needed for better performance (except for the Mushroom data set), in turn leading to even less comprehensible final classifiers.

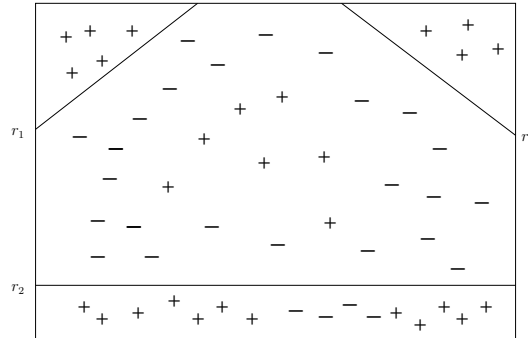
Table 2. Number of nodes per tree for C4.5/*Ensemble-Trees*, accumulated number of nodes for all trees of the respective ensembles

Data set	C4.5	ET _{k=3}	ET _{k=5}	BAGGING	ADABOOST _{RS}	ADABOOST _{RW}
Breast-Cancer	13.6 ± 6.4	7.6 ± 3.5	9.4 ± 5.1	425.4 ± 19.4	485.2 ± 18.2	509.4 ± 14.7
Breast-Wisconsin	14.8 ± 1.5	13.4 ± 2.8	9.0 ± 2.1	154.2 ± 12.2	334.4 ± 12.2	331.8 ± 15.4
Credit-A	19.2 ± 4.2	16.4 ± 6.2	10.6 ± 6.1	226.6 ± 8.5	698.6 ± 17.0	725.2 ± 30.4
Credit-G	86.4 ± 9.2	22.8 ± 6.1	18.8 ± 8.9	896.0 ± 23.7	1212.2 ± 25.7	1239.2 ± 27.9
Heart-Statlog	14.4 ± 2.3	11.2 ± 2.2	10.8 ± 3.9	193.2 ± 14.5	302.6 ± 14.3	310.0 ± 14.5
Hepatitis	6.2 ± 3.0	10.6 ± 4.1	9.2 ± 3.0	86.6 ± 10.4	153.6 ± 7.9	165.4 ± 7.9
Ionosphere	17.2 ± 3.7	12.2 ± 6.0	10.4 ± 4.0	165.6 ± 7.6	243.4 ± 8.9	253.0 ± 9.2
Molec. Biol. Prom.	11.6 ± 1.0	5.4 ± 0.8	6.4 ± 1.3	98.2 ± 8.3	83.4 ± 3.9	85.8 ± 6.0
Mushroom	17	17	21	153.8 ± 5.6	17	169.4 ± 15.0
Tic-Tac-Toe	53.4 ± 2.8	3	2.8 ± 0.6	573.4 ± 9.2	700.8 ± 31.4	736.6 ± 23.7
Trains	3	3	3	30.2 ± 0.6	23.5 ± 7.5	28.4 ± 0.8
Voting-Record	8.2 ± 1.0	13.0 ± 3.4	13.2 ± 4.0	75.6 ± 6.7	184.2 ± 18.0	181.6 ± 15.7

Ensemble-Trees also typically have somewhat fewer nodes than classical C4.5 decision trees, due to the more expressive tests in the nodes. The cases where this trend does not hold (the Hepatitis, and Voting-Record data sets) or is exaggerated (the Breast-Cancer, and Credit-G data sets) are also the ones where *Ensemble-Trees* perform best compared to the other approaches. However, in the case of the Tic-Tac-Toe data set, the small number of nodes is actually a symptom of an underlying characteristic, with another being the atrocious performance of *Ensemble-Trees*, compared to the other methods.

To explain the mechanism at work here, Figure 1 shows binary class data points in two-dimensional space, and the decision surfaces of three rules r_1, r_2, r_3 . As can be seen, each of these rules predicts the positive class, advocating that

Fig. 1. Binary class data and decision surfaces of three discriminatory rules



the instances they cover be sorted to the left. Additionally, however, all of them advocate the sorting of the instances covered by the other two rules to the right. The result of the majority vote on this is that all instances are sorted to the right, the left subset is empty and thus its size less than m , leading to the

formation of a leaf. The rather aggressive pre-pruning effected by this stopping criterion becomes problematic on data sets with small, non-overlapping sub-regions in the data. Tic-Tac-Toe is such a data set, as indicated by the rather large number of nodes (and therefore leaves) in – pruned – C4.5 decision trees. The expected stabilization of trees w.r.t. changes in the data, however, cannot be

Table 3. Averaged maximal depths for C4.5 trees/*Ensemble-Trees*

Data set	C4.5	ET _{k=3}	ET _{k=5}
Breast-Cancer	4.1 ± 1.5	2.90 ± 1.2	3.5 ± 2.2
Breast-Wisconsin	4.4 ± 0.5	4.2 ± 0.9	3.9 ± 0.9
Credit-A	6.7 ± 1.3	5.8 ± 2.7	4.0 ± 2.7
Credit-G	23.9 ± 2.3	7.9 ± 2.1	18.8 ± 8.9
Heart-Statlog	3.2 ± 0.8	3.8 ± 0.8	3.6 ± 1.0
Hepatitis	1.1 ± 1.5	4.7 ± 2.0	4.0 ± 1.3
Ionosphere	6.9 ± 1.6	5.6 ± 3.0	4.6 ± 1.8
Molec. Biol. Prom.	2.8 ± 1.0	2.2 ± 0.4	2.7 ± 0.7
Mushroom	4	5	5
Tic-Tac-Toe	6.0	1	0.9 ± 0.3
Trains	1	1	1
Voting-Record	2.6 ± 0.5	4.2 ± 0.8	4.4 ± 1.1

observed. Neither in the number of nodes, nor in the maximal depths of trees do *Ensemble-Trees* markedly decrease the standard deviation over folds, compared to C4.5 pruned trees. Quite contrary, while the trees are shallower, and mostly smaller, *Ensemble-Trees* often show greater variance in both characteristics than classical decision trees do. Since the size measurements were extracted after

Table 4. Number of nodes per tree for unpruned C4.5/*Ensemble-Trees*

Data set	C4.5	ET _{k=3}	ET _{k=5}
Breast-Cancer	41.0 ± 8.7	7.6 ± 3.5	9.4 ± 5.1
Breast-Wisconsin	19.4 ± 4.9	13.4 ± 2.8	9.0 ± 2.1
Credit-A	35.8 ± 11.2	16.4 ± 6.2	10.6 ± 6.1
Credit-G	147.0 ± 10.1	22.8 ± 6.1	18.8 ± 8.9
Heart-Statlog	26.8 ± 3.3	11.2 ± 2.2	10.8 ± 3.9
Hepatitis	24.8 ± 2.6	10.6 ± 4.1	9.2 ± 3.0
Ionosphere	17.2 ± 3.7	12.2 ± 6.0	10.4 ± 4.0
Molec. Biol. Prom.	11.6 ± 1.0	5.4 ± 0.8	6.4 ± 1.3
Mushroom	21	17	21
Tic-Tac-Toe	68.2 ± 7.3	3	2.8 ± 0.6
Trains	3	3	3
Voting-Record	8.8 ± 1.1	13.0 ± 3.4	13.2 ± 4.0

post-pruning, we finally compare unpruned C4.5 trees to *Ensemble-Trees* in an attempt to understand how much of a stabilization effect post-pruning provides

to the decision trees. The relevant numbers are shown in Tables 4 and 5, with the characteristics of *Ensemble-Trees* duplicated from Tables 2 and 3.

While Table 4 suggests that the reduction in variance for the size of a decision tree is a result of the post-pruning operation, and Table 5 shows similar effects regarding maximal depths of the tree, this still means that *Ensemble-Trees* are structurally not more stable w.r.t. changes in the data than classical decision trees. While the use of ensembles of conjunctive rules as tests could bring more stability to the final tree, this promise remains unfulfilled. A potential improvement could lie in a better solution for the combination strategy, an issue we will investigate in future work.

Table 5. Averaged maximal depths for unpruned C4.5 trees/*Ensemble-Trees*

Data set	C4.5	ET _{k=3}	ET _{k=5}
Breast-Cancer	9.5 ± 2.4	2.9 ± 1.2	3.5 ± 2.2
Breast-Wisconsin	5.7 ± 1.1	4.2 ± 0.9	3.9 ± 0.9
Credit-A	11.3 ± 2.3	5.9 ± 2.7	4.0 ± 2.7
Credit-G	26.1 ± 2.3	7.9 ± 2.1	18.8 ± 8.9
Heart-Statlog	5.2 ± 1.0	3.8 ± 0.8	3.6 ± 1.0
Hepatitis	4.1 ± 0.9	4.7 ± 2.0	4.0 ± 1.3
Ionosphere	10.4 ± 1.2	5.6 ± 3.0	4.6 ± 1.8
Molec. Biol. Prom.	2.8 ± 1.0	2.2 ± 0.4	2.7 ± 0.7
Mushroom	6	5	5
Tic-Tac-Toe	6.6 ± 0.7	1	0.9 ± 0.3
Trains	1	1	1
Voting-Record	2.9 ± 0.6	4.2 ± 0.8	4.4 ± 1.1

A final subject that needs to be addressed is the interpretability of found solutions. Classical decision trees and ensemble methods are the borders of an interval that ranges from

- Easy (decision trees): paths are interpreted as conjunctions, tests as disjunctions, via
- Challenging (BAGGING): individual trees are supposed to describe broadly the same phenomena, with the final classifier being an average of all trees, to
- Hard (BOOSTING): individual trees model local phenomena, with the final classifier being a weighted combination,

with *Ensemble-Trees* residing somewhere on the easier side of BAGGING. While harder to interpret than classical decision trees, the interpretation of a path being a conjunction of test-nodes still holds. At the same time, conjunctive rules in the nodes themselves have a clear semantic, and depending on the combination scheme, the entire rule ensemble (especially since it will be small) should prove not to be too hard to comprehend.

5 Conclusion

In this work we develop the classification technique of *Ensemble-Trees*, an attempt to leverage the power of ensemble techniques in dealing with over-fitting and variance effects, while retaining as much of the easy interpretability of decision trees as possible.

Our experimental evaluation showed that over-fitting effects seem to be prevented more efficiently, leading to better classification accuracy in several cases. The structural stabilization of individual trees that seems possible if more complex tests are used that balance each other's bias, does not materialize however.

While the advantages of *Ensemble-Trees* are clear – no need for post-pruning, shallower trees with fewer nodes, and far easier interpretability than existing ensemble methods using decision trees as weak classifiers, they come with a caveat. There exist data sets where the aggressive pre-pruning that results from the use of ensembles of rules is detrimental to the performance of the classifier. Potential remedies, such as changing the combination strategy, or dynamically adjusting the k parameter, will be explored in future work, with the expectation that this will aid in realizing the full promise of this technique w.r.t. structural stabilization against changes in the data.

References

1. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman & Hall, New York (1984)
2. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
3. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artif. Intell. **101** (1998) 285–297
4. Murthy, S.K.: On Growing Better Decision Trees from Data. PhD thesis, John Hopkins University, Baltimore, Maryland, USA (1997)
5. Kohavi, R., Kunz, C.: Option decision trees with majority votes. In Fisher, D.H., ed.: ICML, Morgan Kaufmann (1997) 161–169
6. Zaki, M.J., Aggarwal, C.C.: XRules: an effective structural classifier for XML data. In Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C., eds.: KDD, Washington, DC, USA, ACM (2003) 316–325
7. Lindgren, T., Boström, H.: Resolving rule conflicts with double induction. In: 5th International Symposium on IDA, Berlin, Germany, Springer (2003) 60–67
8. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: PODS, Dallas, Texas, USA, ACM (2000) 226–236
9. Geamsakul, W., Matsuda, T., Yoshida, T., Motoda, H., Washio, T.: Performance evaluation of decision tree graph-based induction. In Grieser, G., Tanaka, Y., Yamamoto, A., eds.: Discovery Science, Sapporo, Japan, Springer (2003) 128–140
10. Bringmann, B., Zimmermann, A.: Tree² - decision trees for tree structured data. In Jorge, A., Torgo, L., Brazdil, P., Camacho, R., Gama, J., eds.: PKDD, Springer (2005) 46–58
11. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
12. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55** (1997) 119–139
13. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)

Optimal Constraint-Based Decision Tree Learning

Siegfried Nijssen, Elisa Fromont

K.U.Leuven, Dept. of Computer Science, Celestijnenlaan 200A, B-3001 Leuven,
e-mail: {Siegfried.Nijssen, Elisa.Fromont}@cs.kuleuven.be

Received: date / Revised version: date

Abstract We present DL8, an exact algorithm for learning a decision tree that optimizes a ranking function under a wide number of constraints, among which size, depth, accuracy and leaf constraints. To the best of our knowledge, DL8 is the first algorithm that is in practice capable of performing this task. Because the discovery of optimal trees has high theoretical complexity, until now few efforts have been made to compute such trees exactly for real-world datasets. Instead, it is common practice to develop heuristics algorithms. Compared to other algorithms DL8 has the following benefits. First, the algorithm is general enough to solve a large range of learning problems, from privacy preserving decision tree learning, cost-based decision tree learning, to Bayesian tree induction. Our framework allows us to combine these learning problems without requiring the ad-hoc development of new heuristics to solve these problems. Second, the algorithm can be used as a gold standard to evaluate the performance of heuristic constraint-based decision tree learners and to gain new insight in traditional decision tree learners. Third, from the application point of view, it can be used to discover trees that cannot be found by heuristic decision tree learners. The key idea behind our algorithm is that there is a relation between constraints on decision trees and constraints on itemsets. We show that optimal decision trees can be extracted from lattices of itemsets. We give several strategies to efficiently build these lattices. Experiments show that under the same constraints, DL8 obtains better results than C4.5 and cost-based decision tree learners, which confirms that exhaustive search does not always imply overfitting, and shows that DL8 is practically relevant.

Key words Decision tree learning, Formal concepts, Frequent itemset mining, Constraint based mining

1 Introduction

Decision trees are among the most popular predictive models in machine learning and have been studied from many perspectives. Indeed, many decision tree learning problems have been proposed, many algorithms have been studied and their behavior has been investigated. In this article, we study a novel framework that encapsulates a large number of learning problems, and we present an algorithm that can be used to solve these learning problems exactly.

Our main assumption is that many decision tree learning problems can be formulated as *queries* of the following canonical form:

$$\operatorname{argmin}_{T, \varphi(T)} f(T), \quad (\text{Canonical Decision Tree Learning Query})$$

i.e, we are interested in finding the best tree according to the function $f(T)$, among all trees which fulfil the constraints specified in the formula $\varphi(T)$. For instance, we may be interested in:

- finding a tree with small error; in this case $f(T)$ is an error function that we wish to minimize;
- finding small trees; in this case the (ranking) function $f(T)$ should prefer smaller trees among sets of equally accurate trees; for instance, $f(T)$ could be the posterior probability of a tree given a Bayesian prior that prefers small trees (Buntine, 1992; Chipman, George, & McCulloch, 1998);
- finding trees that do not overfit, which could require that every leaf of a decision tree has at least a certain number of examples; this can be seen as a constraint $\varphi(T)$ on the trees of interest;
- inducing a privacy preserving decision tree in which every split in a tree is well-balanced, thus constraining the trees $\varphi(T)$ of interest (Friedman, Schuster, & Wolff, 2006; Machanavajjhala, Kifer, Gehrke, & Venkitasubramaniam, 2007);
- inducing a tree that can be applied in practice without incurring too much classification costs. For example, it can be desirable that the expected costs for classifying examples do not exceed a certain predefined threshold value (Turney, 1995).
- inducing a tree that is *justifiable* from an expert’s perspective, by satisfying predefined constraints on predictions that allowed for given attribute values.

Many algorithms have been proposed to address each of these learning problems. Most common are the algorithms that rely on the principle of top-down induction through heuristics; C4.5 (Quinlan, 1993) and Cart (Breiman, Friedman, Olshen, & Stone, 1984) are the most well known examples of this. These algorithms do not explicitly minimize a global optimization criterion, but rely on the development of a good heuristic to obtain reasonable solutions. In practice, for each new problem setting that was studied, a new heuristic was proposed in the literature.

Other algorithms try to find good solutions by sampling from a posterior distribution (Chipman et al., 1998), or optimize a fitness function in a genetic algorithm (Turney, 1995).

To the best of our knowledge, no framework has been proposed which encapsulates the large number of decision tree learning problems listed above, and no

general algorithm is known for answering the canonical decision tree query exactly. The aim of this article is to develop such a framework, and to introduce the DL8 algorithm, which can be used to answer the listed decision tree learning queries.

The benefit of an exact algorithm is that we do not need to develop new heuristics to deal with many types of learning problems and constraints. We are sure that the answer found is the best that one can hope to achieve according to the predefined optimization criterion and constraints; no fine-tuning of heuristics is necessary. We can also use the results of DL8 to determine how well an existing heuristic decision tree learner approximates a global optimization criterion. This allows us to study the influence of constraints in a more fundamental way than is possible using heuristic decision tree learners.

An important issue is that one is often more interested in optimizing an implicit criterion, such as test set accuracy, than in optimizing an explicit criterion, such as training set accuracy. An interesting question is which explicit optimization criterion serves as the best proxy (or estimate) for test set accuracy. One can argue that global optimization criteria have been used in the pruning phase of traditional learning algorithms such as C4.5; our algorithm can be used to study how well these criteria perform if they would be used as global optimization criteria during tree induction, thus taking away the bias that is the result of the heuristics. A second issue that we can study is the belief that exact learners may be more sensitive to overfitting. In previous research, for instance, Murphy and Pazzani (1997) reported that for small, mostly artificial datasets, small decision trees are not always preferable in terms of generalization ability, while Quinlan and Cameron-Jones (1995) showed that when learning rules, exhaustive searching and overfitting are orthogonal. An efficient algorithm for learning decision trees under constraints allows us to investigate these observations for larger datasets and more complex models. In our experimental section, we compare test set accuracy, training set accuracy and an optimization criterion based on Quinlan's error based pruning.

Furthermore, we perform experiments with cost-based tree induction to illustrate that our algorithm can easily be extended to deal with many types of constraints, and to show that DL8 finds significantly better trees when learning problems become harder due to larger numbers of harder to optimize constraints.

To the best of our knowledge, few attempts have been made to implement a general and exact algorithm for learning decision trees under constraints; most people have not seriously considered the problem as it is known to be NP-complete (Hyafil & Rivest, 1976), and therefore, an efficient algorithm most likely does not exist. This theoretical result however does not imply that the problem is unsolvable in all cases. In the data mining literature, several exponential problems have been shown to be solvable in practice. In particular, the problem of frequent itemset mining has attracted a lot of research (Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996; Zaki, Parthasarathy, Ogihara, & Li, 1997; Han, Pei, & Yin, 2000), and many frequent itemset mining algorithms have been applied successfully despite the exponential nature of this problem. A key feature of DL8 is that it exploits a relation between constraints on itemsets and decision trees. Consequently, we can build on the results of the frequent itemset mining community.

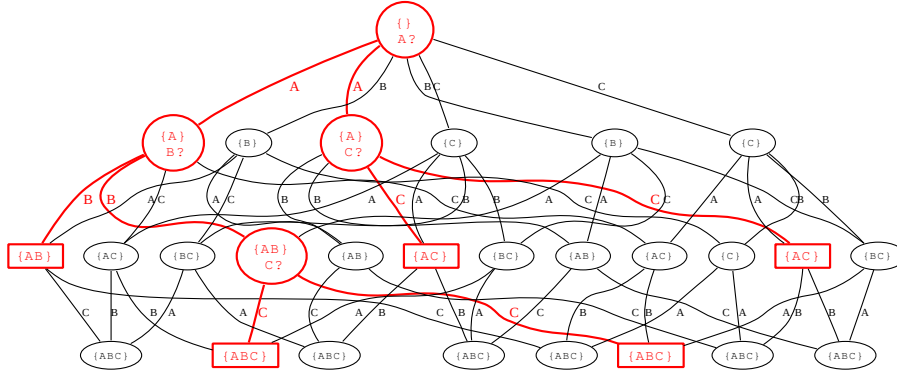


Fig. 1 An itemset lattice for items $\{A, \neg A, B, \neg B, C, \neg C\}$; binary decision tree $A(B(C(l,l),l),C(l,l))$ is hidden in this lattice

Even though our algorithm is not expected to work on all possible datasets, we will provide evidence that for a reasonable number of datasets, our approach is practically feasible.

The article is organized as follows. In Section 2, we introduce the concepts of decision trees and itemsets. In Section 3, we formalize the problems of privacy preserving decision tree induction, cost-based decision tree induction, and Bayesian tree induction; we put these learning problems in a single framework, and discuss the essential properties of the constraints and optimization criteria used in these learning problems. In Section 4, we study the complexity of decision tree induction in our framework. In Sections 5 we introduce the DL8 algorithm. In section 6, we show how our algorithm can be implemented using (modified) frequent itemset miners. In Section 7, we evaluate our algorithm. Section 9 gives further related work. We conclude in Section 10.

2 Itemset Lattices for Decision Tree Mining

Let us first introduce some background information about *frequent itemsets* and *decision trees*.

2.1 Itemset Mining

Let $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ be a set of items and let $D = \{T_1, T_2, \dots, T_n\}$ be a bag of transactions, where each transaction T_k is an itemset such that $T_k \subseteq \mathcal{I}$. A transaction T_k contains a set of items $I \subseteq \mathcal{I}$ iff $I \subseteq T_k$. The transaction identifier set (TID-set) $t(I) \subseteq \{1, 2, \dots, n\}$ of an itemset $I \subseteq \mathcal{I}$ is the set of all identifiers of transactions that contain itemset I .

The frequency of an itemset $I \subseteq \mathcal{I}$ is defined to be the number of transactions that contain the itemset, i.e. $\text{freq}(I) = |t(I)|$; the support of an itemset is $\text{support}(I) = \text{freq}(I)/|D|$. An itemset I is said to be frequent if its support is higher

than a given threshold $minsup$; this is written as $support(I) \geq minsup$ (or, equivalently, $freq(I) \geq minfreq$).

Definition 1 A complete lattice is a partially ordered set in which all elements have both a least upper bound and a greatest lower bound.

The set $2^{\mathcal{I}}$ of all possible subsets of $\mathcal{I}$, together with the inclusion operator $\subseteq$, constitutes a lattice. The least upper bound of two sets is computed by the $\cap$ operator, the greatest lower bound by the $\cup$ operator. There exists a lower bound $\perp = \emptyset$ and a higher bound $\top = \mathcal{I}$ for this lattice. Usually, lattices are depicted as in Figure 1, where edges denote a subset relation between sets; sets are depicted as nodes. The figure only shows parts of the lattice for the 6 items $\mathcal{I} = \{A, \neg A, B, \neg B, C, \neg C\}$.

In this work, we are interested in finding frequent itemsets for databases that contain examples labeled with classes $c \in C$. We assume that every transaction has a class label c_i in the vector of class labels $\mathbf{c}$. If we compute the frequency $freq_c(I)$ of an itemset I for each class c separately, we can associate to each itemset the class label for which its frequency is highest. The resulting rule $I \rightarrow c(I)$, where $c(I) = \operatorname{argmax}_{c' \in C} freq_{c'}(I)$ is called a *class association rule* (Liu, Hsu, & Ma, 1998).

2.2 Decision trees

A decision tree aims at classifying examples by sorting them down a tree. The leaves of a tree provide the classifications of examples (Mitchell, 1997). Each node of a tree specifies a test on one attribute of an example, and each branch of a node corresponds to one of the possible outcomes of the test. We assume that all tests are boolean; nominal attributes are transformed into boolean attributes by mapping each possible value to a separate attribute. Due to the use of itemsets, numerical attributes need to be discretized and binarized beforehand. The input of a decision tree learner is then a binary matrix B , where B_{ij} contains the value of attribute i of example j .

Our results are based on the following observation.

Observation 1 Let us transform a binary table B into transactional form D such that $T_j = \{i | B_{ij} = 1\} \cup \{\neg i | B_{ij} = 0\}$. (Thus, every attribute value is mapped to a positive or a negative item.) Then the examples that are sorted down every node of a decision tree for B are characterized by an itemset of items occurring in D .

For example, consider the decision tree in Figure 2. We can determine the leaf to which an example belongs by checking which of the itemsets $\{B\}$, $\{\neg B, C\}$ and $\{\neg B, \neg C\}$ it includes. We denote the set of these itemsets with $leaves(T)$. Itemsets that correspond to internal nodes are denoted with $internal(T)$. All itemsets that correspond to paths in the tree are denoted with $paths(T)$. In this case, $paths(T) = \{\emptyset, \{B\}, \{\neg B\}, \{\neg B, C\}, \{\neg B, \neg C\}\}$.

Assume given an itemset I that corresponds to an internal node in a decision tree T . Then this internal node contains a test on an attribute and results in two

branches, which correspond to an itemset $I \cup i$ and an itemset $I \cup \neg i$ for some i . We denote these left-hand and right-hand branches of an itemset with $left(I, T)$ and $right(I, T)$, respectively. In our example, $right(\{\neg B\}, T) = \{\neg B, C\}$ and $right(\{\neg B\}, T) = \{\neg B, \neg C\}$.

A common function in decision tree induction is the entropy function. For example, some heuristic learners grow trees by minimizing the entropy in each node of the tree. Assume that we can partition a set of examples in n partitions, each of which contains a fraction p_i of the examples, then we can compute the entropy of this partition with

$$H(p_1, p_2, \dots, p_n) = - \sum_{i=1}^n p_i \log p_i.$$

In heuristic decision tree learners the partitions are usually defined by the class labels.

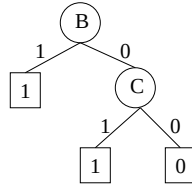


Fig. 2 An example tree

The leaves of a decision tree correspond to class association rules, as leaves have associated classes. In decision tree learning, it is common to specify a minimum number of examples that should be covered by each leaf. For association rules, this would correspond to giving a support threshold.

The accuracy of a decision tree is derived from the number of misclassified examples in the leaves: $accuracy(T) = \frac{|D| - f_e(T)}{|D|}$, where

$$f_e(T) = \sum_{I \in leaves(T)} e(I) \quad \text{and} \quad e(I) = freq(I) - freq_{c(I)}(I).$$

Another function that is commonly used for decision trees, is the size function

$$f_s(T) = |paths(T)|.$$

We believe that the most traditional decision tree learning problem is to find the tree

$$\operatorname{argmin}_T (f_e(T), f_s(T)),$$

by which we mean that we want to find a tree that minimizes error in the first place; ties between trees with equal errors are cut using the size function.

An illustration of the relation between itemsets and decision trees is given in Figure 1. In this figure, every node represents an itemset; an edge denotes a subset relation. Highlighted is one possible decision tree, which is nothing else than a set of itemsets. The branches of the decision tree correspond to subset relations. In this paper we present DL8, an algorithm for learning Decision trees from Lattices.

3 Constraints on Decision Trees

In this section we argue that many decision tree learning problems that have been studied in the literature can be seen as constraint-based decision tree learning problems. Our aim is to formulate a large number of decision learning problems as problems of finding a tree

$$\operatorname{argmin}_{T, \varphi(T)} f(T),$$

where $f(T)$ is a function that expresses a preference among trees, and φ is a formula that expresses a constraint on the trees that should be considered in the minimization.

Usually, $f(T)$ is the error function; $\varphi(T)$ can express a constrain such as that every leaf should be at most at depth d :

$$\varphi(T) = (\forall I \in \text{leaves}(T) : |I| \leq d).$$

Formulating decision tree learning problems in this framework allows us to develop more general algorithms, and provides us with a means to categorize the learning problems.

This section contains several illustrations of constraints on decision trees; readers not interested in these can skip till Section 3.3.

3.1 Privacy Preserving Tree Induction

3.1.1 Definitions A topic that has received considerable attention in recent years is that of privacy preservation in databases. There are several models for privacy protection in data, among which k -anonymity (Sweeney, 2002; Samarati, 2001) and ℓ -diversity (Machanavajjhala et al., 2007). Both models assume that we wish to publish an entire database publicly, including both sensitive and non-sensitive attributes. The aim of these models is to state requirements that data should fulfil to prevent that a hacker with knowledge about public, non-sensitive data of an individual of interest, can derive undesirable, private information from the published database.

These models are also of interest to machine learning, as in some cases one could be interested in publishing the model learned from data instead of the data itself. One needs to ensure that such models cannot be abused to derive sensitive information, and thus, the models for privacy preservation in data have to be mapped to constraints on the predictive models.

The privacy preservation models are based on the principle of *blocks*. Assuming that a database has both public and private attributes, a block is a set of examples that all have the same public, non-sensitive attribute values. A database is said to be k -anonymous if and only if every block has at least size k . Through this requirement, it is ensured that an attacker who knows a combination of values of public attributes, can still not uniquely identify an example in the data. Even though for $k = 2$ it is not possible to uniquely identify an example, an intelligent hacker is likely to identify additional information that allows him to determine which of two instances refers to an individual of interest. Therefore, to reduce this risk, it can be desirable that $k > 2$.

The k -anonymity model has undesirable weaknesses. In particular, it does not take the diversity of the private attributes into account. Assume that in a block all examples have the same value for a private attribute, if the entire dataset is published, the value of the private attribute can be deduced for all individuals in this block. It can be shown that such situations are very likely to happen in any database of reasonable size, and that therefore private information would be leaked about some individuals.

To address this issue, the ℓ -diversity model was proposed. In (Machanavajjhala et al., 2007), this requirement is formalized as follows: for every private attribute its entropy in every block should be higher than a given threshold ℓ . This ensures that every block has a sufficient variety in values for the private attribute. Furthermore, to avoid that some knowledge about one private attribute could be used to derive information about another private attribute, blocks are defined here by taking all attributes, except one, as public attributes.

3.1.2 Application to decision trees The application of k -anonymity in decision tree induction was studied by Friedman et al. (2006). In this study, it was assumed that we want to derive a decision tree from data with both public and private attributes, and we want to publish the decision tree, including statistics about the predictions in the leaves. The formalization of k -anonymity was as follows. Let us denote by $\mathcal{I}_{private}$ the items that are private, and by $\mathcal{I}_{public}$ the items that are public. For every example in the training data, represented by an itemset I , it was required that

$$\sum_{I' \in \text{leaves}(T), I' \cap \mathcal{I}_{public} \subseteq I \cap \mathcal{I}_{public}} \text{freq}(I') \geq k :$$

if we sort an example down the tree, while only considering its public attributes, we should end up in leaves that in total cover a sufficient number of examples.

If all attributes are public, except the class attribute, effectively this constraint requires that every leaf of the decision tree covers at least k examples.

This approach has the same weakness as other settings based k -anonymity, and does not deal with the diversity in attributes. To achieve a higher degree of privacy preservation, we need to extend the approach. One solution is to constrain a decision tree T as follows:

- to achieve k –anonymity, we require every leaf to cover at least k examples;

$$\forall I \in \text{leaves}(T) : \text{freq}(I) \geq k;$$

- to achieve ℓ –diversity for the class attribute, we require the entropy of the class attribute in every leaf to be higher than ℓ ;

$$\forall I \in \text{leaves}(T) : H_l(I) = H\left(\frac{\text{freq}_1(I)}{\text{freq}(I)}, \frac{\text{freq}_2(I)}{\text{freq}(I)}, \dots, \frac{\text{freq}_c(I)}{\text{freq}(I)}\right) \geq \ell;$$

- to achieve ℓ –diversity for other attributes, we require that every test splits the data into two parts that are balanced such that the entropy of the binary split is higher than ℓ . Optionally, we may choose not to impose this constraint as long as no test on a private attribute has been performed;

$$\forall I \in \text{internal}(T) : H_{in}(I) = H\left(\frac{\text{freq}(\text{left}(I, T))}{\text{freq}(I)}, \frac{\text{freq}(\text{right}(I, T))}{\text{freq}(I)}\right) \geq \ell;$$

here $\text{left}(I, T)$ and $\text{right}(I, T)$ denote the left-hand and right-hand child of itemset I in the tree T .

In other words, to achieve k –anonymity and ℓ –diversity, we need to impose a set of constraints on the leaves, and on the splits that are allowed.

Among the trees that satisfy these constraints, we are for instance interested in finding one that maximizes accuracy. Clearly, accuracy and high entropy are contradictory requirements; a tree with high entropy in its leaves, will be very inaccurate. In the ℓ –diversity model the trade-off is determined by the ℓ and k parameters.

Observe also that the ℓ –diversity constraints are particularly hard to deal with in traditional decision tree learners, as this constraint requires a certain amount of entropy in the leaves, while traditional learners aim at reducing this entropy as much as possible.

3.2 Cost-Based Tree Induction

One of the benefits of decision trees is that they are easily interpretable models that can be used as questionnaires. For instance, in the medical domain, a decision tree can be interpreted by a doctor as a sequence of tests to perform to diagnose a patient; an insurance company can interpret it as a sequence of questions to determine if a person is a desirable customer. In such cases, the application of a tree on an example incurs a certain cost: every question might require a certain amount of money or time to be answered. Furthermore, if a person is classified incorrectly, this might induce additional costs, in terms of expected missed revenue, or higher treatment costs. To induce trees under such cost constraints, algorithms for decision tree induction under cost constraints have been proposed (Turney, 1995; Esmeir & Markovitch, 2007).

Formally, these algorithms assume that the following information is given:

- a $c \times c$ misclassification cost matrix C where $C_{i,j}$ is the cost of guessing that an example belongs in class i , when it actually belongs in class j ;
- for every attribute i , a group g_i that it is contained in;
- for every attribute i the cost tc_i when a test on this attribute is performed;
- for every group g , an additional cost tc_g for the first test on an attribute in this group.

The motivation for having both a cost per group and per attribute is that it is often cheaper to ask related questions or perform related medical tests. Therefore later tests in a group of related tests are usually cheaper.

For a path $I \in \text{leaves}(T)$ we can formally define the cost of classifying an example in $t(I)$ as follows:

$$tc(I) = \sum_{i \in I} tc_i + \sum_{g \in \{g_i | i \in I\}} tc_g.$$

The expected costs for performing the tests in a tree are therefore:

$$f_{tc}(T) = \sum_{I \in \text{leaves}(T)} \frac{\text{freq}(I)}{|D|} tc(I).$$

The expected misclassification costs are:

$$f_{mc}(T) = \frac{1}{|D|} \sum_{I \in \text{leaves}(T)} \sum_c C_{c,c(I)} \text{freq}_c(I).$$

In (Turney, 1995; Esmeir & Markovitch, 2007) algorithms were proposed aiming at minimizing $f_c(T) = f_{tc}(T) + f_{mc}(T)$. Other settings are however also useful, and have remained unstudied. For instance, assume that the cost of a test is expressed in terms of the time that is needed to perform the test, while the misclassification cost is in terms of dollars. Combining these costs in a single measure would require time to be expressed in monetary terms, which may be undesirable and unpractical. An alternative could be to explicitly search for a tree that minimizes $f_{mc}(T)$, under the constraint that $tc(I) \leq \text{maxtime}$, for every itemset $I \in \text{leaves}(T)$. This would allow us to find inexpensive trees that have bounds on prediction times. One could evaluate such a query for multiple values of *maxtime* to come to a well-motivated trade-off between classification time and misclassification costs.

3.3 Interpretable and Non-Overfitting Tree Induction

Even though decision trees are symbolic classifiers, they are not always straightforward to interpret. In particular, if a tree contains hundreds of nodes, interpretability can be seriously affected; on top of that, complex trees may not even be good generalizations of the data. It can easily be seen that we can achieve a tree with maximum accuracy on training data when we continue splitting as long as the leaves are not pure; the resulting tree can be very large and is probably not even a good

model of the data. To ensure that we do not learn too complex models, several roads can be taken.

The simplest approach is to impose hard constraints. Most decision tree learners, among which C4.5 (Quinlan, 1993), impose a threshold on the minimum number of examples in a leaf. This ensures that every leaf generalizes over a large enough number of examples, and avoids that the algorithm continues splitting in too small groups of examples. Typical values for this threshold that are mentioned in the literature are 2 and 5. Formally, this constraint requires that

$$\text{freq}(I) \geq \text{minfreq},$$

for all $I \in \text{leaves}(T)$.

Frequency does not take into account to which classes examples belong. It might be desirable that the correlation of the predicted class labels with the true class labels is significant. One way to express this constraint is to require for every leaf $I \in \text{leaves}(T)$ that

$$\chi^2(I) \geq \text{mincorr},$$

where $\chi^2(I)$ is the χ^2 coefficient computed over the two by two contingency table containing the number of examples correctly and incorrectly classified in the leaf, assuming that for all examples the majority class is predicted.

Another hard constraint is to impose a threshold on the number of nodes in the decision tree. Heuristic decision tree learners usually enforce this constraint by first growing a full decision tree with far too many nodes, and by then iteratively pruning this tree until a tree of the desired size is obtained (Quinlan, 1993). We can also see the size of a tree as an optimization criterium:

$$f_s(T) = \sum_{I \in \text{paths}(T)} 1.$$

To avoid overfitting, pruning algorithms are among the most popular methods. Examples include error-based pruning and reduced-error pruning (Quinlan, 1993). We will discuss error-based pruning in more detail here. The main idea behind error-based pruning is to estimate the true error rate of a leaf, given the empirical error, and to turn an internal node into a leaf if this reduces the error estimate.

The true error is estimated by assuming that the class labels of the examples are the result of sampling with the unknown true error rate. Consequently, we can assume that the observed errors are binomially distributed, and we can compute a confidence interval on the true error given the observed error. A worst-case estimate on the true error can then be obtained.

The benefit of error-based pruning is that no training data needs to be set aside to perform pruning, in contrast to reduced-error pruning.

The error-based pruning estimate of error can be used as a global optimization criterion. Let us denote the estimated error of a leaf I with $ee(I)$, then we could be interested in minimizing

$$f_p(T) = \sum_{I \in \text{leaves}(T)} ee(I).$$

A property of the $ee(I)$ measure, such as proposed in (Quinlan, 1993), is that the estimated error of a leaf is at least 0.5 higher than the empirical error. Consequently, a tree with many leaves receives a larger additional penalty than a tree with few leaves. As global optimization criterion, this measure therefore ensures that smaller trees are preferred.

Instead of hard constraints, also soft constraints can be used. In this case, a prior can be defined to represent a preference for certain hypotheses in the hypothesis space. In (Buntine, 1992; Chipman et al., 1998; Angelopoulos & Cussens, 2005) the following approach is taken. We are interested in finding the tree that, given the data, is the most likely; in other words, we seek to maximize the posterior $p(T|D, \mathbf{c})$. Using Bayes' rule, we can formulate the optimization of $p(T|D, \mathbf{c})$ as finding

$$\begin{aligned} \operatorname{argmax}_T p(T|D, \mathbf{c}) &= \operatorname{argmax}_T \frac{p(\mathbf{c}|T, D)p(T|D)}{p(\mathbf{c}|D)} \\ &= \operatorname{argmin}_T -\log(p(\mathbf{c}|T, D)) - \log(p(T|D)). \end{aligned}$$

Please note that D is assumed to contain all data, except the vector of class attributes $\mathbf{c}$.

Let us define

$$f_b(T) = -\log(p(\mathbf{c}|T, D)) - \log(p(T|D)).$$

A technical complication is that traditional decision trees perform a single prediction in every leaf, and that therefore any data with misclassified examples would have zero probability. To address this issue, the decision trees are treated as *density estimation trees* in this context; a leaf contains a class distribution for examples ending in that leaf. The set of parameters of the tree (i.e. the densities in the leaves) is denoted by Θ ; for every leaf I , Θ contains the probability $\theta_{I,c}$ of examples of class c in that leaf. We need to find the tree structure that maximizes

$$p(\mathbf{c}|T, D) = \int p(\mathbf{c}|T, \Theta, D)p(\Theta|T, D)d\Theta.$$

Assuming that the parameters in the leaves are independent, and that the parameters $p(\Theta|T)$ for every leaf are identically Dirichlet distributed with parameters α , Chipman et al. (1998) showed that an exact analytical form of the integral is

$$p(\mathbf{c}|T, D) = \prod_{I \in \text{leaves}(T)} \left(\frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \right) \left(\frac{\prod_c \Gamma(\text{freq}_c(I) + \alpha_c)}{\Gamma(\text{freq}(I) + \sum_c \alpha_c)} \right),$$

where Γ is the standard gamma function that extends the factorial to real numbers. Observe that the logarithm of $p(\mathbf{c}|T, D)$ is the sum over terms that are computed independently for the leaves of the tree.

In (Chipman et al., 1998; Angelopoulos & Cussens, 2005) the following prior on decision trees was studied.

$$p(T|D) = \prod_{I \in \text{paths}(T)} P_{\text{node}}(I, T, D)$$

where

$$P_{node}(I, T, D) = \begin{cases} 1, & \text{if } children(I) = 0; \\ 1 - \alpha(1 + |I|)^{-\beta}, & \text{if } I \text{ is a leaf in } T \text{ and } children(I) \neq 0; \\ \frac{\alpha(1+|I|)^{-\beta}}{children(I)}, & \text{otherwise;} \end{cases}$$

Here $children(I)$ is the number of tests that can still be performed to split the examples in $t(I)$, while also respecting all hard constraints. Clearly, only hard constraints that do not depend on the class attributes $\mathbf{c}$, such as minimum frequency, can be used.

Parameters α and β determine the size of the tree. Observe again that the log-probability is the sum over terms that are computed independently for all nodes in the tree. Combining this prior, the overall log-likelihood can be seen as the sum of two terms, one that represents that accuracy of the tree, and one that represents a penalty term for undesirable trees.

Other priors were proposed in (Buntine, 1992).

3.4 Discussion

The constraints introduced in the previous sections are summarized in Figure 3. We make a distinction between:

- optimization criteria $f(T)$;
- constraints that are evaluated on nodes in the tree, and are of the shape $\forall I \in paths(T) : \varphi(T, I)$;
- constraints that are evaluated on trees as a whole, and are of the shape $f(T) \leq \tau$.

For each of these categories, Figure 3 also summarizes properties of the constraints that are important when building algorithms.

All constraints on nodes that we considered are *order independent*: we can evaluate if a node satisfies a constraint without considering the order of the tests leading to that node. Consequently, once we have evaluated a constraint for one order of the tests, we have also evaluated it for every other order.

All optimisation criteria that we considered are *additive*: they can be written as

$$f(T) = \sum_{I \in paths(T)} f(T, I),$$

where $f(T, I)$ is a function that computes a score for a path in a tree.

These properties allow us to state the problem that we study in this article. Our aim is to derive exact answers to the *query*

$$\operatorname{argmin}_{T, \varphi(T)} f(T),$$

where

- f is an additive optimization criterion;

Property	Positive Examples	Negative Examples
<i>Node constraints</i>		
Syntax Independent: constraint on a node $I \in \text{paths}(T)$ that can be evaluated by only considering the examples $t(I)$ covered by I .	$\text{freq}(I) \geq \text{minfreq}$, $\chi^2(I) \geq \text{mincorr}$, $H_l(I) \geq \ell$	$ I \leq d$, $tc(I) \leq \text{maxcost}$
Anti-Monotonic: constraint on a node $I \in \text{paths}(T)$ for which $\forall I' \subseteq I : \varphi(I) \Rightarrow \varphi(I')$.	$\text{freq}(I) \geq \text{minfreq}$, $ I \leq d$, $tc(I) \leq \text{maxcost}$	$\chi^2(I) \geq \text{mincorr}$, $H_l(I) \geq \ell$
Leaf: constraint that is applied only to the leaves; usually a constraint that is not anti-monotonic	$\forall I \in \text{leaves}(I) :$ $\text{freq}(I) \geq \text{minfreq}$, $\forall I \in \text{leaves}(I) :$ $\chi^2(I) \geq \text{minfreq}$	$\forall I \in \text{paths}(I) :$ $\text{freq}(I) \geq \text{minfreq}$
Tree Independent: constraint that can be evaluated without considering the tree in which it is contained.	$\text{freq}(I) \geq \text{minfreq}$, $\chi^2(I) \geq \text{minfreq}$, $tc(I) \leq \text{maxcost}$.	$H_{in}(I) \geq \ell$
Locally Tree Dependent: constraint on a node that can be evaluated by only considering the node and its children	$H_{in}(I) \geq \ell$	$\text{freq}(I) \geq \text{minfreq}$, $\chi^2(I) \geq \text{minfreq}$, $tc(I) \leq \text{maxcost}$.
<i>Optimization criteria</i>		
Leaf: optimization criterion which sums over the leaves only.	f_e, f_c, f_{ee}	f_s, f_b .
Syntax Independent: additive criterion in which $f(T, I)$ can be evaluated from $t(I)$ only	f_e, f_{ee}, f_s	f_b, f_c
<i>Global tree constraints: $f(T) \leq \tau$</i>		
Syntax Independent: a constraint based on an additive syntax independent criterion Polynomially bounded: a constraint $f(T) \leq \tau$ is polynomially bounded if τ is an integer and there is a polynomial $p(\tau)$ such that $ \{f(T) f(T) \leq \tau\} \leq p(\tau)$ i.e, the set of possible values $f(T)$ must be reasonably small given a constraint on $f(T)$.	$f_e(T) \leq \tau$, $f_{ee}(T) \leq \tau$, $f_s(T) \leq \tau$ $f_s(T) \leq \tau, f_e(T) \leq \tau, f_c(T) \leq \tau$	$f_b(T) \leq \tau, f_c(T) \leq \tau$ $f_c(T) \leq \tau, f_b(T) \leq \tau$

Fig. 3 Optimizations constraints and constraints on the tree as a whole

- φ is a conjunctive formula with node constraints that are only:
 - tree independent or locally tree dependent;
 - order independent.
 Additionally, φ may only contain additive global tree constraints of the form $f(T) \leq \tau$.
- φ contains at least one anti-monotonic node constraint;
- all global tree constraints are polynomially bounded.

We will see that, depending on the type of constraints used in φ , additional optimizations are possible that make answering this query more feasible in practice.

The most simple query that we consider within our framework is the following query: $\operatorname{argmin}_{T, \varphi(T)} f_e(T)$, where $\varphi(T) = (\forall I \in \text{leaves}(T) : \text{freq}(I) \geq \text{minfreq})$, and discovers the most accurate tree given a constraint on the minimum number of examples in every leaf. We will illustrate our algorithm first for this query, before extending it to deal with more complex constraints.

3.5 New Queries

An important advantage of our framework is that we can formalize arbitrary combinations of constraints. We formalize several such queries here.

Cost-Effective Privacy-Preserving Trees: If we are interested in finding a cheap decision tree that preserves privacy, we might pose the following query:

$$\operatorname{argmin}_{T, \varphi(T)} f_c(T),$$

where

$$\begin{aligned} \varphi(T) = & (\forall I \in \text{internal}(T) : H_{in}(I) \geq \ell) \wedge \\ & (\forall I \in \text{leaves}(T) : H_l(I) \geq \ell \wedge \text{freq}(I) \geq \text{minfreq}). \end{aligned}$$

Cost-Effective Small Non-Overfitting Trees: We can ask for a tree that is small, optimizes Quinlan's error-based pruning criterion, and has limited costs on every branch:

$$\operatorname{argmin}_{T, \varphi(T)} f_{ee}(T),$$

where

$$\begin{aligned} \varphi(T) = & (f_s(T) \leq \text{maxsize}) \wedge \\ & (\forall I \in \text{leaves}(T) : \text{tc}(I) \leq \text{maxcost} \wedge \text{freq}(I) \geq \text{minfreq}). \end{aligned}$$

Privacy-Preserving Trees under a Bayesian Prior: If we want to publish a tree that maximizes a criterion under a Bayesian prior, we can express this as finding a tree

$$\operatorname{argmin}_{T, \varphi(T)} f_b(T),$$

where

$$\begin{aligned} \varphi(T) = & (\forall I \in \text{internal}(T) : H_{in}(I) \geq \ell) \wedge \\ & (\forall I \in \text{leaves}(T) : H_l(I) \geq \ell \wedge \text{freq}(I) \geq \text{minfreq}). \end{aligned}$$

4 Complexity of Decision Tree Learning

It is instructive first to study the computational complexity of decision tree learning queries. The most well-known work results were obtained by Hyafil and Rivest (1976). These results state that the following problem is NP complete:

$$\operatorname{argmin}_{T, \varphi(T)} f_c(T) \quad \text{where } \varphi(T) = (\forall I \in \text{leaves}(T) : \text{freq}(I) \geq 1),$$

where infinite misclassification costs are assumed.

This problem setting is an instance of the framework that we proposed in this article, so we can assume that queries in our framework are NP complete in the worst case.

The proof of Hyafil and Rivest (1976) can easily be extended to some other instances of our framework, for instance, f_s can also be used as optimization criterion.

The belief that most decision tree learning problems are NP complete has led to a large number of algorithms that search for good instead of optimal solutions. Most popular are the greedy, heuristic approaches, of which CART and C4.5 are the most well-known examples (Breiman et al., 1984; Quinlan, 1993). Genetic algorithms have been used to set the parameters of a greedy algorithm that induces trees under cost constraints (Turney, 1995). An any-time learning algorithm for learning trees under cost constraints has been studied in (Esmeir & Markovitch, 2007). In this algorithm, repeatedly revisions of subtrees of a decision tree are enumerated, hoping to find an improvement over an earlier tree. Trees under Bayesian optimization criteria are typically computed using Markov Chain Monte Carlo (MCMC) algorithms, and sample a large number of trees (Chipman et al., 1998; Angelopoulos & Cussens, 2005).

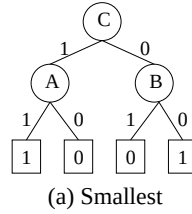
To the best of our knowledge, no single algorithm has been shown to be capable of exactly answering the broad range of decision tree learning queries that we are studying here.

Despite the fact that decision tree learning under constraints is NP complete in the general case, one could wonder how well heuristic decision tree learners are able to approximate optimal trees in theory. The XOR problem (Page & Ray, 2003a) is well-known difficult problem for heuristic tree learners. Assume given the data in Figure 4. In this data, none of the splits will yield information gain (ratio). Most decision tree learners will therefore decide not to split, while a perfect decision tree of 3 nodes exists. We can further extend this example. Assume that we have a database with equal amounts of positive and negative examples, and that the smallest tree with no errors on this data has n nodes. Let us extend this database with two attributes, such that we can predict the class from the two new attributes using an XOR. Then the optimal tree for this new data has 3 tests, while the heuristic learner will continue splitting to build a tree of at least n nodes. If size was our optimization criterion, this example shows that heuristic decision tree learners can obtain arbitrarily bad results.

It was argued by (Page & Ray, 2003a) that the XOR problem not only occurs in artificial data, but that also real-world examples are known: for instance, the

A	B	Class
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 4 XOR Example data



A	B	C	Class	Repeated
1	1	1	1	30×
1	1	0	0	20×
0	1	0	0	8×
0	1	1	0	12×
0	0	0	1	12×
0	0	1	0	18×

Fig. 5 Example database 1

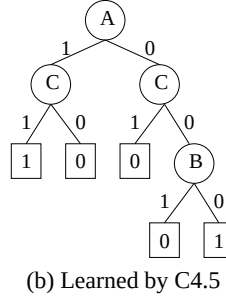


Fig. 6 Two accurate trees for example database 1

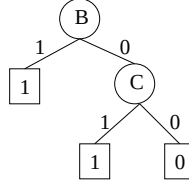
survival of *Drosophila* (fruitflies) depends on gender and gene activity: only if a fruitfly is female without active Sxl gene, or is male with active Sxl gene, does it survive. This concept is only likely to be captured well by learners that can deal with the XOR problem.

Other situations in which heuristic decision tree learners are tricked into making suboptimal decisions, are often due to the presence of misleading class distributions. Most heuristics are sensitive to making choices that predict large numbers of examples immediately correctly, even if this would result in larger models later on. We give some examples below.

As a first example consider the database in Figure 5, which is a variation of the XOR problem. Assume that we are interested in finding the smallest (or shallowest) most accurate decision tree, and we do not have a constraint on the number of examples in a leaf. Then the optimal tree is given in Figure 6(a), but C4.5 will find the tree in Figure 6(b), as the information gain (cq. ratio) of *A* is 0.098 (resp. 0.098), while the information gain of *C* is 0.029 (resp. 0.030).

As a second example consider the database in Figure 7, in which we have 2 target classes. Assume that we have reasons to believe that only a difference of 10 examples between a majority and a minority class provides significant evidence to prefer one class above the other in a leaf; therefore, we are interested in finding an optimal decision tree in which there is a difference of at least 10 examples between majority and minority classes. A tree satisfying these constraints exists, and is given in Figure 8. Clearly in this example *A* and *B* have approximately the same predictive power. However, C4.5 will prefer attribute *A* in the root, as it has information gain 0.33 (resp. ratio 0.54), while *B* only has information gain 0.26 (resp. ratio 0.37). C4.5, however, by making this choice, takes away examples

A	B	C	Class	Repeated
1	1	0	1	40×
1	1	1	1	40×
1	0	1	1	5×
0	0	0	0	10×
0	0	1	1	5×

Fig. 7 Example database 2**Fig. 8** The most accurate tree for example database 2

that could have been used to find a significant next split; no further splits can be performed. On the other hand, an optimal decision tree learner can prefer initially sub-optimal splits, if this is useful to build statistical support for tests deeper down the tree.

5 Building Optimal Decision Trees from Lattices

We will introduce our algorithm stepwise. First, we will introduce the main ideas behind our algorithm using a simple example query. Subsequently, we will provide extensions and more details of this algorithm.

5.1 Finding Accurate Trees

The learning problem that we address in this section is the discovery of the single tree which optimizes

$$\operatorname{argmin}_{T, \varphi(T)} f_e(T) \text{ where } \varphi(T) = (\forall I \in \text{leaves}(T) : \text{freq}(I) \geq \text{minfreq}). \quad (1)$$

In other words, we are looking for the most accurate tree in which every leaf has at least *minfreq* examples.

Pseudo-code of the algorithm is given in Algorithm 1. In this code, $c(I)$ is the function that computes the default class prediction for an itemset I . Function $l(\cdot)$ creates a leaf with this class. Function $n(i, T_1, T_2)$ creates an internal node which performs a test on attribute i , and has trees T_1 and T_2 as left-hand respectively right-hand subtrees. Function $f_{e, t(I)}(T)$ computes the error of a tree on the transactions contained in $t(I)$.

There are several properties that we exploit in our algorithm.

1. The error function is additive. Consequently, the error of a tree with test i in the root can be computed by summing the errors of the subtrees for examples $t(\{i\})$ and $t(\{\neg i\})$. We can independently minimize the errors of these two subtrees to obtain the tree with minimal error and i in its root.
2. The most accurate tree for a set of transactions $t(I)$ is independent of the order in which the tests in I are performed; once we have determined and stored the most accurate tree for the transactions covered by itemset I , we never need to compute it again.

Algorithm 1 DL8-SIMPLE(*minfreq*)

```

1: return DL8-SIMPLE-RECURSIVE( $\emptyset$ )
2:
3: procedure DL8-SIMPLE-RECURSIVE( $I$ )
4:   if DL8-SIMPLE-RECURSIVE( $I$ ) was computed before then
5:     return stored result
6:    $\mathcal{C} \leftarrow \{l(c(I))\}$ 
7:   for all  $i \in \mathcal{I}$  do
8:     if  $\text{freq}(I \cup \{i\}) \geq \text{minfreq}$  and  $\text{freq}(I \cup \{\neg i\}) \geq \text{minfreq}$  then
9:        $T_1 \leftarrow \text{DL8-SIMPLE-RECURSIVE}(I \cup \{i\})$ 
10:       $T_2 \leftarrow \text{DL8-SIMPLE-RECURSIVE}(I \cup \{\neg i\})$ 
11:       $\mathcal{C} \leftarrow \mathcal{C} \cup \{n(i, T_1, T_2)\}$ 
12:     end if
13:   end for
14:    $T \leftarrow \text{argmin}_{T \in \mathcal{C}} f_{e,t(I)}(T)$ 
15:   store  $T$  as the result for  $I$  and return  $T$ 
16: end procedure

```

3. Frequency is anti-monotonic: once a test splits in two sets of examples, one of which contains less than *minfreq* examples, we know that the test cannot be contained in a tree that satisfies the frequency constraint.

We exploit these observations as follows. In line 8 we exploit the anti-monotonicity to ignore paths that cannot be part of decision trees; in lines 9–10 we recursively determine the most accurate subtrees of test i , independently from each other; in line 15 we store the resulting tree; in line 4 we check that we do not recompute a tree that we have already computed earlier. The loop of line 7 ensures that every test is considered as the root of the tree.

The DL8-SIMPLE algorithm can be extended and optimized in many ways, several of which we discuss in the next sections.

5.2 Closed Itemsets and Generalized DL8-SIMPLE

Let us consider the query of formula (1) further. Our first optimization is based on the observation that two itemsets I and I' that cover the same set of examples (i.e., $t(I) = t(I')$), must have the same associated optimal decision tree. To reduce the number of itemsets that we have to store, we should avoid storing such duplicate sets of results.

The solution that we propose is to compute for every itemset its *closure*. The closure of an itemset I is the largest itemset that all transactions in $t(I)$ have in common. More formally, let $i(t)$ be the function which computes

$$i(t) = \bigcap_{k \in t} T_k$$

for a TID-set t , then the *closure* of itemset I is the itemset $i(t(I))$. An itemset I is called *closed* iff $I = i(t(I))$ (Pasquier, Bastide, Taouil, & Lakhal, 1999). If $t(I_1) = t(I_2)$ it is easy to see that also $i(t(I_1)) = i(t(I_2))$.

In our algorithm, instead of associating decision trees to itemsets, we can also associate decision trees to closed itemsets. We modify Algorithm 1 such that it checks in line 4 if a decision tree has already been computed for the closure of I ; in line 15, we associate computed decision tree(s) to the closure of I instead of to I itself. We call this algorithm DL8-CLOSED.

In practice this means that we build a datastructure of closed itemsets instead of ordinary itemsets. Lattices of closed itemsets are also known as *concept lattices*; closed itemsets are also known as *formal concepts* (Ganter & Wille, 1999).

We can generalize DL8-SIMPLE to deal with other anti-monotonic constraints and additive optimization criteria. In principle, we can replace the f_e function with any other function that is additive, and the $\text{freq}(I) \geq \text{minfreq}$ constraint with any other anti-monotonic predicate.

However, we cannot answer all decision tree learning queries from concept lattices. The use of a concept lattice is based on the assumption that the optimal trees for two itemsets that cover the same set of examples are the same. We can only make this assumption for constraints and optimization criteria that are syntax independent. As an example, consider the anti-monotonic constraint that every path in a tree has length at most d . Then the optimal tree that classifies the transactions $t(I) = t(I')$, where I and I' are itemsets of different lengths, can have maximum depth $d - |I|$ for itemset I and maximum depth $d - |I'|$ for itemset I' . These trees are not necessarily the same.

Consequently, we can only use DL8-CLOSED when answering syntax independent queries.

5.3 Non-Anti-Monotonic Constraints

For every itemset I for which it is called, DL8-SIMPLE-RECURSIVE assumes that it is a possibility that I is a leaf (line 6). When we are dealing with constraints on the leaves that are not anti-monotonic, we cannot make this assumption, as we cannot test such constraints in line 8. We need to modify DL8-SIMPLE (Algorithm 1) as follows. In line 6 we need to evaluate the non-anti-monotonic constraint; the set of candidate trees $\mathcal{C}$ should remain empty in this line if the constraint is not satisfied. After looping over all tests, the possibility should be taken into account that $\mathcal{C}$ remains empty. In this case, the function should return an empty tree. If an empty tree is returned in line 9 or line 10 for a test i , no subtree with i in its root can be constructed while satisfying the constraints.

5.4 Purity Constraints

For some combinations of optimization criteria and anti-monotonic constraints we can develop further pruning conditions. In this section we develop a pruning strategy that takes into account the *purity* of a leaf. An itemset I is called pure if all examples covered by I belong to the same class. If f_e is the optimization criterion and minimum frequency the anti-monotonic constraint, it does not make sense to

split the examples $t(I)$ of a pure node I further, as any resulting decision tree will never be more accurate, and unnecessarily large.

Purity is not a constraint on leaves of a tree: both itemsets I that are pure and not pure can be leaves. We can deal with purity in DL8-SIMPLE by inserting a test before entering the loop over possible tests of line (7). We should not enter this loop if the itemset I is pure.

For optimization criteria other than f_e , similar observations hold. Cost-based and Bayesian optimization functions can even only increase in value if we continue splitting for pure sets of examples. Furthermore, also for our examples of non-anti-monotonic leaf constraints, such as entropy constraints, further splitting of pure examples is redundant, as we cannot hope to improve the entropy of a pure set of examples. Only some constraints which are not usually considered useful, such as a *minimum leaf depth*, could require further splitting of pure sets of examples.

For particular queries, we can improve the concept of purity further. Assume again that we restrict ourselves to the problem of finding the most accurate tree given a leaf frequency constraint. In this case, we can relax the definition of purity.

Definition 2 For a given itemset I , let us sort the frequencies in the classes in descending order, $\text{freq}_1, \dots, \text{freq}_n$. Let minfreq be the minimum frequency used to build the lattice. An itemset I is loose-pure if $(\text{minfreq} - \sum_{i=2}^n \text{freq}_i(I)) > \text{freq}_2(I)$.

Let us illustrate this definition of loose-purity on an example. Given a prediction problem with 4 classes and $\text{minfreq} = 5$, assume that for an itemset (i.e. a node) we have:

Class	1	2	3	4
Number of examples	10	1	1	1

It does not make sense to split this node to increase the global accuracy of the tree. Indeed, the best split would separate all current errors (from class 2, 3 and 4) from the majority class, but since any leaf must contain at least 5 examples, and since none of these classes (in particular not the second most frequent class) have enough examples to take a majority in a leaf ($5 - 3 = 2 > 1$) compared to the first class, the error (3 in this example) in the tree cannot decrease. As a further example, if class 2 would have 2 examples, the node could still be split: we could create a split such that 2 examples of class 2 become classified correctly, 2 examples of class 3 and 4 remain classified incorrectly, and 1 example of class 1 becomes classified incorrectly; in total, a decrease in error of 1 could be obtained in the example.

Loose-purity is a stronger constraint than purity. We prove now that splitting a loose-pure node will never improve accuracy; consequently, when looking for the smallest most accurate tree, loose-purity can be used instead of purity.

Theorem 1 If $\text{freq}(I) \geq \text{minfreq}$, $\text{loose-pure}(I) = \text{true}$ and class k is the majority class in I , then for all $I' \supset I$ such that $\text{freq}(I') \geq \text{minfreq}$, class k is the majority class of I' .

Proof Let class 1 be the majority class in the examples $t(I)$. Then $\text{freq}(I) \geq \text{minfreq} \Leftrightarrow \sum_{i=1}^n \text{freq}_i(I) \geq \text{minfreq} \Leftrightarrow \text{freq}_1 \geq (\text{minfreq} - \sum_{i=2}^n \text{freq}_i(I))$. Since I is loose-pure we know that $(\text{minfreq} - \sum_{i=2}^n \text{freq}_i(I)) > \text{freq}_2(I) \geq 0 \Leftrightarrow \forall i, 2 \leq i \leq n, \text{freq}_i < \text{minfreq}$.

For class k ($k \neq 1$) to be the majority class in I' with $I' \supset I$ and $\text{freq}(I') \geq \text{minfreq}$, the number of examples of class k in I' should be higher than the minimum number of examples from class 1 that will still be in I' . This number is at least $(\text{minfreq} - \sum_{i=2}^n \text{freq}_i(I))$. So, for k ($k \neq 1$) to be the new majority class in I' , we must have $\text{freq}_k \geq (\text{minfreq} - \sum_{i=2}^n \text{freq}_i(I))$ which contradicts the definition of loose-purity for I ; therefore class 1 must be the majority class in I' .

For this particular type of learning problem, this theorem shows that we can use loose-purity instead of purity and still obtain correct answers.

5.5 Queries with Global Tree Constraints

More fundamental changes are needed to deal with global tree constraints, such as a maximum size constraint. We will first consider a simple case in which we only have one polynomially bounded tree constraint, such as size, and we want to find a tree $\text{argmin}_{T, \varphi(T)} f_1(T)$, where $\varphi(T) = f_2(T) \leq \tau \wedge (\forall I \in \text{leaves}(T) : \text{freq}(I) \geq \text{minfreq})$; both f_1 and f_2 are assumed to be additive optimization functions.

Let $\mathcal{R}$ be the set of values $\{f_2(T) | f_2(T) \leq \tau\}$. We need to restrict ourselves to trees whose f_2 values are in the range $\mathcal{R}$. Given the additivity of the f_2 function, a tree with a particular f_2 value can only be constructed from two subtrees with lower f_2 values. To find a tree with $f_2(T) \leq \tau$, we need to make sure that we find subtrees T' with values $f_2(T') < \tau$. We propose to do this by maintaining for every itemset I for every value in the range $r \in \mathcal{R}$ the optimal tree with $f_{2,t(I)}(T) = r$.

This can be achieved by modifying DL8-SIMPLE as indicated in Algorithm 2. In this algorithm, we have also included the modifications to deal with non-anti-monotonic leaf constraints (through predicate *pnon-anti-monotonic*) and purity constraints (through predicate *pure*). The main modification for global tree constraints is in line 22, where we compute for every allowed value $r \in \mathcal{R}$ the optimal tree according to the f_1 function. In line 4 of DL8 we can compute the tree that optimizes f_1 by determining $\text{argmin}_{T \in \mathcal{T}} f_1(T)$.

We can upgrade this algorithm to deal with multiple global tree constraints by assuming that f_2 can also be a multi-dimensional function. The range of this function is determined by the thresholds of all global tree constraints; for instance, if $\varphi(T) = f_s(T) \leq \text{maxsize} \wedge f_e(T) \leq \text{maxerror}$, we would have $\mathcal{R} = [0, \text{maxsize}] \times [0, \text{maxerror}]$.

An important issue is that if $\mathcal{R}$ is large, the amount of information that we need to store for an itemset can be large. In some cases, we can improve the space complexity by some pre- and post-processing of the query.

For instance, assume we would want to find some tree that has fewer errors than a threshold *maxerror* and is smaller than a threshold *maxsize*, without using an

Algorithm 2 $\text{DL8}(p_{\text{non-anti-monotonic}}, p_{\text{anti-monotonic}}, f_1, f_2, T)$

```

1: Optimize the  $f_1$  and  $f_2$  functions
2: Compute the range  $\mathcal{R}$  from the tree constraints  $f_2$  and bounds  $T$ 
3:  $\mathcal{T} \leftarrow \text{DL8-RECURSIVE}(\emptyset)$ 
4: Compute the optimal tree from  $\mathcal{T}$ 
5:
6: procedure DL8-RECURSIVE( $I$ )
7:   if DL8-RECURSIVE( $I$ ) was computed before then
8:     return stored result
9:   if  $p_{\text{non-anti-monotonic}}(I)$  then
10:     $\mathcal{C} \leftarrow \{l(c(I))\}$ 
11:   else
12:     $\mathcal{C} \leftarrow \emptyset$ 
13:   if not  $\text{pure}(I)$  then
14:     for all  $i \in \mathcal{I}$  do
15:       if  $p_{\text{anti-monotonic}}(I \cup \{i\})$  and  $p_{\text{anti-monotonic}}(I \cup \{\neg i\})$  then
16:          $\mathcal{T}_1 \leftarrow \text{DL8-RECURSIVE}(I \cup \{i\})$ 
17:          $\mathcal{T}_2 \leftarrow \text{DL8-RECURSIVE}(I \cup \{\neg i\})$ 
18:         for all  $T_1 \in \mathcal{T}_1, T_2 \in \mathcal{T}_2$  do
19:            $\mathcal{C} \leftarrow \mathcal{C} \cup \{n(i, T_1, T_2)\}$ 
20:         end if
21:       end for
22:    $\mathcal{T} \leftarrow \{\text{argmin}_{T, f_2, t(I)}(T) = r \mid f_1, t(I)(T) \mid r \in \mathcal{R}\}$ 
23:   store  $\mathcal{T}$  as the result for  $I$  and return  $\mathcal{T}$ 
24: end procedure

```

optimization criterion. Then we should use one of the global tree constraints as optimization criterion—for instance, the error function. Once we have found the most accurate tree given only the size constraint, we can check if this tree satisfies the initial error constraint.

Further tweaking is possible for a query in which the same additive function is used both as ranking function and as one of the global tree constraints. In this case we can arbitrarily choose one of the global tree constraints as ranking function, and remove this function from the global tree constraints during the search. DL8 will compute in its result set $\mathcal{T}$ a tree for every possible value of the global tree constraints. From the set $\mathcal{T}$ we can return the tree that is associated to the smallest value of the original ranking function, and which satisfies all global tree constraints.

For example, assume we wish to answer the following query: $\text{argmin}_{T, \varphi(T)} f_s(T)$, where $\varphi(T) = f_e(T) \leq \text{maxerror} \wedge f_s(T) \leq \text{maxsize} \wedge (\forall I \in \text{leaves}(T) : \text{freq}(I) \geq \text{minfreq})$. In principle we could maintain for every itemset a set of trees of $O(\text{maxerror} \times \text{maxsize})$, but such an approach is likely to fail.

Instead we can choose to use f_e as optimization function, i.e., we use $f_1(T) = f_e(T)$ and $f_2(T) = f_s(T)$. Then we let DL8 first find all these trees:

$$\mathcal{T} = \{ \text{argmin}_{T, f_s(T)=s} f_e(T) \mid 0 \leq s \leq \text{maxsize} \}.$$

From this set, we can pick the tree $\operatorname{argmin}_{T \in \mathcal{T}, \varphi(T)} f_s(T)$. This transformation reduces the amount of information that needs to be stored for every itemset significantly.

Our requirement that global tree constraints are polynomially bounded is a formalization of the requirement that the amount of information that we need to store for every itemset, should be limited. A query should be rewritten such that all global tree constraints are polynomially bounded.

This example illustrates that there is a strong analogy between decision tree learning and database querying: in some cases it can be beneficial to perform query rewriting before query execution, similar to what is commonly done in relational databases.

6 Implementing DL8

The most time consuming operations in DL8 are those that require access to the data. Essentially we need to collect the frequencies of a large amount of itemsets to answer a decision tree learning query. In this section we present two methods for determining these frequencies efficiently. The first method is aimed at queries that contain syntax dependent constraints (for which we need all itemsets), the second method is aimed at queries which are syntax independent (for which closed itemsets are sufficient).

6.1 All Itemsets

The most straightforward implementation of DL8 would access the data when a constraint needs to be evaluated: for instance, to evaluate a frequency constraint in line 15 of DL8, we could simply scan the entire data.

However, given that the problem of mining frequent itemset has been studied extensively in the literature, a relevant question is: can we use frequent itemset mining algorithms in some way to improve the practical efficiency of DL8?

The simplest way to use a frequent itemset miner, would be as follows. If we have a minimum leaf size constraint of *minfreq*, we could run a frequent itemset miner with this minimum frequency constraint, and store its output in an efficiently indexed data structure, such as a trie. In this data structure we should store the frequency of every itemset for every class label. When we run DL8, we can retrieve the frequencies of all necessary itemsets from this data structure instead of accessing the data.

Many frequent itemset miners have been studied in the literature; all of these can be used with small modifications to output the frequent itemsets in a convenient form and determine frequencies in multiple classes (Agrawal et al., 1996; Zaki et al., 1997; Han et al., 2000; Uno, Kiyomi, & Arimura, 2004). However, given that APRIORI (Agrawal et al., 1996) already builds a data structure of itemsets internally, the runtime overhead for building a dictionary of itemsets is expected to be smallest for APRIORI.

If we assume that the output of the frequent itemset miner consists of a graph structure such as Figure 1, then DL8 operates in time linear in the number of edges of this graph.

Unfortunately, the frequent itemset mining approach may compute frequencies of itemsets that can never be part of a decision tree. For instance, assume that $\{A\}$ is a frequent itemset, but $\neg A$ is not; then no tree will contain a test for attribute A ; itemset $\{A\}$ is redundant. We will show now that an additional anti-monotonic constraint can be used in the frequent itemset mining process to make sure that no such redundant itemsets are enumerated.

In this discussion, we ignore non-anti-monotonic leaf constraints.

If we consider the DL8 algorithm, an itemset $I = \{i_1, \dots, i_n\}$ is stored only if there is an order $[i_{k_1}, i_{k_2}, \dots, i_{k_n}]$ of the items in I (which corresponds to an order of recursive calls to DL8-RECURSIVE) such that for none of the proper prefixes $I' = [i_{k_1}, i_{k_2}, \dots, i_{k_m}]$ ($m < n$) of this order

- the $\neg pure(I')$ predicate is false in line (13);
- the conjunction $p_{anti-monotonic}(I' \cup \{i_{k_{m+1}}\}) \wedge p_{anti-monotonic}(I' \cup \{\neg i_{k_{m+1}}\})$ is false in line (15).

It is helpful to negate the *pure* predicate, as one can easily see that $\neg pure$ is an anti-monotonic predicate (every superset of a pure itemset, must also be pure). Furthermore we call $\neg pure$ an *internal* constraint as it is required to hold for every node in a tree.

We can now formalize the principle of itemset *relevancy*.

Definition 3 Let p_1 be an anti-monotonic node constraint and p_2 be an anti-monotonic internal constraint. Then the relevancy of I , denoted by $rel(I)$, is defined by

$$rel(I) = \begin{cases} p_1(I), & \text{if } I = \emptyset & (\text{Case 1}) \\ \text{true}, & \text{if } \exists i \in I \text{ s.t.} \\ & rel(I - i) \wedge p_2(I - i) \wedge \\ & p_1(I) \wedge p_1(I - i \cup \neg i) & (\text{Case 2}) \\ \text{false}, & \text{otherwise} & (\text{Case 3}) \end{cases}$$

Theorem 2 Let $\mathcal{L}_1$ be the set of itemsets stored by DL8, and let $\mathcal{L}_2$ be the set of itemsets $\{I \subseteq \mathcal{I} \mid rel(I) = \text{true}\}$. Then $\mathcal{L}_1 = \mathcal{L}_2$.

Proof We consider both directions.

“ $\Rightarrow$ ”: if an itemset is stored by DL8, there must be an order of the items in which each prefix satisfies the constraints as defined in line (13) and line (15). Then we can repeatedly pick the last item in this order to find the items that satisfy the constraints in case 2 of the definition of $rel(I)$.

“ $\Leftarrow$ ”: if an itemset is relevant, we can construct an order in which the items can be added in the recursion without violating the constraints, as follows. For a relevant itemset there must be an item $i \in I$ such that case 2 holds. Let this be the last item in the order; then recursively consider the itemset $I - i$. As this itemset is also relevant, we can again obtain an item $i' \in I - i$, and put this on the second last position in the order, and so on.

Relevancy is a property that can be pushed in a frequent itemset mining process.

Theorem 3 *Itemset relevancy is an anti-monotonic constraint.*

Proof By induction. The base case is trivial: if the $\emptyset$ itemset is not relevant then none of its supersets is relevant. Assume that for all itemsets X', X upto size $|X| = n$ we have shown that if $X' \subset X$: $\neg rel(X') \Rightarrow \neg rel(X)$. Assume that $Y = X \cup i$ and that X is not relevant. To prove that Y is not relevant, we need to consider every $j \in Y$, and consider whether case 2 of the definition is true for this j :

- $i = j$: certainly $Y - i = X$ is not relevant;
- $i \neq j$: we know that $j \in X$, and given that X is not relevant, either
 - $rel(X - j) = false$: in this case $rel(Y - j) = rel(X - j \cup i) = false$ (inductive assumption);
 - $p_1(X) = false$: in this case $p_1(Y) = false$ (anti-monotonicity of p_1);
 - $p_1(X - j \cup \neg j) = false$: in this case $p_1(Y - j \cup \neg j) = p_1(X - j \cup \neg j \cup i) = false$ (anti-monotonicity of p_1);
 - $p_2(X - j) = false$: in this case $p_2(Y - j) = p_2(X - j \cup i) = false$ (anti-monotonicity of p_2).

Consequently, $rel(Y)$ can only be *false*.

It is relatively easy to integrate the computation of relevancy in frequent itemset mining algorithms, as long as the order of itemset generation is such that all subsets of an itemset I are enumerated before I is enumerated itself. Assume that we have already computed all relevant itemsets that are a subset of an itemset I . Then we can determine for each $i \in I$ if the itemset $I - i$ is part of this set, and if so, we can derive the class frequencies of $I - i \cup \neg i$ using the formula $freq_k(I - i \cup \neg i) = freq_k(I - i) - freq_k(I)$. If for each i either $I - i$ is not relevant, or the predicate $p_1(I - i \cup \neg i)$ fails, we can prune I .

Pruning of this kind can be integrated in both depth-first and breadth-first frequent itemset miners.

We implemented two versions of DL8 in which the relevancy constraints are pushed in the frequent itemset mining process: DL8-APRIORI, which is based on APRIORI (Agrawal et al., 1996), and DL8-ECLAT, which is based on ECLAT (Zaki et al., 1997).

6.2 Closed Itemsets

DL8-ECLAT and DL8-APRIORI are based on a stepwise approach in which we first construct a lattice of itemsets using an existing mining algorithm and then run DL8. One could consider a similar approach when dealing with closed itemsets. However, we are faced with a fundamental problem if we want to do this. The DL8-CLOSED algorithm needs to be able to compute the closure of any given itemset. In practice it has been found that the closure of any given itemset is not very efficiently computable from a set of closed itemsets (Mielikäinen, Panov, &

Dzeroski, 2006). The complexity is known to become higher when the size of the lattice grows, while we can expect to deal with very large lattices in DL8.

If we would like to implement a step-wise approach efficiently in terms of runtime, it would be necessary that the closed itemset miner not only outputs closed itemsets, but also a data structure that allows the efficient determination of the closure of an itemset. However, we would need additional storage space for this data structure.

Given these observations, we believe that an integrated approach is most promising. In this approach, we use some of the techniques of closed itemset miners to speed-up the DL8-CLOSED algorithm. The remainder of this section is devoted to an outline of the choices that we made in this integrated algorithm.

The main idea is that during the search, we keep track of those items and transactions that are ‘active’. As parameters to DL8-RECURSIVE we add the following:

- the item i that was last added to I ;
- a set of *active items*, which includes item i , and represents all tests that can still be added to the itemset $I - i$;
- a set of *active transaction identifiers* storing $t(I - \{i\})$;
- the set of all items C that are in the closure of $I - i$, but are not part of the set of active items.

In the first call to DL8-RECURSIVE, all items and transactions are active. At the start of each recursive call (before line 7 of DL8-RECURSIVE is executed) we scan each active transaction, and test if it contains the item i ; for each active transaction that contains item i , we determine which other active items it contains. We use this scan to compute the frequency of the active items, and build the new set of active transaction identifiers $t(I)$. Those active items of which the frequency equals that of I , are added to the closure C . After this scan of the data, we build a new set of active items. For every item we determine if $p_{anti-monotonic}(I \cup \{i\})$ and $p_{anti-monotonic}(I \cup \{\neg i\})$ are true; if so, and if not $i \notin C$, we add the item to the new set of active items. In line 14 we only traverse the set of active items; the test of line 15 is then redundant. Finally, in line 16 and 17 the updated sets of active transactions and active items are passed to the recursive calls.

This mechanism for maintaining sets of active transactions is akin to the idea of maintaining projected databases that is implemented in ECLAT (Zaki et al., 1997) and FP-GROWTH (Han et al., 2000). In contrast to these algorithms, we know in our case that we have to maintain projections that contain both an item i and its negation $\neg i$. As we know that $|t(I)| = |t(I \cup i)| + |t(I \cup \neg i)|$, it is less beneficial to maintain TID-sets as in ECLAT, and we preferred this solution.

The main reason for calling DL8-RECURSIVE with the set of active transactions $t(I - \{i\})$ instead of $t(I)$, is that we can optimize the memory use of the algorithm. Instead of repeatedly allocating new memory for storing sets of active items and transactions, we can now maintain a single array to store these sets across all recursive calls. A projection is obtained by reordering the items in this array. Consequently, the memory use of our algorithm is entirely determined by the amount of memory that is needed to store the database and the closed itemsets C with their associated information; the memory use is $\theta(|D| + |C|)$.

Datasets	#Ex	#Test	Datasets	#Ex	#Test
anneal	812	36	tumor	336	18
a-credit	653	56	segment	2310	55
balance	625	13	soybean	630	45
breast	683	28	splice	3190	3466
chess	3196	41	thyroid	3247	36
diabetes	768	25	vehicle	846	55
g-credit	1000	77	vote	435	49
heart	296	35	vowel	990	48
ionosphere	351	99	yeast	1484	23
mushroom	8124	116	zoo	101	15
pendigits	7494	49			

Fig. 9 Datasets description

We store the closed itemsets in a trie data-structure, as is common in many other frequent itemset mining algorithms. Which information we store for every itemsets, depends on the query. If we are looking only for the most accurate decision tree, we associate to every itemset I fields for storing the *error* and the *root* of the decision tree classifying $t(I)$. The root field represents the test in the root of the tree for $t(I)$; it is not necessary to explicitly store the entire associated tree for every itemset, as we can retrieve the left-hand and right-hand subtrees by recursively retrieving the roots of subtrees from the trie data structure. In our implementation, for this query, 32 bits additional bit are associated to every frequent closed itemset to compute a decision tree, which is a relatively small overhead per itemset.

7 Experiments

In this section we compare the different versions of DL8 in terms of efficiency; furthermore, we compare the quality of the constructed trees with those found by the J48 decision tree learner implemented in WEKA (Witten & Frank, 2005). All experiments were performed on Intel Pentium 4 machines with in between 1GB and 4GB of main memory, running Linux. DL8 and the frequent itemset miners were implemented in C++.

The experiments were performed on UCI datasets (Newman, Hettich, Blake, & Merz, 1998). Numerical data were discretized before applying the learning algorithms using WEKA's unsupervised discretization method with a number of bins equal to 4. We limited the number of bins in order to limit the number of created attributes. Figure 9 gives a brief description of the datasets that we used in terms of the number of examples and the number of attributes after binarization.

7.1 Efficiency

The applicability of DL8 is limited by two factors: the amount of itemsets that need to be stored, and the time that it takes to compute these itemsets. We first

Algorithm	Uses relevancy	Closed	Builds tree
DL8-CLOSED	X	X	X
DL8-APRIORI	X		X
DL8-ECLAT	X		X
APRIORI-FREQ			
APRIORI-FREQ+DL8			X
ECLAT-FREQ			
LCM-FREQ			
LCM-CLOSED		X	

Fig. 10 Properties of the algorithms used in the experiments

evaluate experimentally how these factors are influenced by constraints and properties of the data. Furthermore, we determine how the different approaches for computing the itemset lattices compare. A summary of the algorithms can be found in Figure 10. Besides DL8-APRIORI, DL8-ECLAT and DL8-CLOSED, we also include unmodified implementations of the frequent itemset miners APRIORI (Agrawal et al., 1996), ECLAT (Zaki et al., 1997) and LCM (Uno et al., 2004) in the comparison. These implementations were obtained from the FIMI website (Bayardo, Goethals, & Zaki, 2004). The inclusion of unmodified algorithms allows us to determine how well relevancy pruning works, and allows us to determine the trade-off between relevancy pruning and trie construction.

Results for 8 datasets are listed in Figures 11 and 12. We chose datasets that cover a broad range of dataset properties, including both datasets with large and small numbers of attributes and transactions. In these runs we computed the most accurate tree given only a minimum frequency constraint. We aborted runs of algorithms that lasted for longer than 1800s. The results clearly show that in all cases the number of closed relevant itemsets is the smallest. The difference between the number of relevant itemsets and the number of frequent itemsets becomes smaller for lower minimum frequency values. The number of frequent itemsets is so large in most cases, that it is impossible to compute or store them within a reasonable amount of time or space. In those datasets where we can use low minimum frequencies (15 or smaller), the closed itemset miner LCM is usually the fastest; for low frequency values the number of closed itemsets is almost the same as the number of relevant closed itemsets. Bear in mind, however, that LCM does not output the itemsets in a form that can be used efficiently by DL8.

DL8-CLOSED is usually faster than DL8-APRIORI or DL8-ECLAT, but the experiments also reveal that for high minimum support values, the differences between closed relevant itemsets and relevant itemsets are small; the overhead of DL8-CLOSED seems too large.

In those cases where we can store the entire output of APRIORI in memory, we see that the additional runtime for storing results is significant. On the other hand, if we perform relevancy pruning, the resulting algorithm is usually faster than the original itemset miner.

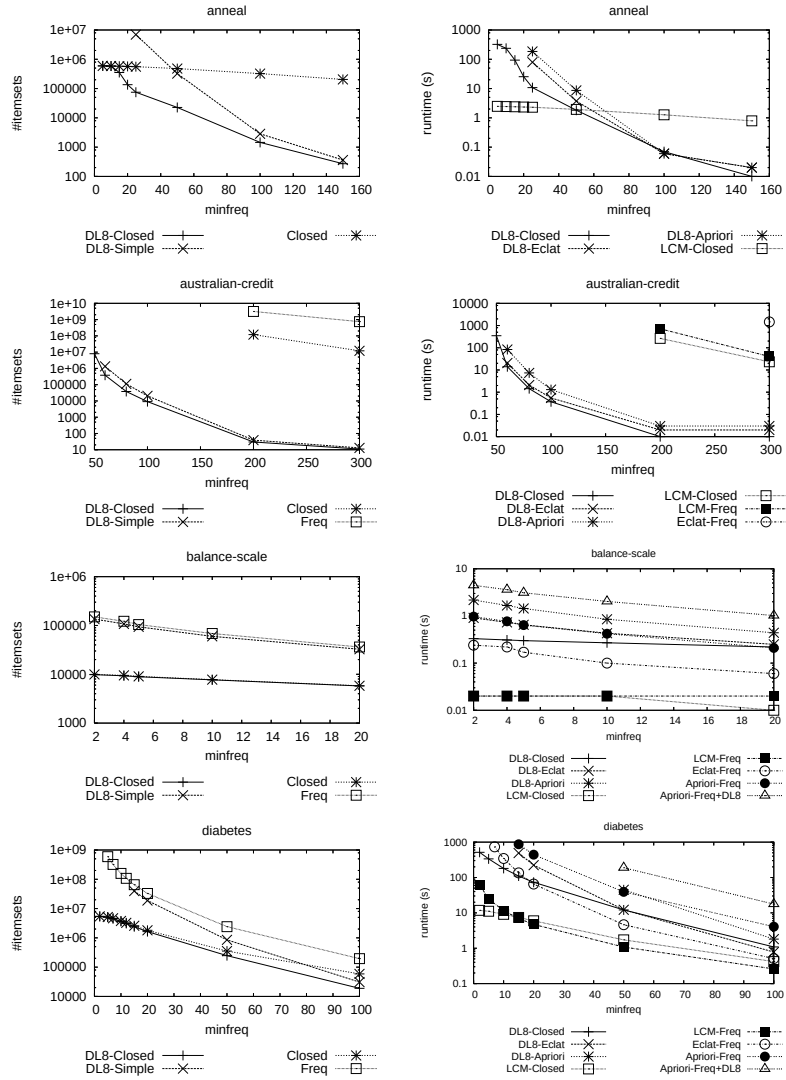


Fig. 11 Comparison of the different miners on 8 UCI datasets (1/2)

For the datasets with larger number of attributes, such as ionosphere and splice, we found that only DL8-CLOSED managed to run for support thresholds lower than 25%, but still was unable to run for support thresholds lower than 10%.

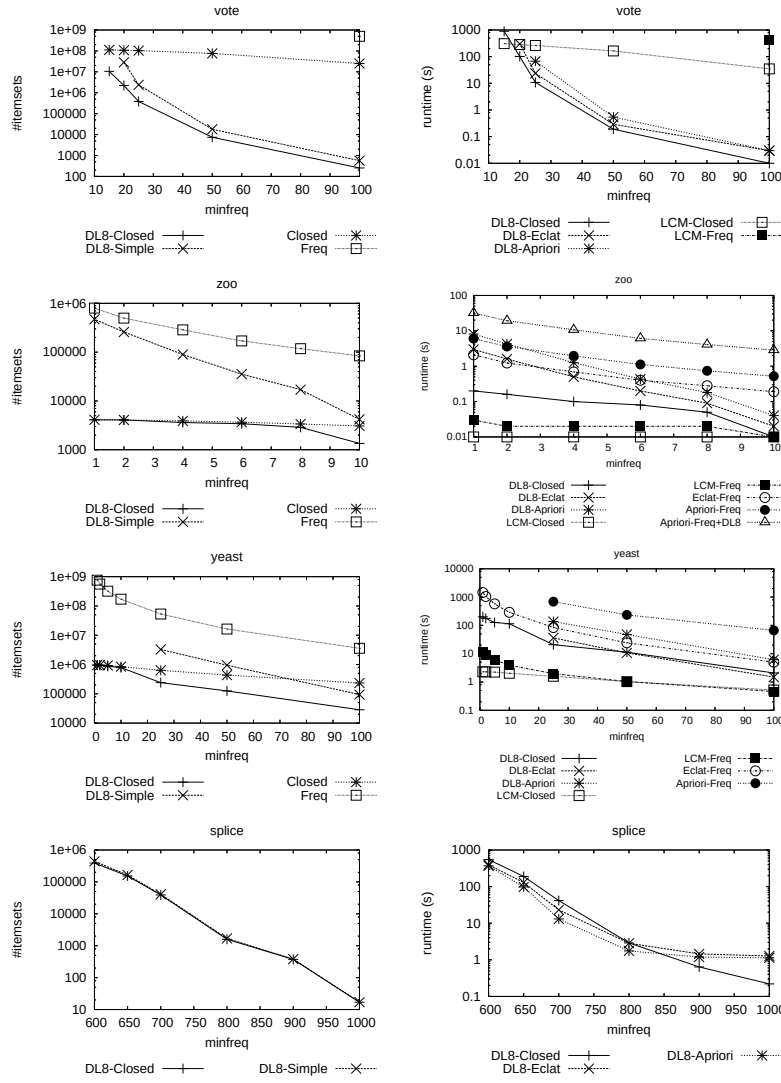


Fig. 12 Comparison of the different miners on 8 UCI datasets (2/2)

7.2 Accuracy

In this section we compare the accuracies of the decision trees learned by DL8 and J48 on twenty different UCI datasets. J48 is the Java implementation of C4.5 (Quinlan, 1993) in WEKA. We used a stratified 10-fold cross-validation to compute the training and test accuracies of both systems. The bottleneck of our algorithm is the in-memory construction of the lattice, and, consequently, the application of our algorithm is limited by the amount of memory available for the construction of this lattice. For the results of Figures 13 and 14, we used minimum frequency

Datasets	Freq		Train acc		Unpruned Test acc		Size	
	#	%	J48	DL8	J48	DL8	J48	DL8
anneal	2	.2	.89	.89	.82	.82	106.6	87.8
a-credit	40	6.1	.87	.89	.85	.88	6.4	11.0
balance	2	.3	.90	.90	.82	.81	99.0	114.4
breast-w	2	.3	.98	1.00	.95	.94	31.6	48.0
chess	200	6.2	.91	.91	.91	.91	9.0	8.6
diabetes	2	.2	.90	.99	.68	.66	200.2	288.4
german-credit	150	15	.72	.74	.71	.73	6.4	7.0
german-credit	100	10	.73	.75	.70	.70	6.4	11.6
heart-c	2	.6	.94	1.00	.76	.74	67.6	74.4
ionosphere	50	14.2	.83	.86	.79	.84	4.0	7.4
ionosphere	40	11.3	.89	.89	.88	.88	5.0	6.8
mushroom	600	7.4	.92	.98	.92	.98	5.0	13.8
pendigits	470	6.3	.68	.75	.67	.75	21.0	21.0
p-tumor	2	.5	.63	.71	.40	.36	116.4	152.2
segment	150	6.5	.77	.86	.76	.85	15.6	16.8
segment	120	5.2	.84	.87	.84	.87	19.8	25.8
soybean	40	6.3	.58	.65	.57	.66	17.0	20.6
splice	700	21.9	.74	.74	.74	.73	5.0	5.0
thyroid	80	2.4	.91	.92	.91	.91	1.0	13.4
thyroid	40	1.2	.92	.92	.91	.91	9.2	34.4
vehicle	50	5.9	.63	.71	.59	.67	17.0	22.4
vote	10	2.3	.96	.98	.94	.93	4.6	29.6
vowel	65	6.6	.40	.47	.35	.43	19.2	22.6
yeast	2	.1	.74	.82	.49	.48	501.2	724.2

Fig. 13 Comparison of J48 and DL8, without pruning and with the same support constraint

Datasets	Freq		Train acc		Pruned Test acc		Size		J48 Pruned Freq = 2				
	#	%	J48	DL8	J48	DL8	J48	DL8	Test acc	No D.	S	No D.	Size
anneal	2	.2	.86	.87	.82	.82	44.4	45.6	.82	.88	0	-	44.4
a-credit	40	6.1	.87	.89	.85	.88	5.0	11.0	.84	.86	+	0	36.4
balance	2	.3	.89	.89	.80	.80	72.4	65.4	.80	.79	0	0	72.4
breast-w	2	.3	.97	.98	.96	.96	15.6	18.0	.96	.95	0	0	15.6
chess	200	6.2	.91	.95	.90	.95	8.6	13.0	.99	1	-	-	54.4
diabetes	2	.2	.84	.92	.74	.71	69.0	135.2	.74	.73	0	0	69.0
g-credit	150	15.0	.72	.74	.71	.73	5.8	6.8	.71	.70	0	0	163.0
g-credit	100	10.0	.73	.75	.70	.71	6.2	9.6	.71	.70	0	0	163.0
heart-c	2	.6	.90	.97	.78	.77	31.6	50.2	.78	.80	0	0	31.6
ionosph	50	14.2	.83	.86	.79	.84	4.0	6.8	.86	.91	0	-	34.6
ionosph	40	11.3	.89	.89	.88	.88	5.0	5.6	.86	.91	0	0	34.6
mushro	600	7.4	.92	.98	.92	.98	5.0	13.8	1	1	-	-	16.8
pendigits	470	6.3	.68	.75	.67	.75	21.0	21.2	.95	.96	-	-	340.0
p-tumor	2	.5	.60	.67	.40	.40	81.2	105.2	.40	.40	0	0	81.2
segment	150	6.5	.78	.86	.76	.85	15.6	16.4	.95	.97	-	-	112.6
segment	120	5.2	.84	.87	.84	.86	19.8	21.2	.95	.97	-	-	112.6
soybean	40	6.3	.58	.65	.57	.66	17.0	20.6	.82	.82	-	-	88.0
splice	700	21.9	.74	.74	.74	.73	5.0	5.0	.94	.94	-	-	126.8
thyroid	40	1.2	.91	.92	.91	.91	3.2	7.8	.91	.99	0	-	34.2
vehicle	50	5.9	.63	.71	.59	.67	14.8	22.4	.70	.72	0	-	138.0
vote	20	4.5	.96	.96	.96	.94	3.0	7.6	.96	.96	0	0	12.6
vote	10	2.3	.96	.97	.96	.95	3.0	10.4	.96	.96	0	0	12.6
vowel	65	6.6	.40	.47	.35	.43	18.6	21.8	.78	.82	-	-	290.0
yeast	2	.1	.68	.75	.53	.50	186.0	307.2	.53	.56	-	-	186.0

Fig. 14 Comparison of J48 (pruned) and DL8 when using the f_{ee} optimization function, for the same support and for the default support constraint (2) of J48 in the last three columns

as leaf constraint. The frequency was lowered to the lowest value that still allowed the computation to be performed within the memory of our computers. For some datasets, we also give results for higher frequency values to evaluate the influence of frequency on the accuracy of the trees. This evaluation is presented in more details in Figure 16.

For J48, results are provided for pruned trees (Figure 14) and unpruned trees (Figure 13); for DL8 results are provided in which the f_e (unpruned; Figure 13) and f_{ee} (pruned; Figure 14) functions are optimized (see Section 3). First, both algorithms were applied with the same minimum frequency setting. We used a corrected two-tailed t-test (Nadeau & Bengio, 2003) with a significance threshold of 5% to compare the test accuracies of both systems. A test set accuracy result is in bold when it is significantly better than its counterpart result on the other system. In the last five columns of Figure 14, we also give results for J48 with its default $minfreq = 2$ setting; we applied J48 both on the original, undiscretized data and the discretized data that was also used as input to DL8. The test set accuracies of J48 are compared to the test set accuracies given by DL8 using the f_{ee} optimization function. The results of the significance test are given in the “S” column : “-” means that J48 is significantly better, “+” that it is significantly worse and “0” not significantly different.

The experiments show that both with and without pruning the optimal trees computed by DL8 have a better training accuracy than the trees computed by J48 with the same frequency values and the same discretization. Furthermore, on the test data, DL8 is significantly better than J48 on 9 of the 20 datasets for the unpruned version and on 8 for the pruned version. DL8 performs worse on test data for one dataset (yeast) when the pruning in J48 is used. The experiments also show that when pruned trees are compared to unpruned ones, the sizes of the trees are on average 1.75 times smaller for J48 and 1.5 time smaller for DL8. After pruning, DL8’s trees are still 1.5 times larger than J48’s ones. In one case (balance-scale), the tree computed by DL8 is significantly smaller on average (65.4 nodes) than the one computed by J48 (72.5 nodes) for the same support and a similar test accuracy, as computed by a corrected two-tailed t-test. In some other cases (*segment*, *vehicle*,...), the test accuracy result of DL8 is significantly better than the one given by J48 for trees only a few nodes (up to 8 nodes) more. This confirms earlier findings which show that smaller trees are not always desirable (Provost & Domingos, 2003). The trees computed by the pruned version of J48 and DL8 on the *vehicle* dataset for $minfreq = 80$ are given in Figure 15; in this case the tree learned by DL8 is highly unbalanced, but we can address this issue by using different constraints if this is desirable.

If we decrease the frequency threshold down to a certain value, the training accuracy increases, but the experiments on the 7 datasets for which we were able to reach a $minfreq$ of 2, indicate that for testing accuracy, low thresholds are not always the best option. For example, Figure 16 shows the evolution of the training and test accuracies of both systems in their pruned and unpruned version when the support increases. As expected, the training accuracies always decrease when the support increases. However, this behavior is less clear for test accuracies.

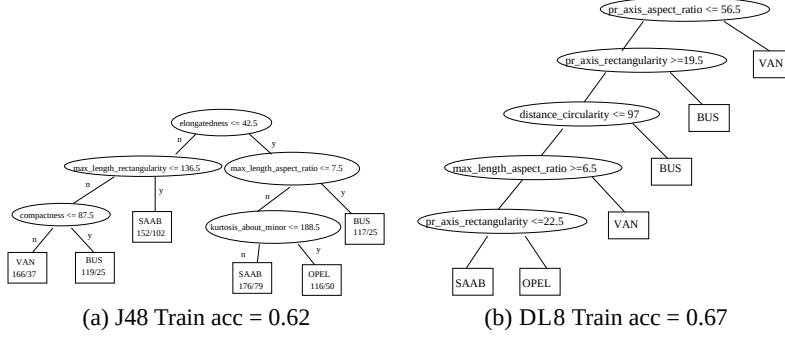


Fig. 15 Trees computed by J48 and DL8 on the “vehicle” dataset for support = 80

For example, for the diabetes dataset, the test accuracies clearly increase with the minimum support.

If we compare the best results of DL8 with those given by J48 with minimum frequency 2, we see that J48’s test accuracies are significantly better on 8 of the 20 datasets used and only worse on one (australian credit). However, for the 11 datasets for which there is no significant difference between the test accuracy of DL8 and J48, the trees are a lot smaller for DL8. Furthermore, the situation is even worse for DL8 if J48 is applied on the undiscretized data.

There are many possible explanations for the worse results of DL8 on these 8 datasets. One possible explanation is that in many of the datasets with bad results, the number of classes is very high. In some datasets, the lowest minimum frequency threshold for which we could run our algorithm was much higher than the number of examples in the smallest class, which makes it impossible to classify all examples correctly. To test this hypothesis, we decided to change the minimum frequency constraint into a disjunction of minimum support constraints; every class is given the same minimum support constraint. In this way, we allow that a leaf covers a small number of examples if all examples belong to the same class, but we do not allow that a leaf contains a small number of examples if they belong to many different classes. The results of these experiments are listed in Figure 17. As we can see, the accuracies increase in all cases, although not always enough to beat J48 with its default setting.

One of the strengths of DL8 is that we can explicitly constrain the size or accuracy. We therefore studied the relation between decision tree accuracies and sizes in more detail. In Figure 18, we show results in which the average size of trees constructed by J48, is taken as a constraint on the size of trees mined by DL8. None of the results given by DL8 are significantly better nor significantly worse than those given by J48.

DL8 can also compute, for every possible size of a decision tree, the smallest error on training data that can possibly be achieved. For two datasets, the results of such a query are given in Figure 19. In general, if we increase the size of a decision tree, its accuracy improves quickly at first. Only small improvements can be obtained by further increasing the size of the tree. If we lower the frequency

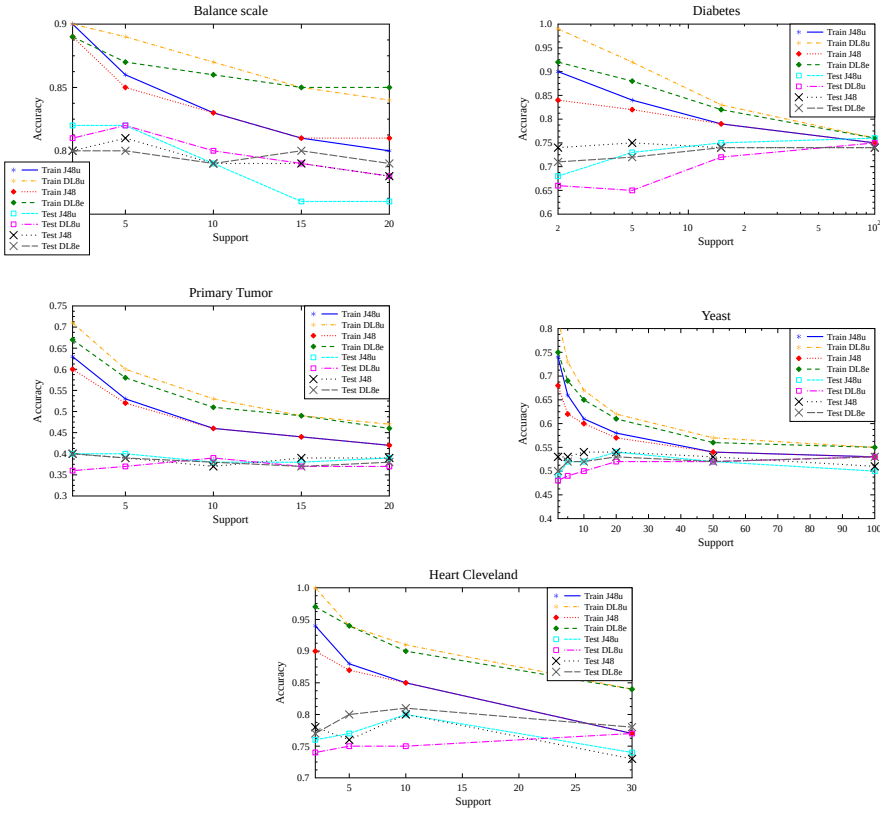


Fig. 16 Evolution of the training and test accuracies of J48 (unpruned and pruned) and DL8 (f_e and f_{ee} optimization functions) for various minimum frequency thresholds

Datasets	minsups (%)	Trainacc	Testacc	Size	Depth
pendigits	32	0.785	0.788	7.2	4.1
segment	22	0.911	0.899	33.6	9.3
soybean	40	0.756	0.734	33.2	10.3
splice	40	0.740	0.731	5.0	3.0
vowel	15	0.700	0.630	79.0	9.3
yeast	2	0.660	0.521	136.4	13.1

Fig. 17 Results for a disjunction of class minimum support constraints

threshold, we can obtain more accurate trees, but only if we allow a sufficient amount of nodes in the tree.

Figures such as Figure 19 are of practical interest, as they allow a user to trade-off the interpretability and the accuracy of a model.

Datasets	Sup	Max size DL8	Test acc		Size	
			J48	DL8	J48	DL8
diabetes	15	27	0.75	0.74	26.4	27.0
g-credit	100	7	0.70	0.72	6.7	7.0
heart-c	10	14	0.80	0.80	14.0	13.0
vote	15	4	0.95	0.96	3.4	3.0
yeast	2	190	0.53	0.53	186.0	189

Fig. 18 Influence of the size constraint on the test accuracy of DL8 (unpruned)

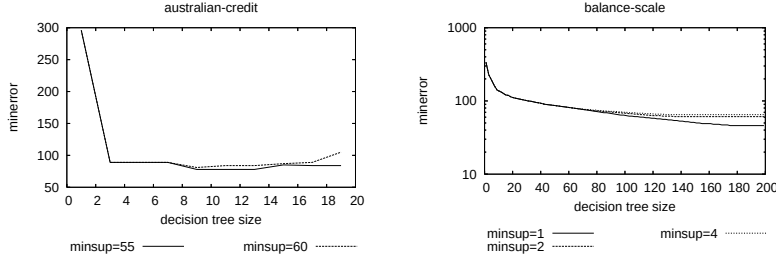


Fig. 19 Errors of decision trees on training data as function of tree size

The most surprising conclusion that may however be drawn from all our experiments, is that optimal trees perform remarkably well on most datasets. Our algorithm investigates a vast search space, and its results are still competitive. The experiments indicate that the constraints that we employed, either on size, or on minimum support, are sufficient to reduce model complexities and achieve good predictive accuracies. Our experiments confirm the results that were obtained by Quinlan and Cameron-Jones (1995) for rule sets, and show that overfitting and exhaustive search are orthogonal.

8 Optimal Cost-sensitive Decision Tree Learning

In this section we compare DL8 to ICET (Inexpensive Classification with Expensive Tests) (Turney, 1995), introduced in section 3.2. The aim of these two algorithms is to induce decision trees which minimize $f_c(T) = f_{tc}(T) + f_{mc}(T)$; ICET uses a genetic algorithm for this purpose. The comparison is made using the five well known datasets from the UCI repository (Newman et al., 1998) for which test costs are provided (BUPA Liver Disorders, Heart Disease, Hepatitis Prognosis, Pima Indians Diabetes and Thyroid Disease). We binarized the input data to DL8 similarly as in the previous section. A comparison is given in Figure 20. The figure shows the *average cost of classification* given by the algorithms as a percentage of the *standard cost* of classification for different costs of misclassification error.

The *average cost of classification* is computed by dividing the total cost of applying the learned decision tree on all test examples by the number of examples in the test set. The total cost of using a given decision tree is $tc(I) = \sum_{i \in I} tc_i +$

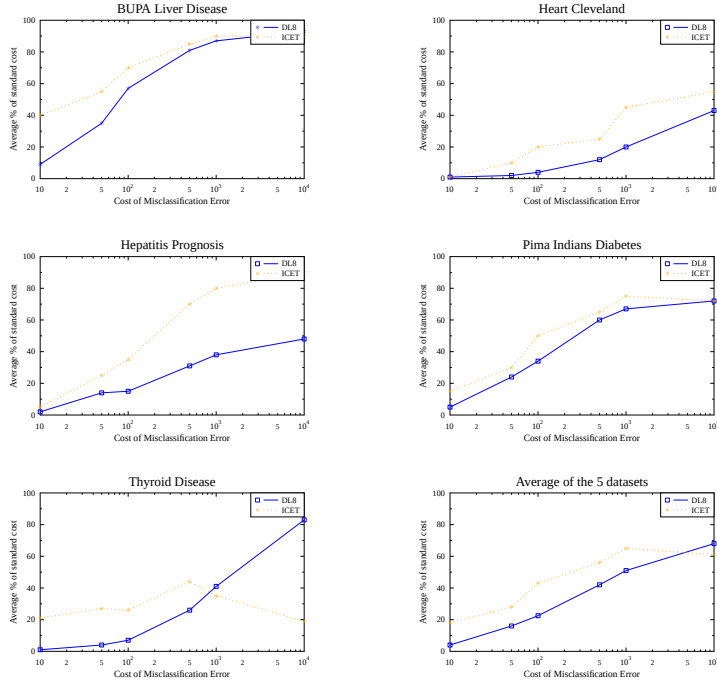


Fig. 20 Comparison of the cost-sensitive decision trees ICET and DL8 with various support (the lower curve, the better)

$\sum_{g \in \{g_i | i \in I\}} tc_g$, i.e. the sum of all tests that are chosen and the misclassification cost as specified in the misclassification matrix $C_{i,j}$. Let $f_c \in [0, 1]$ be the support of class c in the given dataset, i.e. the fraction of the examples in the dataset that belong in class c . Let T be the total cost of performing all possible tests (counting only once the additional cost for the tests in the same group). The *standard cost* is $T + \min_c (1 - f_c) \max_{i,j} C_{i,j}$. The second term is computed from the support of the majority class in the dataset and the highest misclassification cost that an algorithm can have if examples are incorrectly classified as the majority class.

In the experiments, we vary the misclassification costs (as specified in the matrix $C_{i,j}$) from \$10 to \$10,000. For the sake of simplicity, we consider simple cost matrices i.e. all misclassification costs are equal. The lowest frequency threshold we could use for DL8 is 2 for the BUPA Liver Disorders dataset, 16 for the Heart Disease dataset, 5 for the Hepatitis Prognosis dataset, 15 for the Pima Indians Diabetes dataset and 55 for the Thyroid Disease dataset. Note that these supports can be higher than those reported in Figures 13 and 14 because the syntax dependent optimization criterion required us to use DL8-ECLAT instead of DL8-CLOSED.

The results show a better performance for DL8 for 4 of the 5 datasets. However, for the ann-thyroid dataset, DL8's results are worse for high misclassification costs ($> 10^3$). Further investigations revealed that this behavior is the result of the low number of bins that we used in our discretization, which resulted in an er-

ror rate that was close that of a majority classifier in this very unbalanced dataset (3 classes with a distribution (93, 191, 3488)). Once the same discretization was used, the error rates were more similar to each other, and the difference in behavior disappeared.

9 Related work

The search for decision trees that optimize a global criterion directly dates back to the 70s, when several algorithms for building such trees were proposed. Their applicability was however limited. Research in this direction was therefore almost abandoned in the last two decades. Heuristic tree learners were believed to be nearly optimal for prediction purposes. In this section, we will provide a brief overview of the early work that has been done on searching optimal decision trees. To clarify the relation of these old results with our work, we will summarize them in more modern terminology.

Garey (1972) proposed an algorithm for constructing an optimal *binary identification procedure*. In this setting, a binary database is given in which every example belongs to a different class. Furthermore, every example has a weight and every attribute has a cost. The target is to build a decision tree in which there is exactly one leaf for every example; the expected cost for classifying examples should be minimal. First, a dynamic programming solution is proposed in which bottom-up all subsets of the examples are considered; then, an optimization is introduced in which a distinction is made between two types of attributes: attributes that uniquely identify one example, and attributes that separate classes of examples.

Meisel and Michalopoulos (1973) studied a slightly different setting, which we would call more common today: multiple examples can have the same class labels, and even numerical attributes are allowed. The first step of this approach builds an overfitting decision tree that classifies all examples correctly; all possible boundaries in the discretization achieved by this tree partition the feature space. The probability of an example is assumed to be the fraction of examples in its corresponding partition in feature space. The task is to find a 100% accurate decision tree with lowest expected cost, where every test has unit cost and examples are distributed according to the previously determined probabilities. The problem is solved by dynamic programming over all possible subsets of tests.

In 1977, it was shown by Payne and Meisel (1977) that Meisel's algorithm can also be applied for finding optimal decision trees under many other types of optimization criteria, for instance, for finding trees of minimal height or minimal numbers of nodes. Furthermore, techniques were presented to deal with the situation that some partitions do not contain any examples.

The dynamic programming algorithm of Payne and Meisel is very similar to our algorithm. The main difference is that we build a lattice of tests under a different set of constraints, and choose different datastructures to store the intermediate results of the dynamic programming. The clear link between the work of Payne and Meisel, and results obtained in the data mining and formal concept analysis communities, has not been observed before.

Independently, Schumacher and Sevcik (1976) studied the problem of converting decision tables into decision trees. A decision table is a table which contains (1) a row for every possible example in the feature space, and (2) a probability for every example. The problem is not to learn a predictor for unseen examples — there are no unseen examples —, but to compress the decision table into a compact representation that allows to retrieve the class of an example as quickly as possible. Again, a dynamic programming solution was proposed to compute an optimal tree.

Lew’s algorithm of 1978 (Lew, 1978) is a variation of the algorithm of Schumacher; the main difference is that in Lew’s algorithm it is possible to specify the input decision table in a condensed form, for instance, by using wildcards as attributes. Other extensions allow entries in the decision table to overlap, and to deal with data that is not binary.

More recently, pruning strategies of decision trees have been studied by Garafalakis, Hyun, Rastogi, and Shim (2003). DL8, and its early predecessors, can be conceived as an application of Garafalakis’ pruning strategy to another type of datastructure.

Related is also the work of Moore and Lee on the *ADtree* data structure (Moore & Lee, 1998). Both ADtrees and itemset lattices aim at speeding up the lookup of itemset frequencies during the construction of decision trees, where ADtrees have the benefit that they are computed without frequency constraint. However, this is achieved by not storing specializations of itemsets that are already relatively infrequent; for these itemsets subsets of the data are stored instead. For our bottom-up procedure to work it is necessary that all itemsets that fulfil the given constraints, are stored with associated information. This is not straightforwardly achieved in ADtrees.

The *tree*-relevancy constraint is closely related to the condensed representation of δ -free itemsets (Boulicaut, Bykowski, & Rigotti, 2003). Indeed, for $\delta = \text{minsup} \times |D|$ and $p(I) := (\text{freq}(I) \geq \text{minfreq})$, it can be shown that if an itemset is δ -free, it is also *tree*-relevant. DL8-CLOSED employs ideas that have also been exploited in the formal concept analysis (FCA) community and in closed itemset miners (Uno et al., 2004).

A popular topic in data mining is currently the selection of itemsets from a large set of itemsets found by a frequent itemset mining algorithm (Yan, Cheng, Han, & Xin, 2005; Knobbe & Ho, 2006; De Raedt & Zimmermann, 2007). DL8 can be seen as one such algorithm for selecting itemsets. It is however the first algorithm that outputs a well-known type of model, and provides accuracy guarantees for this model.

10 Conclusions

We presented DL8, an algorithm for finding decision trees that maximize an optimization criterion under constraints, and successfully applied this algorithm on a large number of datasets.

We showed that there is a clear link between DL8 and frequent itemset miners, which means that it is possible to apply many of the optimizations that have been

proposed for itemset miners also when mining decision trees under constraints. The investigation that we presented here is only a starting point in this direction; it is an open question how fast decision tree miners could become if they were thoroughly integrated with algorithms such as LCM or FP-Growth. Our investigations showed that high runtimes are however not as much a problem as the amount of memory required for storing huge amounts of itemsets. A challenging question for future research is what kind of condensed representations could be developed to represent the information that is used by DL8 more compactly.

The issue of efficiency remains important, as the applicability of our algorithm is limited by the amount of itemsets that can be stored. We mainly used two means to limit this amount: we used higher frequency thresholds and limited the number of bins in the binarization. In our experiments we found that these choices can, but not always do affect the predictive performance of decision trees learned by our algorithm. For the same frequency thresholds and the same discretization, we found that the trees learned by DL8 are usually significantly more accurate than trees learned by C4.5. When we ignore the constraints and compare the best settings of both algorithms, J48 performs significantly better in 55% of the datasets.

To support our claim that DL8 performs better under similar constraint settings, we also used DL8 to perform cost-based learning. In these experiments we again found that our algorithm can obtain better results than other algorithms, but in some cases also performs worse if the discretization is not fine-grained enough.

Still, our conclusion that trees mined under declarative constraints perform well both on training and test data, means that constraint-based tree miners deserve further study. Our experiments showed that an important question is how to deal with numerical attributes in a more integrated way. Incorporating and testing additional types of constraints, such as constraints on the discretisation, or decision tree depth, seems well worth investigating.

Despite these shortcomings, many open questions regarding the instability of decision trees, the influence of size constraints, heuristics, pruning strategies, and so on, may already be answered by further studies of the results of DL8. DL8's results could be compared to many other types of decision tree learners (Page & Ray, 2003b; Provost & Domingos, 2003).

Given that DL8 can be seen as a relatively cheap type of post-processing on a set of itemsets, DL8 suits itself perfectly for interactive data mining on stored sets of patterns. This means that DL8 might be a key component of inductive databases (Imielinski & Mannila, 1996) that contain both patterns and data.

Acknowledgements Siegfried Nijssen was supported by the EU FET IST project “Inductive Querying”, contract number FP6-516169. Élisà Fromont was supported through the GOA project 2003/8, “Inductive Knowledge bases”, and the FWO project “Foundations for inductive databases”. The authors thank Luc De Raedt and Hendrik Blockeel for many interesting discussions; Ferenc Bodon and Bart Goethals for putting online their implementations of respectively APRIORI and ECLAT, which we used to implement DL8, and Takeaki Uno for providing LCM. We also wish to thank Daan Fierens for preprocessing the data that we used in our experiments.

References

- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In *Advances in knowledge discovery and data mining* (p. 307-328). AAAI/MIT Press.
- Angelopoulos, N., & Cussens, J. (2005). Exploiting informative priors for bayesian classification and regression trees. In L. P. Kaelbling & A. Safiotti (Eds.), *Ijcai* (p. 641-646). Professional Book Center.
- Bayardo, R. J., Goethals, B., & Zaki, M. J. (Eds.). (2004). *Fimi '04, proceedings of the ieee icdm workshop on frequent itemset mining implementations, brighton, uk*. CEUR-WS.org.
- Boulıcaut, J.-F., Bykowski, A., & Rigotti, C. (2003). Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1), 5–22.
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, California, U.S.A.: Wadsworth Publishing Company.
- Buntine, W. (1992). Learning classification trees. In *Statistics and computing* 2 (pp. 63–73).
- Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, 93(443), 935–947.
- De Raedt, L., & Zimmermann, A. (2007). Constraint-based pattern set mining. In SIAM International Conference on Data Mining.
- Esmeir, S., & Markovitch, S. (2007). Anytime induction of cost-sensitive trees. In *Proceedings of the 21st annual conference on neural information processing systems (nips-2007)* (p. To appear). Vancouver, B.C., Canada.
- Friedman, A., Schuster, A., & Wolff, R. (2006). -anonymous decision tree induction. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *Pkdd* (Vol. 4213, pp. 151–162). Springer.
- Ganter, B., & Wille, R. (1999). *Formal concept analysis: Mathematical foundations*. Springer-Verlag.
- Garey, M. R. (1972, September). Optimal binary identification procedures. *SIAM Journal of Applied Mathematics*, 23(2), 173-186.
- Garofalakis, M. N., Hyun, D., Rastogi, R., & Shim, K. (2003). Building decision trees with constraints. *Data Min. Knowl. Discov.*, 7(2), 187-214.
- Han, J., Pei, J., & Yin, Y. (2000, 05). Mining frequent patterns without candidate generation. In W. Chen, J. Naughton, & P. A. Bernstein (Eds.), *2000 acm sigmod intl. conference on management of data* (pp. 1–12). ACM Press.
- Hyafil, L., & Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1), 15-17.
- Imielinski, T., & Mannila, H. (1996). A database perspective on knowledge discovery. *Comm. Of The Acn*, 39, 58–64.
- Knobbe, A., & Ho, E. (2006). Maximally informative k -itemsets and their efficient discovery. In *Proceeding of the 12th acm sigkdd international conference on knowledge discovery in data mining (kdd'06)* (pp. 237–244).

- Lew, A. (1978). Optimal conversion of extended-entry decision tables with general cost criteria. *Commun. ACM*, 21(4), 269–279.
- Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *Knowledge discovery and data mining* (p. 80-86).
- Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). *l*-diversity: Privacy beyond *k*-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), 3.
- Meisel, W. S., & Michalopoulos, D. (1973). A partitioning algorithm with application in pattern classification and the optimization of decision tree. *IEEE Trans. Comput.*, C-22, 93-103.
- Mielikäinen, T., Panov, P., & Dzeroski, S. (2006). Itemset support queries using frequent itemsets and their condensed representations. In L. Todorovski, N. Lavrac, & K. P. Jantke (Eds.), *Discovery science* (Vol. 4265, pp. 161–172). Springer.
- Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.
- Moore, A., & Lee, M. S. (1998, March). Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8, 67-91.
- Murphy, P. M., & Pazzani, M. J. (1997). Exploring the decision forest: an empirical investigation of Occam's razor in decision tree induction. In *Computational learning theory and natural learning systems* (Vol. IV: Making Learning Systems Practical, pp. 171–187). MIT Press.
- Nadeau, C., & Bengio, Y. (2003). Inference for the generalization error. *Mach. Learn.*, 52(3), 239-281.
- Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases*.
- Page, D., & Ray, S. (2003a). Skewing: An efficient alternative to lookahead for decision tree induction. In G. Gottlob & T. Walsh (Eds.), *Ijcai* (p. 601-612). Morgan Kaufmann.
- Page, D., & Ray, S. (2003b). Skewing: An efficient alternative to lookahead for decision tree induction. In *Proceedings of the eighteenth international joint conference on artificial intelligence (ijcai'03)* (pp. 601–612).
- Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1), 25–46.
- Payne, H. J., & Meisel, W. S. (1977). An algorithm for constructing optimal binary decision trees. *IEEE Trans. Computers*, 26(9), 905-916.
- Provost, F., & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, 52, 199–215.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.
- Quinlan, J. R., & Cameron-Jones, R. M. (1995). Oversearching and layered search in empirical learning. In *IJCAI* (p. 1019-1024).
- Samarati, P. (2001). Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6), 1010–1027.
- Schumacher, H., & Sevcik, K. C. (1976). The synthetic approach to decision table conversion. *Commun. ACM*, 19(6), 343–351.

- Sweeney, L. (2002). k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 557–570.
- Turney, P. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2, 369–409.
- Uno, T., Kiyomi, M., & Arimura, H. (2004). Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *Fimi '04, proceedings of the ieee icdm workshop on frequent itemset mining implementations, brighton, uk, november 1* (Vol. 126). CEUR-WS.org.
- Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* (2nd Edition ed.). San Francisco: Morgan Kaufmann.
- Yan, X., Cheng, H., Han, J., & Xin, D. (2005). Summarizing itemset patterns: a profile-based approach. In *Proceeding of the 11th acm sigkdd international conference on knowledge discovery in data mining (kdd'05)* (pp. 314–323).
- Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). *New algorithms for fast discovery of association rules* (Tech. Rep. No. TR651).

Bayes Optimal Classification for Decision Trees

Siegfried Nijssen

SIEGFRIED.NIJSEN@CS.KULEUVEN.BE

K.U. Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

We present an algorithm for exact Bayes optimal classification from a hypothesis space of decision trees satisfying leaf constraints. Our contribution is that we reduce this classification problem to the problem of finding a rule-based classifier with appropriate weights. We show that these rules and weights can be computed in linear time from the output of a modified frequent itemset mining algorithm, which means that we can compute the classifier in practice, despite the exponential worst-case complexity. In experiments we compare the Bayes optimal predictions with those of the maximum a posteriori hypothesis.

1. Introduction

We study the problem of Bayes optimal classification for density estimation trees. A density estimation tree in this context is a decision tree which has a probability density for a class attribute in each of its leaves. One can distinguish two Bayesian approaches to density estimation using a space of such trees.

In the first approach a single *maximum a posteriori* (MAP) density estimation tree is identified first:

$$T = \arg \max_T P(T|\mathbf{X}, \vec{y}),$$

where $\mathbf{X}$ and $\vec{y}$ together constitute the training data. The posterior probability $P(T|\mathbf{X}, \vec{y})$ of a hypothesis T is usually the product of a prior and a likelihood. The MAP hypothesis can then be used to classify a test example x' using the densities in the leaves.

The second approach is to marginalize over all possible trees, instead of preferring a single one:

$$\arg \max_c P(c|x', \mathbf{X}, \vec{y}) = \arg \max_c \sum_T P(c|x', T)P(T|\mathbf{X}, \vec{y}).$$

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

Predictions that are performed using this second approach are called *Bayes optimal predictions*. It has been claimed that “no single tree classifier using the same prior knowledge as an optimal Bayesian classifier can obtain better performance on average” (Mitchell, 1997). The Bayesian point of view is that Bayesian averaging cancels out the effects of overfitted models (Buntine, 1990), and “solves” overfitting problems.

This claim was challenged by Domingos (2000). Domingos demonstrated experimentally that an ensemble of decision trees that are weighted according to posterior probabilities performs worse than uniformly weighted hypotheses. It was found that one overfitting tree usually dominates an ensemble.

However, these results were obtained by sampling from the hypothesis space. Even though Domingos argued that similar issues should also occur in the truly optimal approach, this claim could not be checked in practice as the exact computation of Bayes optimal predictions was considered to be impractical. Indeed, in (Chipman et al., 1998) it was already claimed that “exhaustive evaluation ... over all trees will not be feasible, except in trivially small problems, because of the sheer number of trees”. Similar claims were made in other papers studying Bayesian tree induction (Buntine, 1992; Chipman et al., 1998; Angelopoulos & Cussens, 2005; Oliver & Hand, 1995), and have led to the use of sampling techniques such as Markov Chain Monte Carlo sampling.

In this paper we present an algorithm that can be used to evaluate Domingos’ claim in a reasonable number of non-trivial settings. Our algorithm allows us to exactly compute the Bayes optimal predictions given priors that assign non-zero probability to trees that satisfy certain constraints. An example of a constraint is that every leaf covers a significant number of examples; this constraint has been used very often in the literature (Buntine, 1992; Quinlan, 1993; Chipman et al., 1998; Angelopoulos & Cussens, 2005; Oliver & Hand, 1995).

Our algorithm is an extension of our earlier work, in

which we developed the DL8 algorithm for determining one tree that maximizes accuracy (Nijssen & Fromont, 2007). DL8 is based on dynamic programming on a pre-computed lattice of itemsets, and scans these itemsets decreasing in size. Its time complexity is linear in the size of the lattice. In this paper we extend this algorithm to a Bayesian setting. From a technical point of view, the main contribution is that we prove that a different pass over the lattice allows us to perform Bayes optimal predictions without increasing the asymptotic complexity of building the lattice.

The task that our algorithm addresses is similar to the task addressed in (Cleary & Trigg, 1998). Compared to this earlier work, we study the more common Dirichlet priors also considered in (Chipman et al., 1998; Angelopoulos & Cussens, 2005); furthermore, by exploiting the link to itemset mining, our algorithm is more efficient, and its results are more interpretable.

The paper is organized as follows. Notation and concepts are introduced in Section 2. Bayes optimal classification is formalized in Section 3. We show how to map this problem to the problem of finding itemsets and building a classifier with weighted rules in Section 4. Experiments are performed in Section 5.

2. Preliminaries

Before we are ready to formalize our problem and our proposed solution, we require some notation. We restrict ourselves to binary data; we assume that data is converted in this form in a preprocessing step. The data is stored in binary matrix $\mathbf{X}$, of which each row $\vec{x}_k$ corresponds to one example. Every example $\vec{x}_k$ has a class label y_k out of a total number of C class labels. Class labels are collected in a vector $\vec{y}$.

We assume that the reader is familiar with the concept of decision trees (see (Breiman et al., 1984; Quinlan, 1993) for details). Essential in our work is a link between decision trees and *itemsets*. Itemsets are a concept that was introduced in the data mining literature (Agrawal et al., 1996). If $\mathcal{I}$ is a domain of items, $I \subseteq \mathcal{I}$ is an itemset. In our case, we assume that we have *two* types of items: for every attribute there is a positive item i that represents a positive value, and a negative item $\neg i$ that represents a negative value. An example $\vec{x}$ can be represented as an itemset

$$\{i|x_i = 1\} \cup \{\neg i|x_i = 0\}.$$

Thus, for a data matrix with n columns, we have that $\mathcal{I} = \mathcal{I}_{pos} \cup \mathcal{I}_{neg}$, where $\mathcal{I}_{pos} = \{1, 2, \dots, n\}$ and $\mathcal{I}_{neg} = \{\neg 1, \neg 2, \dots, \neg n\}$. We overload the use of the $\subseteq$ operator: when I is an itemset, and $\vec{x}$ is an example,

we use $I \subseteq \vec{x}$ to denote that I is a subset of $\vec{x}$ after translating $\vec{x}$ into an itemset.

Every sequence of test outcomes in a decision tree, starting from the root of the tree to an arbitrary node deeper down the tree, can be represented as an itemset. For instance, a decision tree with B in the root, and A in its right-hand branch can be represented by:

$$T = \{\emptyset, \{B\}, \{\neg B\}, \{\neg B, A\}, \{\neg B, \neg A\}\}.$$

Every itemset in T corresponds to one node in the tree. By $\mathcal{T}$ we denote all subsets of $2^{\mathcal{I}}$ that represent decision trees. A decision tree structure is an element $T \in \mathcal{T}$. Consequently, when T is a decision tree we can write $I \in T$ to determine if the itemset I corresponds to a path occurring in the tree.

An itemset is an unordered set: given an itemset in a tree, we cannot derive from this itemset in which order its tests appear in the tree. This order can only be determined by considering all itemsets in a tree T .

We are not always interested in all nodes of a tree. The subset of itemsets that correspond to the leaves of a tree T will be denoted by $leaves(T)$; in our example,

$$leaves(T) = \{\{B\}, \{\neg B, A\}, \{\neg B, \neg A\}\}.$$

The most common example of a decision tree is the *classification tree*, in which every leaf is labeled with a single class. In a *density estimation tree*, on the other hand, we attach a class *distribution* to each leaf, represented by a vector $\vec{\theta}_I$; for each class c this vector contains the probability θ_{Ic} that examples $\vec{x} \supseteq I$ belong to class c . All the parameters of the leaves of a tree are denoted by Θ . The vectors in Θ are thus indexed by the itemsets representing leaves of the tree.

For the evaluation of a tree T on a binary matrix $\mathbf{X}$, it is useful to have a shorthand notation for the number of examples covered by a leaf:

$$f(I, \mathbf{X}) = |\{\vec{x}_k | \vec{x}_k \supseteq I\}|;$$

usually we omit the matrix $\mathbf{X}$ in our notation, as we assume the training data to be fixed. We call $f(I, \mathbf{X})$ the *frequency* of I . Class-based frequency is given by:

$$f_c(I, \mathbf{X}, \vec{y}) = |\{\vec{x}_k | \vec{x}_k \supseteq I, y_k = c\}|.$$

The *frequent itemset mining* problem is the problem of finding all $I \subseteq \mathcal{I}$ such that $f(I) \geq \gamma$, for a given threshold γ . Many algorithms for computing this set exist (Agrawal et al., 1996; Goethals & Zaki, 2003). They are based on the property that the frequency constraint is *anti-monotonic*. A binary constraint p on itemsets is called anti-monotonic iff $\forall I' \subseteq I : p(I) =$

$true \implies p(I') = true$. Consequently, these algorithms do not need to search through all supersets $I' \supseteq I$ of an itemset I that is found to be infrequent.

One application of itemsets is in the construction of rule-based classifiers (CMAR (Li et al., 2001) is an example). Many rule-based classifiers traverse rules sequentially when predicting examples. Here, we study a rule-based classifier that derives a prediction from all rules through voting. Such a classifier can be seen as a subset $\mathcal{P} \subseteq 2^{\mathcal{I}}$ of itemsets, each of which has a weight vector $\vec{w}(I)$. We predict an example $\vec{x}$ by computing

$$\arg \max_c \sum_{\{I \in \mathcal{P} | I \subseteq \vec{x}\}} w_c(I),$$

where we thus pick the class that gets most votes of all rules in the ruleset; each rule votes with a certain weight on each class. The aim of this paper is to show that we can derive a set of itemsets $\mathcal{P}$ and weights $\vec{w}(I)$ for all $I \in \mathcal{P}$ such that the predictions of the rule-based classifier equal those of a Bayes optimal classifier. The rules in $\mathcal{P}$ represent all paths that can occur in trees in the hypothesis space.

3. Problem Specification

In this section we formalize the problem of Bayes optimal classification for a hypotheses space of decision trees. Central in the Bayesian approach is that we first define the probability of the data given a tree structure T and parameters Θ :

$$P(\vec{y} | \mathbf{X}, T, \Theta) = \prod_{I \in \text{leaves}(T)} \prod_{c=1}^C (\theta_{Ic})^{f_c(I)}$$

In Bayes optimal classification we are interested in finding for a particular example $\vec{x}'$ the class y' which maximizes the probability

$$\begin{aligned} y' &= \arg \max_c P(c | \vec{x}', \mathbf{X}, \vec{y}) \\ &= \arg \max_c \sum_{T \in \mathcal{T}} \int_{\Theta} P(c | \vec{x}', T, \Theta) P(T, \Theta | \mathbf{X}, \vec{y}) d\Theta, \end{aligned} \quad (1)$$

where we sum over the space of all decision trees and integrate over all possible distributions in the leaves of each tree. Applying Bayes' rule on the second term, and observing that Θ is dependent on the tree T , we can rewrite this into

$$\sum_{T \in \mathcal{T}} \int_{\Theta} P(c | \vec{x}', T, \Theta) P(\vec{y} | T, \Theta, \mathbf{X}) P(\Theta | T, \mathbf{X}) P(T | \mathbf{X}) d\Theta; \quad (2)$$

in this formula $P(T | \mathbf{X})$ is the probability of a tree given that we have seen all data except the class labels.

Our method is based on the idea that we can constrain the space of decision trees by manipulating this term.

A first possibility is that we set $P(T | \mathbf{X}) = 0$ if there is a leaf $I \in \text{leaves}(T)$ such that $f(I) < \gamma$, for a frequency threshold γ . We call such leaves *small leaves*. The class estimates of a small leaf are often unreliable, and it is therefore common in many algorithms to consider only large leaves.

Additionally, we can set $P(T | \mathbf{X}) = 0$ if the depth of the decision tree exceeds a predefined threshold.

Both limitations impose *hard constraints* on the trees that are considered to be feasible estimators. We denote trees in $\mathcal{T}$ that satisfy all hard constraints by $\mathcal{L}$.

In the simplest case we can assume a uniform distribution on the trees that satisfy the hard constraints. Effectively, this would mean that we set $P(\Theta | T, \mathbf{X}) = 1$ in Equation 2 for all $T \in \mathcal{L}$. However, we will study a more sophisticated prior in this paper to show the power of our method. The aim of this prior, which was proposed in (Chipman et al., 1998), is to give more weight to smaller trees; it can be seen as a *soft constraint*. This prior is defined as follows.

$$P(T | \mathbf{X}) = \prod_{I \in T} P_{\text{node}}(I, T, \mathbf{X})$$

Here, the term $P_{\text{node}}(I, T, \mathbf{X})$ is defined as follows.

$$P_{\text{node}}(I, T, \mathbf{X}) = \begin{cases} P_{\text{leaf}}(I, \mathbf{X}), & \text{if } I \text{ is a leaf in } T; \\ P_{\text{intern}}(I, \mathbf{X}), & \text{if } I \text{ is internal in } T; \end{cases}$$

where

$$P_{\text{leaf}}(I, \mathbf{X}) = \begin{cases} 0, & \text{if } f(I) < \gamma \text{ or } |I| > \delta; \\ 1, & \text{else if } |I| = \delta \text{ or } e(I) = 0; \\ 1 - \alpha(1 + |I|)^{-\beta}, & \text{otherwise;} \end{cases}$$

and

$$P_{\text{intern}}(I, \mathbf{X}) = \begin{cases} 0, & \text{if } f(I) < \gamma \text{ or } |I| \geq \delta \\ & \text{or } e(I) = 0; \\ \alpha(1 + |I|)^{-\beta} / e(I), & \text{otherwise;} \end{cases}$$

Here $e(I)$ is the size of the set $\{i \in \mathcal{I}_{\text{pos}} | f(I \cup i) \geq \gamma \wedge f(I \cup \neg i) \geq \gamma\}$, which consists of all possible tests that can still be performed to split the examples covered by itemset I .

The term $\alpha(1 + |I|)^{-\beta}$ makes it less likely that nodes at a higher depth are split. The term $e(I)$ determines how many tests are still available if a test is to be performed. We assume that tests are apriori equally likely, independent of the order in which the previous tests on the path have been performed. An alternative could be to give more likelihood to tests that are well-balanced.

Note that P_{leaf} and P_{intern} are computed for an itemset I almost independently from the tree T : we only need to know if I is a leaf or not.

As common in Bayesian approaches, we assume that the parameters in every leaf of the tree are Dirichlet distributed with the same parameter vector $\vec{\alpha}$, i.e.

$$P(\Theta|T, \mathbf{X}) = P(\Theta|T) = \prod_{I \in \text{leaves}(T)} \text{Dir}(\vec{\theta}_I|\vec{\alpha}),$$

where

$$\text{Dir}(\vec{\theta}_I|\vec{\alpha}) = \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \prod_c \theta_{Ic}^{\alpha_c - 1},$$

and Γ is the gamma function.

Finally, it can be seen that

$$P(c|\vec{x}', T, \Theta) = \theta_{I(T, \vec{x}')c},$$

where $I(T, \vec{x}')$ is the leaf of T for which $I \subseteq \vec{x}'$.

We now have formalized all terms of Equation 2.

4. Solution Strategy

An essential step in our solution strategy is the construction of the set

$$\mathcal{P} = \{I|T \in \mathcal{L}, I \in T\},$$

which consists of all itemsets in trees that satisfy the hard constraints. Only these paths are needed when we wish to compute the posterior distribution over class labels, and are used as rules in our rule-based classifier. The weights of these rules are obtained by rewriting the Bayesian optimization criterion for a test example $\vec{x}'$ (Equation 1) as

$$\arg \max_c \sum_{\{I \in \mathcal{P} | I \subseteq \vec{x}'\}} w_c(I)$$

where

$$w_c(I) = \sum_{T \in \mathcal{L}, \text{ has leaf } I} \left(\prod_{I \in T} P_{\text{node}}(I, T, \mathbf{X}) \right) \left(\int_{\Theta} \theta_{Ic} \prod_{I' \in \text{leaves}(T)} \text{Dir}(\vec{\theta}_{I'}|\vec{\alpha}) \prod_c \theta_{I'c}^{f_c(I')} d\Theta \right). \quad (3)$$

The idea behind this rewrite is that the set of all trees in $\mathcal{L}$ can be partitioned by considering in which leaf a test example ends up. An example ends in exactly one leaf in every tree, and thus every tree belongs to one partition as determined by that leaf. We sum first over

all possible leaves that can contain the example, and then over all trees having that leaf. The weights of the rules in our classifier consist of the terms $w_c(I)$, and will be computed from the training data in the training phase; the sum of the weights $w_c(I)$ is computed for a test example in the classification phase.

This rewrite shows that in the training phase we need to compute weights for all itemsets that are in $\mathcal{P}$. We will discuss now how to compute these.

In the formulation above we multiply over all leaves, including the leaf that we assumed the example ended up in. Taking this special leaf apart we obtain:

$$w_c(I) = W_c(I) \sum_{T \in \mathcal{L}, \text{ has leaf } I} \prod_{I' \in T, I' \neq I} V(I', T); \quad (4)$$

where

$$W_c(I) = P_{\text{leaf}}(I, \mathbf{X}) \int_{\vec{\theta}_I} \theta_{Ic} \text{Dir}(\vec{\theta}_I|\vec{\alpha}) \prod_c \theta_{Ic}^{f_c(I)} d\vec{\theta}_I$$

and

$$V(I, T) = \begin{cases} P_{\text{intern}}(I, \mathbf{X}) & \text{if } I \text{ is internal in } T; \\ P_{\text{leaf}}(I, \mathbf{X}) \int_{\vec{\theta}_I} \text{Dir}(\vec{\theta}_I|\vec{\alpha}) \prod_c \theta_{Ic}^{f_c(I)} d\vec{\theta}_I, & \text{otherwise.} \end{cases}$$

This rewrite is correct due to the fact that we can move the integral of Equation 3 within the product over the leaves: the parameters of the leaves are independent from each other.

Let us write the integrals in closed form. First consider $W_c(I)$. As the Dirichlet distribution is the conjugate prior of the binomial distribution, we have

$$W_c(I) = P_{\text{leaf}}(I, \mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \int_{\vec{\theta}_I} \theta_{Ic} \prod_c \theta_{Ic}^{\alpha_c - 1 + f_c(I)} d\vec{\theta}_I = P_{\text{leaf}}(I, \mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \frac{\prod_c \Gamma(\alpha_c + f'_c(I))}{\Gamma(\sum_c \alpha_c + f'_c(I))}$$

Here $f'_{c'}(I) = f_{c'}(I)$ if $c \neq c'$, else $f'_{c'}(I) = f_{c'}(I) + 1$.

Similarly, we can compute $V(I, T)$ as follows.

$$V(I, T) = \begin{cases} V_{\text{intern}}(I) = P_{\text{intern}}(I, \mathbf{X}), & \text{if } I \text{ is internal in } T; \\ V_{\text{leaf}}(I) = P_{\text{leaf}}(I, \mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \frac{\prod_c \Gamma(\alpha_c + f_c(I))}{\Gamma(\sum_c \alpha_c + f_c(I))}, & \text{otherwise.} \end{cases}$$

The remaining question is now how to avoid summing all trees of Equation 4 explicitly. In the following, we will derive a dynamic programming algorithm to

implicitly compute this sum. We use a variable that is defined as follows.

$$u(I) = \sum_{T \in \mathcal{L}(I)} \prod_{I' \in T} V(I', T); \quad (5)$$

Here we define $\mathcal{L}(I)$ as follows:

$$\mathcal{L}(I) = \{\{I' \in T \mid I' \supseteq I\} \mid \text{all } T \in \mathcal{L} \text{ for which } I \in T\};$$

thus, $\mathcal{L}(I)$ consists of all subtrees that can be put below an itemset I while satisfying the hard constraints. As usual, we represent a subtree by listing all its paths.

For this variable we will first prove the following.

Theorem 1. *The following recursive relation holds for $u(I)$:*

$$u(I) = V_{leaf}(I) + \sum_{i \in \mathcal{I}_{pos} \text{ s.t. } I \cup i, I \cup \neg i \in \mathcal{P}} V_{intern}(I) u(I \cup i) u(I \cup \neg i).$$

Proof. We prove this by induction. Assume that for all itemsets $|I| > k$ our definition holds. Let us fill in our definition in the recursive formula, then we get:

$$u(I) = V_{leaf}(I) + \sum_{i \in \mathcal{I}_{pos}, \text{ s.t. } I \cup i, I \cup \neg i \in \mathcal{P}} \sum_{T \in \mathcal{L}(I \cup i)} \sum_{T' \in \mathcal{L}(I \cup \neg i)} V_{intern}(I) \prod_{I' \in T} V(I', T) \prod_{I' \in T'} V(I', T');$$

This can be written as Equation 5 to prove our claim: the term for V_{leaf} corresponds to the possibility that I is a leaf, the first sum passes over all possible tests if the node is internal, the second and third sum traverse all possible left-hand and right-hand subtrees; the product within the three sums is over all nodes in each resulting tree. $\square$

We can use this formula to write $w_c(I)$ as follows.

Theorem 2. *The formula $w_c(I)$ can be written as:*

$$w_c(I) = W_c(I) \sum_{\pi \in \Pi(I)} \prod_{i=1}^{|I|} V_{intern}(\{\pi_1, \dots, \pi_{i-1}\}) u(\{\pi_1, \dots, \pi_{i-1}, \neg \pi_i\}).$$

Here, $\Pi(I)$ contains all permutations $(\pi_1, \dots, \pi_n)$ of the items in I for which it holds that $\forall 1 \leq i \leq n : \{\pi_1, \dots, \pi_i\}, \{\pi_1, \dots, \pi_{i-1}, \neg \pi_i\} \in \mathcal{P}$.

Proof. The set of permutations $\Pi(I)$ consists of all (ordered) paths that can be constructed from the items in I and that fulfill the constraints on size and frequency. Each tree $T \in \mathcal{L}$ with $I \in T$ must have exactly one of these paths. Given one such path, Equation 4 requires us to sum over all trees that contain this path. Each tree in this sum consists of a particular choice of subtrees for each sidebranch of the path. Every node in a tree $T \in \mathcal{L}$ with $I \in T$ is either (1) part of the path to node I or (2) part of a sidebranch; this means that we can decompose the product $\prod_{I' \in T, I' \neq I} V(I', T)$, which is part of Equation 4, into a product for nodes in sidebranches, and a product for nodes on the path to I . The term for nodes on the path is computed by

$$W_c(I) \prod_{i=1}^{|I|} V_{intern}(\{\pi_1, \dots, \pi_{i-1}\});$$

considering the side branches, $u(I)$ sums over all possible subtrees below sidebranches of the path $\{\pi_1, \dots, \pi_n\}$; using the product-of-sums rule that $\prod_{i=1}^n \sum_{j=1}^{m_i} \alpha_{ij} = \sum_{i_1=1}^{m_1} \dots \sum_{i_n=1}^{m_n} x_{1\alpha_1} \dots \alpha_{ni_n}$, where $\sum_{j=1}^{m_i} \alpha_{ij}$ corresponds to a u -value of a sidebranch, we can deduce that the product $\prod_{i=1}^{|I|} u(\{\pi_1, \dots, \pi_{i-1}, \neg \pi_i\})$ sums over all possible combinations of side branches. $\square$

Given their potentially exponential number it is undesirable to enumerate all permutations of item orders for every itemset. To avoid this let us define

$$v(I) = \sum_{\pi \in \Pi(I)} \prod_{k=1}^{|I|} V_{intern}(\{\pi_1, \dots, \pi_{k-1}\}) u(\{\pi_1, \dots, \pi_{k-1}, \neg \pi_k\}),$$

such that $w_c(I) = W_c(I) v(I)$.

Theorem 3. *The following recursive relation holds.*

$$v(I) = \begin{cases} 1, & \text{if } I = \emptyset, \\ \sum_{i \in I \text{ s.t. } I-i \cup \neg i \in \mathcal{P}} V_{intern}(I-i) u(I-i \cup \neg i) v(I-i), & \text{otherwise.} \end{cases}$$

Proof. This can be shown by induction: if we fill in our definition of $v(I)$ in the recursive formula we get

$$\sum_{i \in I \text{ s.t. } I-i \cup \neg i \in \mathcal{P}} V_{intern}(I-i) u(I-i \cup \neg i) \sum_{\pi \in \Pi(I-i)} \prod_{k=1}^{|I|-1} V_{intern}(\{\pi_1, \dots, \pi_{k-1}\}) u(\{\pi_1, \dots, \neg \pi_k\})$$

Both sums together sum exactly over all possible permutations of the items; the product is exactly over all terms of every permutation. $\square$

Algorithm 1 Compute Bayes Optimal Weights

input The set of itemsets $\mathcal{P}$ and for all $I \in \mathcal{P} : \vec{f}(I)$
output The weight vectors $\vec{w}(I)$ for all $I \in \mathcal{P}$
 1: % Bottom-up Phase
 2: Let n be the size of the largest itemset in $\mathcal{P}$
 3: **for** $k := n$ **downto** 0 **do**
 4: **for all** $I \in \mathcal{P}$ s.t. $|I| = k$ **do**
 5: $u[I] := V_{leaf}(I)$
 6: **for all** $i \in \mathcal{I}_{pos}$ s.t. $I \cup i, I \cup \neg i \in \mathcal{P}$ **do**
 7: $u[I] := u[I] + V_{intern}(I)u[I \cup i]u[I \cup \neg i]$
 8: **end for**
 9: **end for**
 10: **end for**
 11: % Top-down Phase
 12: $v[\emptyset] := 1$
 13: **for** $k := 1$ **to** n **do**
 14: **for all** $I \in \mathcal{P}$ s.t. $|I| = k$ **do**
 15: $v[I] := 0$
 16: **for all** $i \in I$ s.t. $I - i \cup \neg i \in \mathcal{P}$ **do**
 17: $v[I] := v[I] + V_{intern}(I - i)u[I - i \cup \neg i]v[I - i]$
 18: **end for**
 19: **for** $c := 1$ **to** C **do**
 20: $w_c[I] := W_c(I)v[I]$
 21: **end for**
 22: **end for**
 23: **end for**

A summary of our algorithm is given in Algorithm 1. The main idea is to apply the recursive formulas for $u(I)$ and $v(I)$ to perform dynamic programming in two phases: one bottom-up phase to compute the $u(I)$ values, and one top-down phase to compute the $v(I)$ values. Given appropriate data structures to perform the look-up of sub- and supersets of itemsets I , this procedure has complexity $\mathcal{O}(|\mathcal{P}|\delta C)$. As $|\mathcal{P}| = \mathcal{O}(n2^m)$, where n is the number of examples in the training data and m the number of attributes, this algorithm is exponential in the number of attributes.

After the run of this algorithm, for a test example we can compute $q_c(\vec{x}') = \sum_{I \subseteq \vec{x}'} w_c(I)$ for every c . We can easily compute the exact class probability estimates from this: $P(y' = c | \vec{x}', \mathbf{X}, \vec{y}) = \frac{q_c(\vec{x}')}{\sum_{c'} q_{c'}(\vec{x}')}.$

To compute the set $\mathcal{P}$ of paths in feasible trees, we can modify a frequent itemset miner (Goethals & Zaki, 2003), as indicated in our earlier work (Nijssen & Fromont, 2007). We replace the itemset lattice post-processing method of (Nijssen & Fromont, 2007) by the algorithm for computing Bayes optimal weights.

Compared to the OB1 algorithm of Cleary & Trigg (1998), the main advantage of our method is its clear link to frequent itemset mining. OB1 is based on the use of option trees, which have a worst case complexity of $\mathcal{O}(nm!)$ instead of $\mathcal{O}(n2^m)$. Cleary et al. suggest that sharing subtrees in option trees could improve performance; this exactly what our approach achieves

in a fundamental way. The link between weighted rule-based and Bayes optimal classification was also not made by Cleary et al., making the classification phase either more time or space complex. We can interpret predictions by our approach by listing the (maximal) itemsets that contribute most weight to a prediction.

5. Experiments

We do not perform a feasibility study here, as we did such a study in earlier work (Nijssen & Fromont, 2007).

We performed several experiments to determine the importance of the α and β parameters of the size prior. We found that the differences between values $\alpha, \beta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ were often not significant and choose $\alpha = 0.80$ and $\beta = 0.80$ as defaults. We also experimented with a uniform prior. We choose $\vec{\alpha} = (1.0, \dots, 1.0)$ as default for the Dirichlet prior. This setting is common in the literature.

All comparisons were tested using a corrected, two-tailed, paired t -test with a 95% confidence interval.

Artificial Data In our first set of experiments we use generated data. We use this data to confirm the influence of priors and the ability of the Bayes optimal classifier to recognize that data can best be represented by an ensemble of multiple trees.

A common approach is to generate data from a model and to compute how well a learning algorithm recovers this original model. In our setting this approach is however far from trivial, as it is hard to generate a realistic lattice of itemsets: Calders (2007) showed that it is NP-hard to decide if a set of itemset frequencies can occur at all in data. Hence we used an alternative approach. The main idea is that we wish to generate data such that different trees perform best on different parts of the data. We proceed as follows: we first generate n tree structures (in our experiments, all trees are complete trees of depth 7; the trees do not yet have class labels in their leaves); from these n trees we randomly generate a database of given size (4000 examples with 15 binary attributes in our experiments, without class labels). We make sure that every leaf in every tree has at least γ examples (3% of the training data in our experiments). Next, we iterate in a fixed order over these trees to assign classes to the examples in one leaf of each tree; in each tree we pick the leaf which has the largest number of examples without class, and assign a class to these examples, taking care that two adjacent leaves get different majority classes. We aim for pure leaves, but these are less likely for higher numbers of generating trees.

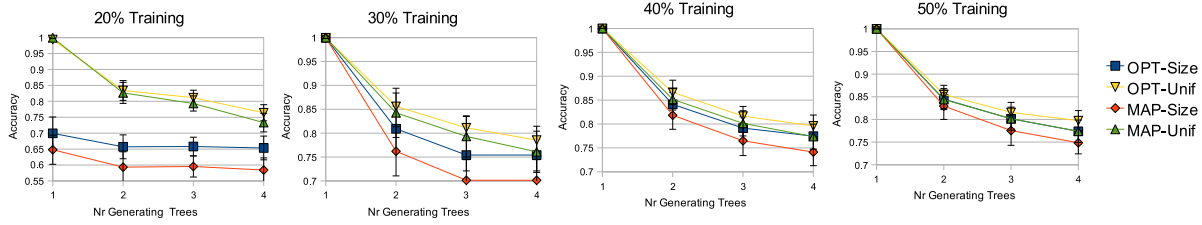


Figure 1. Results on artificial data.

The results of our experiments are reported in Figure 1. The accuracies in these experiments are computed for 20 randomly generated datasets. Each figure represents a different fraction of examples used as training data; remaining examples were as test data. The learners were run using the same depth and support constraints as used to generate the data.

We can learn the following from these experiments.

As all our datasets were created from trees with maximal height, the prior which prefers small trees performs worse than the one which assigns equal weight to all trees. If the amount of training data is small, the size prior forces the learner to prefer trees which are not 100% accurate for data created from one tree.

In all cases, the Bayes optimal approach is significantly more accurate than the corresponding MAP approach, except if the data was created using a single tree; in this case we observe that a single (correct) tree is dominating the trees in the ensembles.

The more training data we provide, the smaller the differences between the approaches are. For the correct prior the optimal approach has a better learning curve.

Additional experiments (not reported here) for other tree depths, dataset sizes and less pure leaves confirm the results above, although sometimes less pronounced.

UCI Data In our next set of experiments we determine the performance of our algorithm on common benchmark data, using ten-fold cross validation.

The frequency and depth constraints in our prior influence the efficiency of the search; too low frequency or too high depth constraints can make the search infeasible. Default values for δ that we considered were 4, 6 and ∞ ; for γ we considered 2, 15 and 50. We relaxed the constraints as much as was computationally possible; experiments (not reported here) show that this usually does not worsen accuracy.

As our algorithm requires binary data, numeric attributes were discretized in equifrequency bins. Only a relatively small number of 4 bins was feasible in all experiments; we used this value in all datasets to

avoid drawing conclusions after parameter overfitting. Where feasible within the range of parameters used, we added results for other numbers of bins to investigate the influence of discretization.

The experiments reported in Figure 1 help to provide more insight in the following questions:

- (Q1) Is a single tree dominating a Bayes optimal classifier in practice?
- (Q2) Are there significant differences between a uniform and a size-based prior in practice?
- (Q3) Is the optimal approach overfitting more in practice than the traditional approach, in this case Weka’s implementation of C4.5?
- (Q4) What is the influence of the 4-bin discretization?

To get an indication about (Q1) we compare the optimal and MAP predictions. We underlined those cases where there is a significant difference between optimal and MAP predictions. We found that in many cases there is indeed no significant difference between these two settings; in particular when hard constraints impose a high bias, such as in the Segment and Vote data, most predictions turn out to be equal. If there is a significant difference, the optimal approach is always the most accurate.

To answer (Q2) we highlighted in bold for each dataset the system that performs significantly better than all other systems. In many cases, the differences between the most accurate settings are not significant; however, our results indicate that a uniform prior performs slightly better than a size prior in the Bayes optimal case; the situation is less clear in the MAP setting.

Answering (Q3), we found not many significant differences between J48’s and Bayes optimal predictions in those cases where we did not have to enforce very hard constraints to turn the search feasible. This supports the claim of Domingos (2000) that Bayes optimal predictions are not really much better. However, our results also indicate that there is no higher risk of overfitting either. The optimal learner does not perform as well as J48 in those cases where the search is only feasible for high frequency or low depth constraints, and

Dataset				Accuracy				
	γ	δ	Bins	Opt - Size	MAP - Size	Opt - Unif	MAP - Unif	J48
Anneal	2	6	4	0.81±0.02	0.80±0.01	0.82±0.03	0.81±0.03	0.82±0.04
Anneal	15	6	10	0.86±0.04	0.86±0.04	0.86±0.04	0.85±0.04	0.89±0.03
Anneal	2	4	10	0.81±0.02	0.81±0.01	0.81±0.01	0.81±0.01	0.89±0.03
Balance	2	∞	4	<u>0.81±0.04</u>	0.76±0.06	0.84±0.03	0.83±0.03	0.76±0.06
Balance	2	∞	10	<u>0.80±0.03</u>	0.74±0.06	0.85±0.03	0.79±0.03	0.78±0.03
Heart	2	6	4	<u>0.82±0.07</u>	0.79±0.05	0.84±0.05	0.73±0.08	0.78±0.06
Heart	2	4	10	0.81±0.06	0.79±0.05	<u>0.81±0.06</u>	0.78±0.04	0.79±0.05
Vote	15	4	—	0.95±0.03	0.96±0.02	0.95±0.03	0.94±0.03	0.96±0.02
Segment	15	4	4	0.78±0.02	0.78±0.02	0.78±0.02	0.78±0.02	0.95±0.02
P-Tumor	2	∞	—	<u>0.40±0.05</u>	0.37±0.05	<u>0.43±0.05</u>	0.37±0.05	0.40±0.05
Yeast	2	6	4	0.52±0.03	0.52±0.03	<u>0.53±0.03</u>	0.52±0.03	0.54±0.05
Yeast	2	4	10	0.51±0.03	0.50±0.03	0.49±0.03	0.49±0.03	0.58±0.03
Diabetes	2	6	4	0.75±0.06	0.74±0.06	<u>0.75±0.05</u>	0.71±0.05	0.74±0.06
Diabetes	2	4	10	0.76±0.05	0.75±0.04	<u>0.77±0.05</u>	0.75±0.05	0.74±0.06
Ionosphere	15	4	4	0.87±0.06	0.87±0.06	0.87±0.06	0.87±0.05	0.86±0.07
Ionosphere	15	4	10	0.91±0.04	0.91±0.04	0.90±0.03	0.88±0.03	0.92±0.03
Vowel	50	6	4	0.42±0.04	0.40±0.04	0.41±0.07	0.38±0.05	0.78±0.04
Vehicle	50	6	4	<u>0.67±0.03</u>	0.66±0.03	0.66±0.03	0.65±0.03	0.70±0.04

Table 1. Experimental results on UCI data. A result is highlighted if it is the best in its row; significant winners of comparisons between MAP and Opt settings are underlined. Bins are not indicated for datasets without numeric attributes.

thus quite unrealistic priors; in (Nijssen & Fromont, 2007) we found that under the same hard constraints J48 is not able to find accurate trees either, and often finds even worse trees in terms of accuracy.

To provide more insight in (Q4), we have added results for different discretizations. In the datasets where we used harder constraints to make the search feasible, a negative effect on accuracy is observed compared to J48. Where the same hard constraints can be used we observe similar accuracies as in J48. The experiments do not indicate that a higher number of bins leads to increased risks of overfitting.

6. Conclusions

Our results indicate that instead of constructing the optimal MAP hypothesis, it is always preferable to use the Bayes optimal setting; even though we found many cases in which the claim of Domingos (2000) is confirmed and a single tree performs equally well, in those cases where there is a significant difference, the comparison is always in favor of the optimal setting. The computation of both kinds of hypothesis remains challenging if no hard constraints are applied, while incorrect constraints can have a negative impact.

Acknowledgments S. Nijssen was supported by the EU FET IST project “IQ”, contract number FP6-516169.

References

- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In *Advances in knowledge discovery and data mining*, 307–328.
- Angelopoulos, N., & Cussens, J. (2005). Exploiting informative priors for Bayesian classification and regression trees. *Proceedings of IJCAI* (pp. 641–646).
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Statistics/Probability Series. Belmont, California, U.S.A.
- Buntine, W. (1990). *A theory of learning classification rules*. Doctoral dissertation, Sydney.
- Buntine, W. (1992). Learning classification trees. *Statistics and Computing* 2 (pp. 63–73).
- Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, 93, 935–947.
- Cleary, J. G., & Trigg, L. E. (1998). *Experiences with OB1, an optimal Bayes decision tree learner* (Technical Report). University of Waikato.
- Domingos, P. (2000). Bayesian averaging of classifiers and the overfitting problem. *Proceedings of ICML* (pp. 223–230).
- Goethals, B., & Zaki, M. J. (Eds.). (2003). *Proceedings of the ICDM 2003 FIMI workshop*, vol. 90 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. *Proceedings of ICDM* (pp. 369–376).
- Nijssen, S., & Fromont, E. (2007). Mining optimal decision trees from itemset lattices. *Proceedings of KDD* (pp. 530–539).
- Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.
- Oliver, J. J., & Hand, D. J. (1995). On pruning and averaging decision trees. *Proceedings of ICML* (pp. 430–437).
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Compressing probabilistic Prolog programs

L. De Raedt · K. Kersting · A. Kimmig · K. Revoredo ·
H. Toivonen

Received: 27 September 2007 / Accepted: 9 October 2007 / Published online: 8 November 2007
Springer Science+Business Media, LLC 2007

Abstract ProbLog is a recently introduced probabilistic extension of Prolog (De Raedt, et al. in Proceedings of the 20th international joint conference on artificial intelligence, pp. 2468–2473, 2007). A ProbLog program defines a distribution over logic programs by specifying for each clause the probability that it belongs to a randomly sampled program, and these probabilities are mutually independent. The semantics of ProbLog is then defined by the success probability of a query in a randomly sampled program.

This paper introduces the theory compression task for ProbLog, which consists of selecting that subset of clauses of a given ProbLog program that maximizes the likelihood w.r.t. a set of positive and negative examples. Experiments in the context of discovering links in real biological networks demonstrate the practical applicability of the approach.

Editors: Stephen Muggleton, Ramon Otero, Simon Colton.

L. De Raedt · A. Kimmig
Departement Computerwetenschappen, K.U. Leuven, Celestijnenlaan 200A, bus 2402, 3001 Heverlee, Belgium

L. De Raedt
e-mail: luc.deraedt@cs.kuleuven.be

A. Kimmig (✉)
e-mail: angelika.kimmig@cs.kuleuven.be

K. Kersting · K. Revoredo
Institut für Informatik, Albert-Ludwigs-Universität, Georges-Köhler-Allee, Gebäude 079, 79110 Freiburg im Breisgau, Germany

K. Kersting
e-mail: kersting@informatik.uni-freiburg.de

K. Revoredo
e-mail: kate@cos.ufrj.br

H. Toivonen
Department of Computer Science, University of Helsinki, P.O. Box 68, 00014 Helsinki, Finland
e-mail: hannu.toivonen@cs.helsinki.fi

Keywords Probabilistic logic · Inductive logic programming · Theory revision · Compression · Network mining · Biological applications · Statistical relational learning

1 Introduction

The past few years have seen a surge of interest in the field of probabilistic logic learning or statistical relational learning (e.g. De Raedt and Kersting 2003; Getoor and Taskar 2007). In this endeavor, many probabilistic logics have been developed. Prominent examples include PHA (Poole 1993), PRISM (Sato and Kameya 2001), SLPs (Muggleton 1996), and probabilistic Datalog (pD) (Fuhr 2000). These frameworks attach probabilities to logical formulae, most often definite clauses, and typically impose further constraints to facilitate the computation of the probability of queries and simplify the learning algorithms for such representations.

Our work on this topic has been motivated by mining of large biological networks where edges are labeled with probabilities. Such networks of biological concepts (genes, proteins, phenotypes, etc.) can be extracted from public databases, and probabilistic links between concepts can be obtained by various prediction techniques (Sevon et al. 2006). Such networks can be easily modeled by our recent probabilistic extension of Prolog, called ProbLog (De Raedt et al. 2007). ProbLog is essentially Prolog where all clauses are labeled with the probability that they belong to a randomly sampled program, and these probabilities are mutually independent. A ProbLog program thus specifies a probability distribution over all possible non-probabilistic subprograms of the ProbLog program. The success probability of a query is then defined simply as the probability that it succeeds in a random subprogram. The semantics of ProbLog is not really new, it closely corresponds to that of pD (Fuhr 2000) and is closely related though different from the pure distributional semantics of (Sato and Kameya 2001), cf. also Sect. 8. However, the key contribution of ProbLog (De Raedt et al. 2007) is the introduction of an effective inference procedure for this semantics, which enables its application to the biological link discovery task.

The key contribution of the present paper is the introduction of the task of compressing a ProbLog theory using a set of positive and negative examples, and the development of an algorithm for realizing this. Theory compression refers to the process of removing as many clauses as possible from the theory in such a manner that the compressed theory explains the examples as well as possible. The compressed theory should be a lot smaller, and therefore easier to understand and employ. It will also contain the essential components of the theory needed to explain the data. The theory compression problem is again motivated by the biological application. In this application, scientists try to analyze large networks of links in order to obtain an understanding of the relationships amongst a typically small number of nodes. The idea now is to remove as many links from these networks as possible using a set of positive and negative examples. The examples take the form of relationships that are either interesting or uninteresting to the scientist. The result should ideally be a small network that contains the essential links and assigns high probabilities to the positive and low probabilities to the negative examples. This task is analogous to a form of theory revision (Wrobel 1996) where the only operation allowed is the deletion of rules or facts. The analogy explains why we have formalized the theory compression task within the ProbLog framework. Within this framework, examples are true and false ground facts, and the task is to find a small subset of a given ProbLog program that maximizes the likelihood of the examples.

This paper is organized as follows: in Sect. 2, the biological motivation for ProbLog and theory compression is discussed; in Sect. 3, the semantics of ProbLog are briefly reviewed;

in Sect. 4, the inference mechanism for computing the success probability of ProbLog queries as introduced by De Raedt et al. (2007) is reviewed; in Sect. 5, the task of probabilistic theory compression is defined; and an algorithm for tackling the compression problem is presented in Sect. 6. Experiments that evaluate the effectiveness of the approach are presented in Sect. 7, and finally, in Sect. 8, we discuss some related work and conclude.

2 Example: ProbLog for biological graphs

As a motivating application, consider link mining in networks of biological concepts. Molecular biological data is available from public sources, such as Ensembl,¹ NCBI Entrez,² and many others. They contain information about various types of objects, such as genes, proteins, tissues, organisms, biological processes, and molecular functions. Information about their known or predicted relationships is also available, e.g., that gene A of organism B codes for protein C, which is expressed in tissue D, or that genes E and F are likely to be related since they co-occur often in scientific articles. Mining such data has been identified as an important and challenging task (cf. Perez-Iratxeta et al. 2002).

For instance, a biologist may be interested in the potential relationships between a given set of proteins. If the original graph contains more than some dozens of nodes, manual and visual analysis is difficult. Within this setting, our goal is to automatically extract a relevant subgraph which contains the most important connections between the given proteins. This result can then be used by the biologist to study the potential relationships much more efficiently.

A collection of interlinked heterogeneous biological data can be conveniently seen as a weighted graph or network of biological concepts, where the weight of an edge corresponds to the probability that the corresponding nodes are related (Sevon et al. 2006). A ProbLog representation of such a graph could simply consist of probabilistic *edge/2* facts though finer grained representations using relations such as *codes/2*, *expresses/2* would also be possible. In a probabilistic graph, the strength of a connection between two nodes can be measured as the probability that a path exists between the two given nodes (Sevon et al. 2006).³ Such queries are easily expressed in ProbLog by defining the (non-probabilistic) predicate *path(N1, N2)* in the usual way, using probabilistic *edge/2* facts. Obviously, logic—and ProbLog—can easily be used to express much more complex possible relations. For simplicity of exposition we here only consider a simple representation of graphs, and in the future will address more complex applications and settings.

3 ProbLog: probabilistic Prolog

A ProbLog program consists—as Prolog—of a set of definite clauses. However, in ProbLog every clause c_i is labeled with the probability p_i that it is true.

Example 1 Within bibliographic data analysis, the similarity structure among items can improve information retrieval results. Consider a collection of papers $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$ and some pairwise similarities *similar(a, c)*, e.g., based on key word analysis. Two items X and Y

¹ www.ensembl.org.

² www.ncbi.nlm.nih.gov/Entrez/.

³ Sevon et al. (2006) view this strength or probability as the product of three factors, indicating the *reliability*, the *relevance* as well as the *rarity* (specificity) of the information.

are related(X, Y) if they are similar (such as a and c) or if X is similar to some item Z which is related to Y . Uncertainty in the data and in the inference can elegantly be represented by the attached probabilities:

1.0 : related(X, Y) : - similar(X, Y).
 0.8 : related(X, Y) : - similar(X, Z), related(Z, Y).
 0.9 : similar(a, c). 0.7 : similar(c, b).
 0.6 : similar(d, c). 0.9 : similar(d, b).

A ProbLog program $T = \{p_1 : c_1, \dots, p_n : c_n\}$ defines a probability distribution over logic programs $L \subseteq L_T = \{c_1, \dots, c_n\}$ in the following way:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i). \quad (1)$$

Unlike in Prolog, where one is typically interested in determining whether a query succeeds or fails, ProbLog specifies the probability that a query succeeds. The *success probability* $P(q|T)$ of a query q in a ProbLog program T is defined by

$$P(q|T) = \sum_{L \subseteq L_T} P(q, L|T) = \sum_{L \subseteq L_T} P(q|L) \cdot P(L|T), \quad (2)$$

where $P(q|L) = 1$ if there exists a θ such that $L \models q\theta$, and $P(q|L) = 0$ otherwise. In other words, the success probability of query q corresponds to the probability that the query q has a proof, given the distribution over logic programs.

4 Computing success probabilities

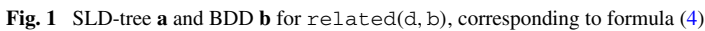
Given a ProbLog program $T = \{p_1 : c_1, \dots, p_n : c_n\}$ and a query q , the trivial way of computing the success probability $P(q|T)$ enumerates all possible logic programs $L \subseteq L_T$ (cf. (2)). Clearly this is infeasible for all but the tiniest programs. Therefore, the inference engine proceeds in two steps (cf. Fuhr 2000; De Raedt et al. 2007). The first step reduces the problem of computing the success probability of a ProbLog query to that of computing the probability of a monotone Boolean DNF formula. The second step then computes the probability of this formula.

4.1 ProbLog queries as DNF formulae

To be able to write down the DNF formula corresponding to a query, we use a Boolean random variable b_i for each clause $p_i : c_i \in T$, indicating whether c_i is in the logic program; i.e., b_i has probability p_i of being true. The probability of a particular proof involving clauses $\{p_1 : d_1, \dots, p_k : d_k\} \subseteq T$ is then the probability of the conjunctive formula $b_1 \wedge \dots \wedge b_k$. Since a goal can have multiple proofs, the probability that goal q succeeds equals the probability that the disjunction of these conjunctions is true. More formally, this yields:

$$P(q|T) = P\left(\bigvee_{b \in pr(q)} \bigwedge_{b_i \in cl(b)} b_i\right) \quad (3)$$

where we use the convention that $pr(q)$ denotes the set of proofs of the goal q and $cl(b)$ denotes the set of Boolean variables (clauses) used in the proof b .



Example 2 There are two proofs of `related(d,b)`, obtained using either the base case and one fact, or the recursive case and two facts. The corresponding SLD-tree is depicted in Fig. 1(a) and—as r_1 is always true—the Boolean formula is

$$(r_1 \wedge s_4) \vee (r_2 \wedge s_3 \wedge r_1 \wedge s_2) = s_4 \vee (r_2 \wedge s_3 \wedge s_2). \quad (4)$$

Computing the probability of DNF formulae is an NP-hard problem (Valiant 1979) even if all variables are independent, as they are in our case. There are several algorithms for transforming a disjunction of conjunctions into mutually disjoint conjunctions, for which the probability is obtained simply as a sum. One basic approach relies on the inclusion-exclusion principle from set theory. It requires the computation of conjunctive probabilities of all sets of conjunctions appearing in the DNF formula. This is clearly intractable in general, but is still used by the probabilistic Datalog engine Hyspirit (Fuhr 2000), which explains why inference with Hyspirit on about 10 or more conjuncts is infeasible according to (Fuhr 2000). In contrast, ProbLog employs the latest advances made in the manipulation and representation of Boolean formulae using binary decision diagrams (BDDs) (Bryant 1986), and is able to deal with up to 100000 conjuncts (De Raedt et al. 2007).

 Springer

and deleting redundant nodes until no further reduction is possible. A node is redundant iff the subgraphs rooted at its children are isomorphic. In Fig. 1(b), dashed edges indicate 0's and lead to low children, solid ones indicate 1's and lead to high children. The two leafs are called the 0- and 1-terminal node respectively.

BDDs are one of the most popular data structures used within many branches of computer science, such as computer architecture and verification, even though their use is perhaps not yet so widespread in artificial intelligence and machine learning (but see Chavira and Darwiche 2007 and Minato et al. 2007 for recent work on Bayesian networks using variants of BDDs). Since their introduction by Bryant (Bryant 1986), there has been a lot of research on BDDs and their computation. Many variants of BDDs and off the shelf systems exist today. An important property of BDDs is that their size is highly dependent on the variable ordering used. In ProbLog we use the general purpose package CUDD⁴ for constructing and manipulating the BDDs and leave the optimization of the variable reordering to the underlying BDD package. Nevertheless, the initial variable ordering when building the BDD is given by the order that they are encountered in the SLD-tree, i.e. from root to leaf. The BDD is built by combining BDDs for subtrees, which allows structure sharing.

Given a BDD, it is easy to compute the probability of the corresponding Boolean function by traversing the BDD from the root node to a leaf. Essentially, at each inner node, probabilities from both children are calculated recursively and combined afterwards, as indicated in Algorithm 1.

Algorithm 1 Probability calculation for BDDs.

PROBABILITY(input: BDD node n)

```

1  if  $n$  is the 1-terminal then return 1
2  if  $n$  is the 0-terminal then return 0
3  let  $p_n$  be the probability of the clause represented by  $n$ 's random variable
4  let  $h$  and  $l$  be the high and low children of  $n$ 
5   $prob(h) := \text{PROBABILITY}(h)$ 
6   $prob(l) := \text{PROBABILITY}(l)$ 
7  return  $p_n \cdot prob(h) + (1 - p_n) \cdot prob(l)$ 

```

4.3 An approximation algorithm

Since for realistic applications, such as large graphs in biology or elsewhere, the size of the DNF can grow exponentially, we have introduced in (De Raedt et al. 2007) an approximation algorithm for computing success probabilities along the lines of (Poole 1992).

The idea is that one can impose a depth-bound on the SLD-tree and use this to compute two DNF formulae. The first one, called *low*, encodes all successful proofs obtained at or above that level. The second one, called *up*, encodes all derivations till that level that have not yet failed, and, hence, for which there is hope that they will still contribute a successful proof. We then have that $P(low) \leq P(q|T) \leq P(up)$. This directly follows from the fact that $low \models d \models up$ where d is the Boolean DNF formula corresponding to the full SLD-tree of the query.

This observation is then turned into an approximation algorithm by using iterative deepening, where the idea is that the depth-bound is gradually increased until convergence, i.e., until $|P(up) - P(low)| \leq \delta$ for some small δ . The approximation algorithm is described in

⁴<http://vlsi.colorado.edu/~fabio/CUDD>.

more detail in (De Raedt et al. 2007), where it is also applied to link discovery in biological graphs.

Example 3 Consider the SLD-tree in Fig. 1(a) only till depth 2. In this case, d_1 encodes the left success path while d_2 additionally encodes the paths up to $\text{related}(c, b)$ and $\text{related}(b, b)$, i.e. $d_1 = (r_1 \wedge s_4)$ and $d_2 = (r_1 \wedge s_4) \vee (r_2 \wedge s_3) \vee (r_2 \wedge s_4)$. The formula for the full SLD-tree is given in (4).

5 Compressing ProbLog theories

Before introducing the ProbLog theory compression problem, it is helpful to consider the corresponding problem in a purely logical setting.⁵ Assume that, as in traditional theory revision (Wrobel 1996; Zelle and Mooney 1994), one is given a set of positive and negative examples in the form of true and false facts. The problem then is to find a theory that best explains the examples, i.e., one that scores best w.r.t. a function such as accuracy. At the same time, the theory should be *small*, that is it should contain less than k clauses. Rather than allowing any type of revision on the original theory, compression only allows for clause deletion. So, logical theory compression aims at finding a small theory that best explains the examples. As a result the compressed theory should be a better fit w.r.t. the data but should also be much easier to understand and to interpret. This holds in particular when starting with large networks containing thousands of nodes and edges and then obtaining a small compressed graph that consists of say 20 edges. In biological databases such as the ones considered in this paper, scientists can easily analyse the interactions in such small networks but have a very hard time with the large networks.

The *ProbLog Theory Compression Problem* is now an adaptation of the traditional theory revision (or compression) problem towards probabilistic Prolog programs. It can be formalized as follows:

Given

- a ProbLog theory S ;
- sets P and N of positive and negative examples in the form of independent and identically-distributed ground facts; and
- a constant $k \in \mathbb{N}$;

find a theory $T \subseteq S$ of size at most k (i.e. $|T| \leq k$) that has a maximum likelihood w.r.t. the examples $E = P \cup N$, i.e.,

$$T = \arg \max_{T \subseteq S \wedge |T| \leq k} \mathcal{L}(E|T), \quad \text{where } \mathcal{L}(E|T) = \prod_{e \in E} \mathcal{L}(e|T), \quad (5)$$

$$\mathcal{L}(e|T) = \begin{cases} P(e|T) & \text{if } e \in P, \\ 1 - P(e|T) & \text{if } e \in N. \end{cases} \quad (6)$$

So, we are interested in finding a small theory that maximizes the likelihood. Here, small means that the number of clauses should be at most k . Also, rather than maximizing the

⁵This can—of course—be modeled within ProbLog by setting the labels of all clauses to 1.

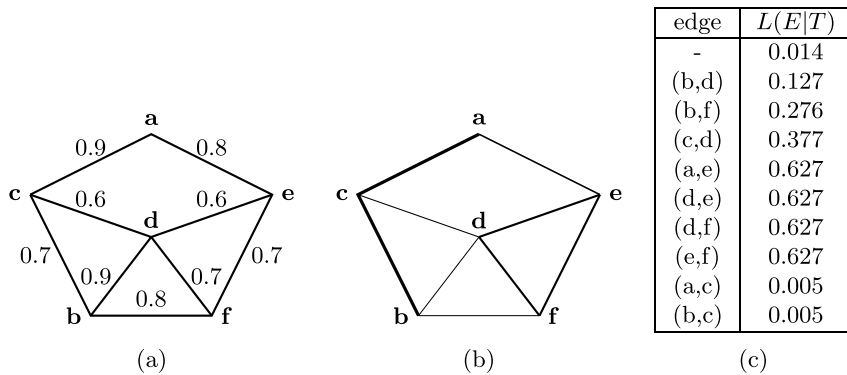


Fig. 2 Illustration of Examples 4 and 5: **a** Initial related theory. **b** Result of compression using positive example $\text{related}(a, b)$ and negative example $\text{related}(d, b)$, where edges are removed greedily (in order of increasing thickness). **c** Likelihoods obtained as edges are removed in the indicated order

accuracy as in purely logical approaches, in ProbLog we maximize the likelihood of the data. Here, a ProbLog theory T is used to determine a relative class distribution: it gives the probability $P(e|T)$ that any given example e is positive. (This is subtly different from specifying the distribution of (positive) examples.) The examples are assumed to be mutually independent, so the total likelihood is obtained as a simple product. For an optimal ProbLog theory T , the probability of the positives is as close to 1 as possible, and for the negatives as close to 0 as possible. However, because we want to allow misclassifications but with a high cost in order to avoid overfitting, to effectively handle noisy data, and to obtain smaller theories, we slightly redefine $P(e|T)$ in (6), as $\hat{P}(e|T) = \max(\min[1 - \epsilon, P(e|T)], \epsilon)$ for some constant $\epsilon > 0$ specified by the user. This avoids the possibility that the likelihood function becomes 0, e.g., when a positive example is not covered by the theory at all.

Example 4 Figure 2(a) graphically depicts a slightly extended version of our bibliographic theory from Example 1. Assume we are interested in items related to item b , and user feedback revealed that item a is indeed related to item b , but d is actually not. We might then use those examples to compress our initial theory to the most relevant parts, giving k as the maximal acceptable size.

6 The ProbLog theory compression algorithm

As already outlined in Fig. 2, the ProbLog compression algorithm removes one clause at a time from the theory, and chooses the clause greedily to be the one whose removal results in the largest likelihood. The more detailed ProbLog compression algorithm as given in Algorithm 2 works in two phases. First, it constructs BDDs for proofs of the examples using the standard ProbLog inference engine. These BDDs then play a key role in the second step where clauses are greedily removed, as they make it very efficient to test the effect of removing a clause.

Algorithm 2 ProbLog theory compression.

COMPRESS(input: $S = \{p_1 : c_1, \dots, p_n : c_n\}$, E, k, ϵ)

- 1 **for** all $e \in E$
- 2 **do call** APPROXIMATE(e, S, δ) to get $DNF(low, e)$ and $BDD(e)$
- 3 where $DNF(low, e)$ is the lower bound DNF formula for e
- 4 and $BDD(e)$ is the BDD corresponding to $DNF(low, e)$
- 5 $R := \{p_i : c_i \mid b_i \text{ (indicator for clause } i \text{) occurs in a } DNF(low, e)\}$
- 6 $BDD(E) := \bigcup_{e \in E} \{BDD(e)\}$
- 7 improves := true
- 8 **while** ($|R| > k$ or improves) and $R \neq \emptyset$
- 9 **do** $ll := \text{LIKELIHOOD}(R, BDD(E), \epsilon)$
- 10 $i := \arg \max_{i \in R} \text{LIKELIHOOD}(R - \{i\}, BDD(E), \epsilon)$
- 11 improves := ($ll \leq \text{LIKELIHOOD}(R - \{i\}, BDD(E), \epsilon)$)
- 12 **if** improves or $|R| > k$
- 13 **then** $R := R - \{i\}$
- 14 **return** R

More precisely, the algorithm starts by calling the approximation algorithm sketched in Sect. 4.3, which computes the DNFs and BDDs for lower and upper bounds. The compression algorithm only employs the lower bound DNFs and BDDs, since they are simpler and, hence, more efficient to use. All clauses used in at least one proof occurring in the (lower bound) BDD of some example constitute the set R of possible revision points. All other clauses do not occur in any proof contributing to probability computation, and can therefore be immediately removed; this step alone often gives high compression factors. Alternatively, if the goal is to minimize the changes to theory, rather than the size of the resulting theory, then all these other clauses should be left intact.

After the set R of revision points has been determined—and the other clauses potentially removed—the ProbLog theory compression algorithm performs a *greedy* search in the space of subsets of R . At each step, the algorithm finds that clause whose deletion results in the best likelihood score, and then deletes it. This process is continued until both $|R| \leq k$ and deleting further clauses does not improve the likelihood.

Compression is efficient, since the (expensive) construction of the BDDs is performed only once per example. Given a BDD for a query q (from the approximation algorithm) one can easily evaluate (in the revision phase) conditional probabilities of the form $P(q|T, b'_1 \wedge \dots \wedge b'_k)$, where the b'_i s are possibly negated Booleans representing the truth-values of clauses. To compute the answer using Algorithm 1, one only needs to reset the probabilities p'_i of the b'_i s. If b'_j is a positive literal, the probability of the corresponding variable is set to 1, if b'_j is a negative literal, it is set to 0. The structure of the BDD remains the same. When compressing theories by only deleting clauses, the b'_i s will be negative, so one has to set $p'_i = 0$ for all b'_i s. Figure 3 gives an illustrative example of deleting one clause.

Example 5 Reconsider the related theory from Example 4, Fig. 2(a), where edges can be used in both directions and a positive example $\text{related}(a, b)$ as well as a negative example $\text{related}(d, b)$ are given. With default probability $\epsilon = 0.005$, the initial likelihood of those two examples is 0.014. The greedy approach first deletes $0.9 : \text{similar}(d, b)$ and thereby increases the likelihood to 0.127. The probability of the positive example $\text{related}(a, b)$ is now 0.863 (was 0.928), and that of the negative example $\text{related}(d, b)$ is 0.853 (was 0.985). The final result is a theory just consisting of the two

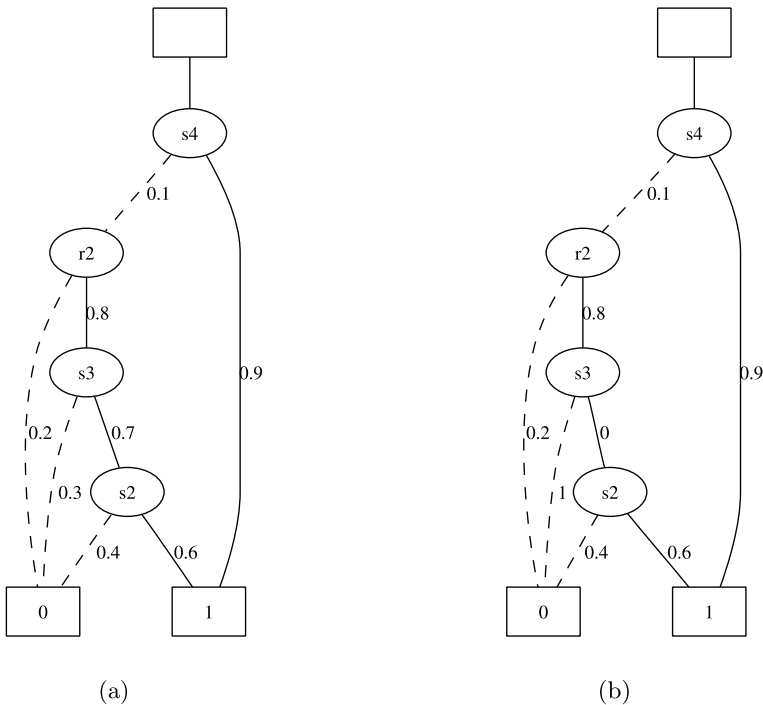


Fig. 3 Effect of deleting clause s_3 in Example 2: **a** Initial BDD: $P(e|T) = 0.9 + 0.1 \cdot 0.8 \cdot 0.7 \cdot 0.6 = 0.9336$. **b** BDD after deleting s_3 by setting its probability to 0: $P(e|T) = 0.9 + 0.1 \cdot 0.8 \cdot 0 \cdot 0.6 = 0.9$

edges $\text{similar}(a, c)$ and $\text{similar}(c, b)$ which leads to a total likelihood of 0.627, cf. Figs. 2(b) and 2(c).

7 Experiments

We performed a number of experiments to study both the quality and the complexity of ProbLog theory compression. The quality issues that we address empirically concern (1) the relationship between the amount of compression and the resulting likelihood of the examples, and (2) the impact of compression on clauses or hold-out test examples, where desired or expected results are known. We next describe two central components of the experiment setting: data and algorithm implementation.

7.1 Data

We performed tests primarily with real biological data. For some statistical analyses we needed a large number of experiments, and for these we used simulated data, i.e., random graphs with better controlled properties.

Real data Real biological graphs were extracted from NCBI Entrez and some other databases as described in (Sevon et al. 2006). Since NCBI Entrez alone has tens of millions of

objects, we adopted a two-staged process of first using a rough but efficient method to extract a graph G that is likely to be a supergraph of the desired final result, and then applying ProbLog on graph G to further identify the most relevant part.

To obtain a realistic test setting with natural example queries, we extracted two graphs related to four random Alzheimer genes (HGNC ids 620, 582, 983, and 8744). Pairs of genes are used as positive examples, in the form `? - path(gene_620, gene_582)`. Since the genes all relate to Alzheimer disease, they are likely to be connected via nodes of shared relevance, and the connections are likely to be stronger than for random pairs of nodes.

A smaller and a larger graph were obtained by taking the union of subgraphs of radius 2 (smaller graph) or 3 (larger graph) from the four genes and producing weights as described in (Sevon et al. 2006). In the smaller graph, gene 8744 was not connected to the other genes, and the corresponding component was left out. As a result the graph consists of 144 edges and 79 nodes. Paths between the three remaining Alzheimer genes were used as positive examples. All our tests use this smaller data and the three positive examples unless otherwise stated. The larger graph consists of 11530 edges and 5220 nodes and was used for scalability experiments. As default parameter values we used probability $\epsilon = 10^{-8}$ and interval width $\delta = 0.1$.

Simulated data Synthetic graphs with a given number of nodes and a given average degree were produced by generating edges randomly and uniformly between nodes. This was done under the constraint that the resulting graph must be connected, and that there can be at most one edge between any pair of nodes. The resulting graph structures tend to be much more challenging than in the real data, due to the lack of structure in the data. The default values for parameters were $\epsilon = 0.001$ and $\delta = 0.2$.

7.2 Algorithm and implementation

The approximation algorithm that builds BDDs is based on iterative deepening. We implemented it in a best-first manner: the algorithm proceeds from the most likely proofs (derivations) to less likely ones, as measured by the product of the probabilities of clauses in a proof. Proofs are added in batches to avoid repetitive BDD building.

We implemented the inference and compression algorithms in Prolog (Yap-5.1.0). For efficiency reasons, proofs for each example are stored internally in a prefix tree. Our ProbLog implementation then builds a BDD for an example by using the available CUDD operators, according to the structure of the prefix tree. The compiled result is saved for use in the revision phase.

7.3 Quality of ProbLog theory compression

The quality of compressed ProbLog theories is hard to investigate using a single objective. We therefore carried out a number of experiments investigating different aspects of quality, which we will now discuss in turn.

How does the likelihood evolve during compression? Figure 4(a) shows how the log-likelihood evolves during compression in the real data, using positive examples only, when all revision points are eventually removed. In one setting (black line), we used the three original paths connecting Alzheimer gene pairs as examples. This means that for each gene pair (g_1, g_2) we provided the example `path(g1, g2)`. To obtain a larger number of results, we artificially generated ten other test settings (grey lines), each time by randomly picking three

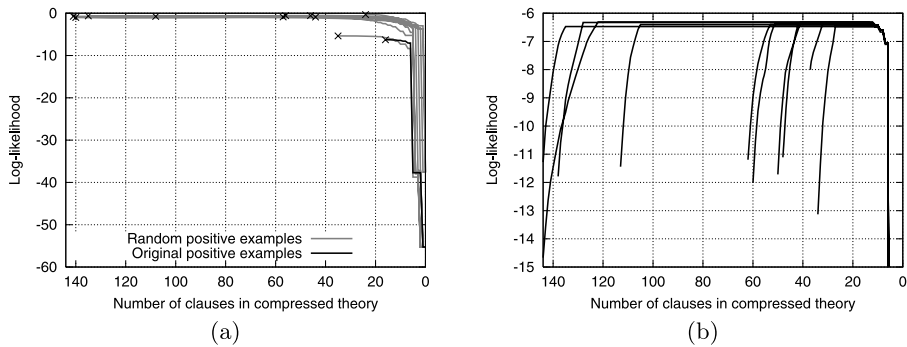


Fig. 4 Evolvement of log-likelihood for 10 test runs with positive examples only (a) and with both positive and negative examples (b). Different starting points of lines reflect the number of clauses in the BDDs used in theory compression

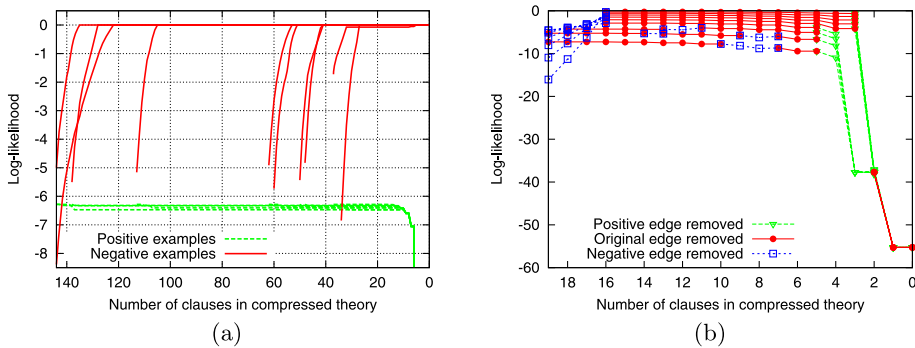


Fig. 5 Evolvement of log-likelihood for 10 test runs with both positive and negative examples (a). Evolvement of log-likelihood for settings with artificially implanted edges with negative and positive effects (b). In (b), linestyle indicates the type of edge that was removed in the corresponding revision step. Likelihoods in the middle reflect the probability on artificial edges: topmost curve is for $p = 0.9$, going down in steps of size 0.1

nodes from the graph and using paths between them as positive examples, i.e., we had three path examples in each of the ten settings. Very high compression rates can be achieved: starting from a theory of 144 clauses, compressing to less than 20 clauses has only a minor effect on the likelihood. The radical drops in the end occur when examples cannot be proven anymore.

To test and illustrate the effect of negative examples (undesired paths), we created 10 new settings with both positive and negative examples. This time the above mentioned 10 sets of random paths were used as negative examples. Each test setting uses the paths of one such set as negative examples together with the original positive examples (paths between Alzheimer genes). Figure 4(b) shows how the total log-likelihood curves have a nice convex shape, quickly reaching high likelihoods, and only dropping with very small theories. A factorization of the log-likelihood to the positive and negative components (Fig. 5(a)) explains this: clauses that affect the negative examples mostly are removed first (resulting in an improvement of the likelihood). Only when no other alternatives exist, clauses important for the positive examples are removed (resulting in a decrease in the likelihood). This suggests

that negative examples can be used effectively to guide the compression process, an issue to be studied shortly in a cross-validation setting.

How appropriately are edges of known positive and negative effects handled? Next we inserted new nodes and edges into the real biological graph, with clearly intended positive or negative effects. We forced the algorithm to remove all generated revision points and obtained a ranking of edges. To get edges with a negative effect, we added a new “negative” node `neg` and edges `edge(gene_620, neg)`, `edge(gene_582, neg)`, `edge(gene_983, neg)`. Negative examples were then specified as paths between the new negative node and each of the three genes. For a positive effect, we added a new “positive” node and three edges in the same way, to get short artificial connections between the three genes. As positive examples, we again used `path(g1, g2)` for the three pairs of genes. All artificial edges were given the same probability p .

(Different values of p lead to different sets of revision points. This depends on how many proofs are needed to reach the probability interval δ . To obtain comparable results, we computed in a first step the revision points using $p = 0.5$ and interval width 0.2. All clauses not appearing as a revision point were excluded from the experiments. On the resulting theory, we then re-ran the compression algorithm for $p = 0.1, 0.2, \dots, 0.9$ using $\delta = 0.0$, i.e., using exact inference.)

Figure 5(b) shows the log-likelihood of the examples as a function of the number of clauses remaining in the theory during compression; the types of removed edges are coded by linestyle. In all cases, the artificial negative edges were removed early, as expected. For probabilities $p \geq 0.5$, the artificial positive edges were always the last ones to be removed. This also corresponds to the expectations, as these edges can contribute quite a lot to the positive examples. Results with different values of p indicate how sensitive the method is to recognize the artificial edges. Their influence drops together with p , but negative edges are always removed before positive ones.

How does compression affect unknown test examples? We next study generalization beyond the (training) examples used in compression. We illustrate this in an alternative setting for ProbLog theory revision, where the goal is to minimize changes to the theory. More specifically, we do not modify those parts of the initial theory that are not relevant to the training examples. This is motivated by the desire to apply the revised theory also on unseen test examples that may be located outside the subgraph relevant to training examples, otherwise they would all be simply removed. Technically this is easily implemented: clauses that are not used in any of the training example BDDs are kept in the compressed theory.

Since the difficulty of compression varies greatly between different graphs and different examples, we used a large number of controlled random graphs and constructed positive and negative examples as follows. First, three nodes were randomly selected: a target node (“disease”) from the middle of the graph, a center for a positive cluster (“disease genes”) closer to the perimeter, and a center for a negative cluster (“irrelevant genes”) at random. Then a set of positive nodes was picked around the positive center, and for each of them a positive example was specified as a path to the target node. In the same way, a cluster of negative nodes and a set of negative examples were constructed. A cluster of nodes is likely to share subpaths to the target node, resulting in a concept that can potentially be learnt. By varying the tightness of the clusters, we can tune the difficulty of the task. In the following experiments, negative nodes were clustered but the tightness of the positive cluster was varied. We generated 1000 random graphs. Each of our random graphs had 20 nodes of average degree 3. There were 3 positive and 3 negative examples, and one of each was

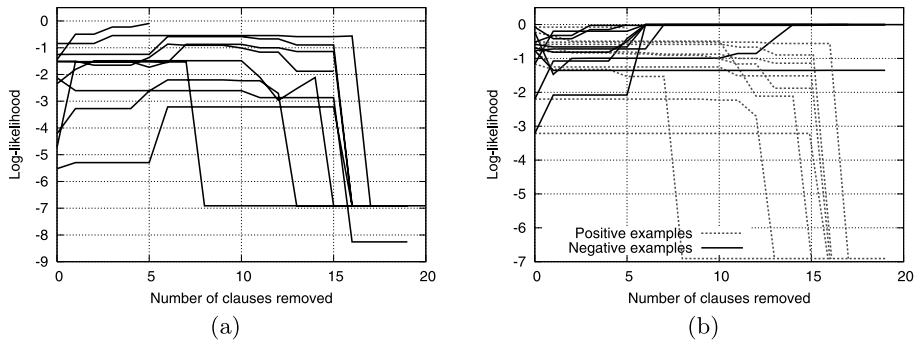


Fig. 6 Evolution of log-likelihoods in test sets for 10 random runs: total log-likelihood (a) and log-likelihoods of positive and negative test examples separately (b)

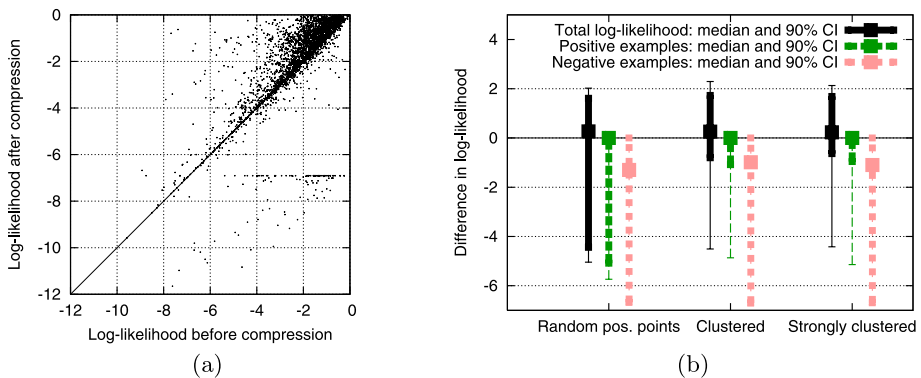


Fig. 7 Log-likelihoods of the test sets before and after compression (a). Distributions of differences of log-likelihoods in test examples before and after compression, for three different densities of positive nodes (b) (thick line: 90% confidence interval; thin line: 95% confidence interval; for negative test examples, differences in $\log(1 - \text{likelihood})$ are reported to make results comparable with positive examples)

always in the hold-out dataset, leaving 2 + 2 examples in the training set. This obviously is a challenging setting.

We compressed each of the ProbLog theories based on the training set only; Fig. 6 shows 10 random traces of the log-likelihoods in the hold-out test examples. The figure also gives a break-down to separate log-likelihoods of positive and negative examples. The behavior is much more mixed than in the training set (cf. Figs. 4(b) and 5(a)), but there is a clear tendency to first improve likelihood (using negative examples) before it drops for very small theories (because positive examples become unprovable).

To study the relationship between likelihood in the training and test sets more systematically, we took for each random graph the compressed theory that gave the maximum likelihood in the training set. A summary over the 1000 runs, using 3-fold cross-validation for each, is given in Fig. 7(a). The first observation is that compression is on average useful for the test set. The improvement over the original likelihood is on average about 30% (0.27 in log-likelihood scale), but the variance is large. The large variance is primarily caused by cases where the positive test example was completely disconnected from the target node,

resulting in the use of probability $\epsilon = 0.001$ for the example, and probabilities ≤ 0.001 for the pair of examples. These cases are visible as a cloud of points below $\log\text{-likelihood} \log(0.001) = -6.9$. The joint likelihood of test examples was improved in 68% of the cases, and decreased only in about 17% of the cases (and stayed the same in 15%). Statistical tests of either the improvement of the log-likelihood (paired t-test) or the proportion of cases of increased likelihood (binomial test) show that the improvement is statistically extremely significant (note that there are $N = 3000$ data points).

Another illustration of the powerful behavior of theory compression is given in Fig. 7(b). It shows the effect of compression, i.e., the change in test set likelihood that resulted from maximum-likelihood theory compression for three different densities of positive examples; it also shows separately the result for positive and negative test examples. Negative test examples experience on average a much clearer change in likelihood than the positive ones, demonstrating that ProbLog compression does indeed learn. Further, in all these settings, the median change in positive test examples is zero, i.e., more than one half of the cases experienced no drop in the probability of positive examples, and for clustered positive examples 90% of the cases are relatively close to zero. Results for the negative examples are markedly different, with much larger proportions of large differences in log-likelihood.

To summarize, all experiments show that our method yields good compression results. The likelihood is improving, known positive and negative examples are respected, and the result generalizes nicely to unknown examples. Does this, however, come at the expense of very high running times? This question will be investigated in the following subsection.

7.4 Complexity of ProbLog theory compression

There are several factors influencing the running time of our compression approach. The aim of the following set of experiments was to figure out the crucial factors on running times.

How do the methods scale up to large graphs? To study the scalability of the methods, we randomly subsampled edges from the larger biological graph, to obtain subgraphs $G_1 \subset G_2 \subset \dots$ with 200, 400, ... edges. Each G_i contains the three genes and consists of one connected component. Average degree of nodes ranges in G_i s approximately from 2 to 3. The ProbLog compression algorithm was then run on the data sets, with k , the maximum size of the compressed theory, set to 15.

Running times are given in Fig. 8(a). Graphs of 200 to 1400 edges were compressed in 3 to 40 minutes, which indicates that the methods can be useful in practical, large link mining tasks.

The results also nicely illustrate how difficult it is to predict the problem complexity: up to 600 edges the running times increase, but then drop when increasing the graph size to 1000 edges. Larger graphs can be computationally easier to handle, if they have additional edges that result in good proofs and remove the need of deep search for obtaining tight bounds.

What is the effect of the probability approximation interval δ ? Smaller values of δ obviously are more demanding. To study the relationship, we ran the method on the smaller biological graph with the three positive examples using varying δ values. Figure 8(b) shows the total running time as well as its decomposition to two phases, finding revision points and removing them. The interval width δ has a major effect on running time; in particular, the complexity of the revision step is greatly affected by δ .

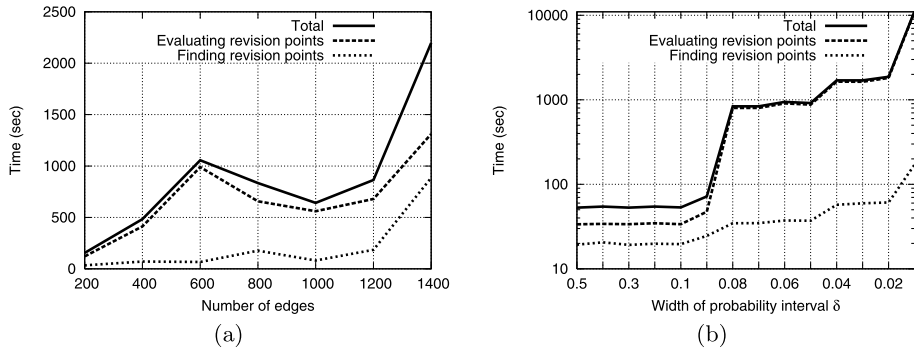


Fig. 8 Running time as a function of graph size (number of edges) **(a)** and as a function of interval width δ **(b)**. Note that in **(b)** the y axis is in log-scale and the x axis is discretized in 0.1 steps for $[0.5, 0.1]$ and in 0.01 steps in $[0.1, 0.01]$

What are the crucial technical factors for running times? We address the two phases, finding revision points and removing them, separately.

Given an interval δ , resources needed for running the approximation algorithm to obtain the revision points are hard to predict, as those are extremely dependent on the sets of proofs and especially stopped derivations encountered until the stopping criterion is reached. This is not only influenced by the structure of the theory and δ , but also by the presentation of examples. (In one case, reversing the order of nodes in some path atoms decreased running time by several orders of magnitude.) Based on our experiments, a relatively good indicator of the complexity is the total size of the BDDs used.

The complexity of the revision phase is more directly related to the number of revision points. Assuming a constant time for using a given BDD to compute a probability, the time complexity is quadratic in the number of revision points. In practice, the cost of using a BDD depends of course on its size, but compared to the building time, calling times are relatively small. One obvious way of improving the efficiency of the revision phase is to greedily remove more than one clause at a time.

Although in Fig. 8(b) the revision time almost always dominates the total time, we have found in our experiments that there is a lot of variance here, too, and in practice either phase can strongly dominate the total time.

Given a fixed value for δ , there is little we can do to affect the size of the BDDs or the number of revision points. However, these observations may be useful for designing alternative parameterizations for the revision algorithm that have more predictable running times (with the cost of less predictable probability intervals).

8 Related work and conclusions

Using the probabilistic variant of Prolog, called ProbLog, we have introduced a novel framework for theory compression in large probabilistic databases. ProbLog's only assumption is that the probabilities of clauses are independent of one another. The semantics of ProbLog are related to those of some other well-known probabilistic extensions of Prolog such as pD (Fuhr 2000), SLPs (Muggleton 1996), PRISM (Sato and Kameya 2001) and PHA (Poole 1993), but also subtly different. For instance, pD assigns a Boolean variable to each *instance* of a clause and its inference procedure only works for very small problems (cf. above). In

SLPs and PRISM, each (repeated) use of a probabilistic clause (or switch) explicitly contributes to the probability of a proof, whereas in ProbLog there is a single random variable that covers all calls to that fact or clause. The number of random variables is thus fixed in ProbLog, but proof-dependent in the other frameworks. Nevertheless, it would be interesting to explore the possibility of transferring algorithms between those systems.

ProbLog has been used to define a new type of theory compression problem, aiming at finding a small subset of a given program that maximizes the likelihood w.r.t. a set of positive and negative examples. This problem is related to the traditional theory revision problem studied in inductive logic programming (Wrobel 1996) in that it allows particular operations on a theory (in this case only deletions) on the basis of positive and negative examples but differs in that it aims at finding a small theory and that is also grounded in a sound probabilistic framework. This framework bears some relationships to the PTR approach by (Koppel et al. 1994) in that possible revisions in PTR are annotated with weights or probabilities. Still, PTR interprets the theory in a purely logical fashion to classify examples. It only uses the weights as a kind of bias during the revision process in which it also updates them using examples in a kind of propagation algorithm. When the weights become close to 0, clauses are deleted. ProbLog compression is also somewhat related to Zelle and Mooney's work on Chill (Zelle and Mooney 1994) in that it specializes an overly general theory but differs again in the use of a probabilistic framework.

A solution to the ProbLog theory compression problem has then been developed. Central in the development of the ProbLog inference and compression algorithms is the use of BDDs (Bryant 1986). BDDs are a popular and effective representation of Boolean functions. Although different variants of BDDs are used by Chavira and Darwiche (2007) and Minato et al. (2007) to benefit from local structure during probability calculation in Bayesian networks, ProbLog is—to the best of our knowledge—the first probabilistic logic that uses BDDs.

Finally, we have shown that ProbLog inference and compression is not only theoretically interesting, but is also applicable on various realistic problems in a biological link discovery domain.

Acknowledgements We are grateful to P. Sevon, L. Eronen, P. Hintsanen and K. Kulovesi for the real data. K. Kersting and A. Kimmig have been supported by the EU IST FET project April II. K. Revoredo has been supported by Brazilian Research Council, CNPq. H. Toivonen has been supported by the Humboldt foundation and Tekes.

References

- Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8), 677–691.
- Chavira, M., & Darwiche, A. (2007). Compiling Bayesian networks using variable elimination. In M. Veloso (Ed.), *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 2443–2449). Menlo Park: AAAI Press.
- De Raedt, L., & Kersting, K. (2003). Probabilistic logic learning. *SIGKDD Explorations*, 5(1), 31–48.
- De Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: a probabilistic Prolog and its application in link discovery. In M. Veloso (Ed.), *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 2468–2473). Menlo Park: AAAI Press.
- Flach, P. A. (1994). *Simply logical: intelligent reasoning by example*. New York: Wiley.
- Fuhr, N. (2000). Probabilistic datalog: implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51, 95–110.
- Getoor, L., & Taskar, B. (Eds.). (2007). *Statistical relational learning*. Cambridge: MIT Press.
- Koppel, M., Feldman, R., & Segre, A. M. (1994). Bias-driven revision of logical domain theories. *Journal of Artificial Intelligence Research*, 1, 159–208.

- Minato, S., Satoh, K., & Sato, T. (2007). Compiling Bayesian networks by symbolic probability calculation based on zero-suppressed BDDs. In M. Veloso (Ed.), *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 2550–2555). Menlo Park: AAAI Press.
- Muggleton, S. H. (1996). Stochastic logic programs. In L. De Raedt (Ed.), *Advances in inductive logic programming*. Amsterdam: IOS Press.
- Perez-Iratxeta, C., Bork, P., & Andrade, M. A. (2002). Association of genes to genetically inherited diseases using data mining. *Nature Genetics*, 31, 316–319.
- Poole, D. (1992). Logic programming, abduction and probability. In *Fifth generation computing systems* (pp. 530–538).
- Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64, 81–129.
- Sato, T., & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of AI Research*, 15, 391–454.
- Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., & Toivonen, H. (2006). Link discovery in graphs derived from biological databases. In U. Leser, F. Naumann, & B. Eckman (Eds.), *Lecture notes in bioinformatics: Vol. 4075. Data integration in the life sciences 2006*. Berlin: Springer.
- Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8, 410–411.
- Wrobel, S. (1996). First Order Theory Refinement. In L. De Raedt (Ed.), *Advances in inductive logic programming*. Amsterdam: IOS Press.
- Zelle, J. M., & Mooney, R. J. (1994). Inducing deterministic Prolog parsers from treebanks: a machine learning approach. In *Proceedings of the 12th national conference on artificial intelligence (AAAI-94)* (pp. 748–753).

Feature Construction Based on Closedness Properties Is Not That Simple

Dominique Gay¹, Nazha Selmaoui¹, and Jean-François Boulicaut²

¹ ERIM EA 3791, University of New Caledonia,
BP R4, F-98851 Nouméa, New Caledonia
{dominique.gay, nazha.selmaoui}@univ-nc.nc

² INSA-Lyon, LIRIS CNRS UMR5205
F-69621 Villeurbanne Cedex, France
jean-francois.boulicaut@insa-lyon.fr

Abstract. Feature construction has been studied extensively, including for 0/1 data samples. Given the recent breakthrough in closedness-related constraint-based mining, we are considering its impact on feature construction for classification tasks. We investigate the use of condensed representations of frequent itemsets (closure equivalence classes) as new features. These itemset types have been proposed to avoid set counting in difficult association rule mining tasks. However, our guess is that their intrinsic properties (say the maximality for the closed itemsets and the minimality for the δ -free itemsets) might influence feature quality. Understanding this remains fairly open and we discuss these issues thanks to itemset properties on the one hand and an experimental validation on various data sets on the other hand.

1 Introduction

Feature construction is one of the major research topics for supporting classification tasks. Based on a set of original features, the idea is to compute new features that may better describe labeled samples such that the predictive accuracy of classifiers can be improved. When considering the case of 0/1 data (i.e., in most of the cases, collections of attribute-value pairs that are true or not within a sample), several authors have proposed to look at feature construction based on patterns that satisfy closedness-related constraints [1,2,3,4,5,6]. Using patterns that hold in 0/1 data as features (e.g., itemsets or association rules) is not new. Indeed, pioneering work on classification based on association rules [7] or emerging pattern discovery [8,9] have given rise to many proposals. Descriptive pattern discovery from unlabeled 0/1 data has been studied extensively during the last decade: many algorithms have been designed to compute every set pattern that satisfies a given constraint (e.g., a conjunction of constraints whose one conjunct is a minimal frequency constraint). One breakthrough into the computational complexity of such mining tasks has been obtained thanks to condensed

representations for frequent itemsets, i.e., rather small collections of patterns from which one can infer the frequency of many sets instead of counting for it (see [10] for a survey). In this paper, we consider closure equivalence classes, i.e., frequent closed sets and their generators [11]. Furthermore, when considering the δ -free itemsets with $\delta > 0$ [12,13], we can consider a “near equivalence” perspective and thus, roughly speaking, the concept of almost-closed itemsets. We want to contribute to difficult classification tasks by using a method based on: (1) the efficient extraction of set patterns that satisfy given constraints, (2) the encoding of the original data into a new data set by using extracted patterns as new features. Clearly, one of the technical difficulties is to discuss the impact of the intrinsic properties of these patterns (i.e., closedness-related properties) on a classification process.

Our work is related to pattern-based classification. Since [7], various authors have considered the use of association rules. These proposals are based on a pruned set of extracted rules built w.r.t. support and confidence ranking. Differences between these methods mainly come from the way they use the selected set of rules when an unseen example x is coming. For example, CBA [7] ranks the rules and it uses the best one to label x . Other algorithms choose the class that maximizes a defined score (CMAR [14] uses *combined effect* of subsets of rules when CPAR [15] uses *average expected accuracy* of the best k rules). Also, starting from ideas for class characterization [16], [17] is an in-depth formalization of all these approaches. Another related research stream concerns emerging patterns [18]. These patterns are frequent in samples of a given class and infrequent for samples from the other classes. Several algorithms have exploited this for feature construction. Some of them select essential ones (CAEP classifier [8]) or the most expressive ones (JEPs classifier [9]). Then, an incoming example is labeled with the class c which maximizes scores based on these sets. Moreover, a few researchers have considered condensed representations of frequent sets for feature construction. Garriga et al. [3] have proposed to characterize a target class with a collection of relevant closed itemsets. Li et al. [1] invoke MDL principle and suggest that free itemsets might be better than closed ones. However, classification experimental results to support such a claim are still lacking. It turns out that the rules studied in [17] are based on 0-free sets such that a minimal body property holds. The relevancy of such a minimality property is also discussed in terms of “near equivalence” in [19]. In [2], we have considered preliminary results on feature construction based on δ -freeness [12,13]. Feature construction approaches based on closedness properties differ in two main aspects: (i) mining can be performed on the whole database or per class, and (ii) we can mine with or without the class labels. The pros and cons of these alternatives are discussed in this paper.

In Section 2, we provide more details on state-of-the-art approaches before introducing our feature construction method. Section 3 reports on our experimental results for UCI data sets [20] and a real-world medical database. Section 4 concludes.

2 Feature Construction Using Closure Equivalence Classes

A *binary database* r is defined as a binary relation $(\mathcal{T}, \mathcal{I}, R)$ where $\mathcal{T}$ is a set of objects (or transactions), $\mathcal{I}$ is a set of attributes (or items) and $R \subseteq \mathcal{T} \times \mathcal{I}$. The *frequency* of an itemset $I \subseteq \mathcal{I}$ in r is $\text{freq}(I, r) = |\text{Objects}(I, r)|$ where $\text{Objects}(I, r) = \{t \in \mathcal{T} \mid \forall i \in I, (t, i) \in R\}$. Let γ be an integer, an itemset I is said to be γ -*frequent* if $\text{freq}(I, r) \geq \gamma$.

Considering that “*what is frequent may be interesting*” is intuitive, Cheng et al. [4] brought some evidence to support such a claim and they have linked frequency with other interestingness measures such as Information Gain and Fisher score. Since the number of frequent itemsets can be huge in dense databases, it is now common to use condensed representations (e.g., free itemsets, closed ones, non derivable itemsets [10]) to save space and time during the frequent itemset mining task and to avoid some redundancy.

Definition 1 (Closed itemset). *An itemset I is a closed itemset in r iff there is no superset of I with the same frequency than I in r , i.e., $\nexists I' \supset I$ s.t. $\text{freq}(I', r) = \text{freq}(I, r)$. Another definition exploits the closure operation $cl : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{I})$. Assume that *Items* is the dual operator for *Objects*: given $T \subseteq \mathcal{T}$, $\text{Items}(T, r) = \{i \in \mathcal{I} \mid \forall t \in T, (t, i) \in R\}$, and assume $cl(I, r) \equiv \text{Items}(\text{Objects}(I, r), r)$: the itemset I is a closed itemset in r iff $I = cl(I, r)$.*

Since [11], it is common to formalize the fact that many itemsets have the same closure by means of *closure equivalence relation*.

Definition 2 (Closure equivalence). *Two itemsets I and J are said to be equivalent in r (denoted $I \sim_{cl} J$) iff $cl(I, r) = cl(J, r)$. Thus, a closure equivalence class (CEC) is made of itemsets that have the same closure, i.e., they are all supported by the same set of objects ($\text{Objects}(I, r) = \text{Objects}(J, r)$).*

Each CEC contains exactly one maximal itemset (w.r.t. set inclusion) which is a closed itemset. It may contain several minimal itemsets which are 0-free itemsets according to the terminology in [12] (also called key patterns in [11]).

Example 1. Considering Tab. 1, we have $r = (\mathcal{T}, \mathcal{I}, R)$, $\mathcal{T} = \{t_1, \dots, t_6\}$, and $\mathcal{I} = \{A, B, C, D, c_1, c_2\}$, c_1 and c_2 being the class labels. For a frequency threshold $\gamma = 2$, itemsets AB and AC are γ -frequent. $ABCc_1$ is a γ -frequent closed itemset. Considering the equivalence class $\mathcal{C} = \{AB, AC, ABC, ABc_1, ACc_1, ABCc_1\}$, AB and AC are its minimal elements (i.e., they are 0-free itemsets) and $ABCc_1$ is the maximal element, i.e., one of the closed itemsets in this toy database.

2.1 Freeness or Closedness?

Two different approaches for feature construction based on condensed representations have been considered so far. In, e.g., [1,5], the authors mine free itemsets and closed itemsets (i.e., CECs) once the class attribute has been removed from

Table 1. A toy example of a binary labeled database

r	A	B	C	D	c_1	c_2
t_1	1	1	1	1	1	0
t_2	1	1	1	0	1	0
t_3	0	1	1	0	1	0
t_4	1	0	0	1	1	0
t_5	0	1	1	0	0	1
t_6	0	1	0	1	0	1

the entire database. Other proposals, e.g., [3,4], consider (closed) itemset mining from samples of each class separately.

Looking at the first direction of research, we may consider that closed sets, because of their maximality, are good candidates for characterizing labeled data, but not necessarily suitable to predict classes for unseen samples. Moreover, thanks to their minimality, free itemsets might be better for predictive tasks. Due to closedness properties, every itemset of a given closure equivalence class $\mathcal{C}$ in r covers exactly the same set of objects. Thus, free itemsets and their associated closed are equivalent w.r.t. interestingness measures based on frequencies. As a result, it is unclear whether choosing a free itemset or its closure to characterize a class is important or not. Let us now consider an incoming sample x (test phase) that is exactly described by the itemset Y (i.e., all its properties that are true are in Y). Furthermore, assume that we have $F \subseteq Y \subseteq cl(F, r)$ where F is a free itemset from the closure equivalence class $\mathcal{C}_F$. Using free itemsets to label x will not lead to the same decision than using closed itemsets. Indeed, $x \supseteq F$ and it satisfies rule $F \Rightarrow c$ while $x \not\supseteq cl(Y, r)$ and it does not satisfy rule $cl(F, r) \Rightarrow c$. Following that direction of work, Baralis et al. have proposed classification rules based on free itemsets [17].

On the other hand, for the “per-class” approach, let us consider w.l.o.g a two-class classification problem. In such a context, the equivalence between free itemsets and their associated closed ones is lost. The intuition is that, for a given free itemset Y in r_{c_1} -database restricted to samples of class c_1 - and its closure $X = cl(Y, r_{c_1})$, X is more relevant than Y since $Objects(X, r_{c_1}) = Objects(Y, r_{c_1})$ and $Objects(X, r_{c_2}) \subseteq Objects(Y, r_{c_2})$. The closed itemsets (say $X = cl(X, r_{c_1})$) such that there is no other closed itemset (say $X' = cl(X', r)$) for which $cl(X, r_{c_2}) = cl(X', r_{c_2})$ are chosen as relevant itemsets to characterize c_1 . In some cases, a free itemset Y could be equivalent to its closure $X = cl(Y, r_{c_1})$, i.e., $Objects(X, r_{c_2}) = Objects(Y, r_{c_2})$. Here, for the same reason as above, a free itemset may be chosen instead of its closed counterpart. Note that relevancy of closed itemsets does not avoid conflicting rules, i.e., we can have two closed itemsets X relevant for c_1 and Y relevant for c_2 with $X \subseteq Y$.

Moreover, these approaches need for a post-processing of the extracted patterns. Indeed, we not only look for closedness-related properties but we have also to exploit interesting measures to keep only the ones that are discriminating. To avoid such a post-processing, we propose to use syntactic constraint (i.e., keeping

the class attribute during the mining phase) to mine class-discriminant closure equivalence classes.

2.2 What Is Interesting in Closure Equivalence Classes?

In Fig. 1, we report the different kinds of CECs that can be obtained when considering class attributes during the mining phase.

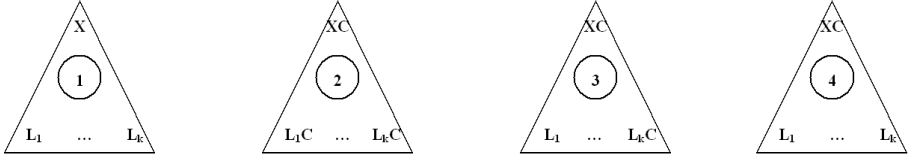


Fig. 1. Different types of CECs

These CECs have nice properties that are useful to our purpose: since association rules with a maximal confidence (no exception, also called hereafter exact rules) stand between a free itemset and its closure, we are interested in CECs whose closure contains a class attribute to characterize classes. Thus, we may neglect Case 1 in Fig. 1.

Definition 3 (Association rule). *Given $r = \{\mathcal{I}, \mathcal{R}\}$, an association rule π on r is an expression $I \Rightarrow J$, where $I \subseteq \mathcal{I}$ and $J \subseteq \mathcal{I} \setminus I$. The frequency of the rule π is $\text{freq}(I \cup J, r)$ and its confidence is $\text{conf}(\pi, r) = \text{freq}(I \cup J, r) / \text{freq}(I, r)$. It provides a ratio about the numbers of exceptions for π in r . When J turns to be a single class attribute, π is called a classification rule.*

From Case 3 (resp. Case 4), we can extract the exact classification rule $\pi_3 : L_1 \Rightarrow C$ (resp. the exact rules $\pi_{4_1} : L_1 \Rightarrow C \cdots \pi_{4_k} : L_k \Rightarrow C$). Note that if we are interested in exact rules only, we also neglect Case 2: $L_1 C$ is a free itemset and it implies there is no exact rule $I \Rightarrow J$ such that $I \cup J \subseteq L_1 C$. Thus, we are interested in CECs whose closed itemset contains a class attribute and whose free itemsets (at least one) do not contain a class attribute. This also leads to a closedness-related condensed representation of Jumping Emerging Patterns [21]. Unfortunately, in pattern-based classification (a fortiori in associative classification), for a given frequency threshold γ , mining exact rules is restrictive since they can be rare and the training database may not be covered by the rule set. In a relaxed setting, we consider association rules that enable exceptions.

Definition 4 (δ -strong rule, δ -free itemset). *Let δ be an integer. A δ -strong rule is an association rule of the form $I \Rightarrow^\delta J$ which is violated in at most δ objects, and where $I \subseteq \mathcal{I}$ and $J \subseteq \mathcal{I} \setminus I$. An itemset $I \subseteq \mathcal{I}$ is a δ -free itemset iff there is no δ -strong rule which holds between its proper subsets. When $\delta = 0$, δ is omitted, and we talk about strong rules, and free itemsets.*

When the right-hand side is a single item i , saying that $I \Rightarrow^\delta i$ is a δ -strong rule in r means that $\text{freq}(I, r) - \text{freq}(I \cup \{i\}) \leq \delta$. When this item is a class attribute, a δ -strong rule is called a δ -strong classification rule [16].

The set of δ -strong rules can be built from δ -free itemsets and their δ -closures.

Definition 5 (δ -closure). *Let δ be an integer. The δ -closure of an itemset I on r is $cl_\delta : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{I})$ s.t. $cl_\delta(I, r) = \{i \in \mathcal{I} \mid \text{freq}(I, r) - \text{freq}(I \cup \{i\}) \leq \delta\}$. Once again, when $\delta = 0$, $cl_0(I, r) = \{i \in \mathcal{I} \mid \text{freq}(I, r) = \text{freq}(I \cup \{i\})\}$ and it corresponds to the closure operator that we already defined. We can also group itemsets by δ -closure equivalence classes: two δ -free itemsets I and J are δ -equivalent ($I \sim_{cl_\delta} J$) if $cl_\delta(I, r) = cl_\delta(J, r)$.*

The intuition is that the δ -closure of a set I is the superset X of I such that every added attribute is almost always true for the objects which satisfy the properties from I : at most δ false values (or exceptions) are enabled. The computation of every frequent δ -free set (i.e., sets which are both frequent and δ -free) can be performed efficiently [13]. Given threshold values for γ (frequency) and δ (freeness), the used `AC_like`¹ implementation outputs each δ -free frequent itemset and its associated δ -closure. Considering Table 1, a frequency threshold $\gamma = 3$ and a number of exceptions $\delta = 1$, itemset C is a 3-frequent 1-free itemset ; items B and c_1 belong to its δ -closure and $\pi : C \Rightarrow^\delta c_1$ is a 1-strong classification rule.

2.3 Information and Equivalence Classes

We get more information from δ -closure equivalence classes than with other approaches. Indeed, when considering contingency tables (See Tab. 2), for all the studied approaches, f_{*1} and f_{*0} are known (class distribution). However, if we consider the proposals from [3,4] based on frequent closed itemsets mined per class, we get directly the value f_{11} (i.e., $\text{freq}(X \cup c, r)$) and the value for f_{01} can be inferred. Closure equivalence classes in [5] only inform us on f_{1*} (i.e., $\text{freq}(X, r)$) and f_{0*} . In our approach, when mining γ -frequent δ -free itemsets whose closure contains a class attribute, $f_{1*} \geq \gamma$ and we have a lower bound $f_{11} \geq \gamma - \delta$ and an upper bound $f_{10} \leq \delta$ for frequencies on X . We can also infer other bounds for f_{01} and f_{00} ².

Table 2. Contingency table for a δ -strong classification rule $X \Rightarrow^\delta c$

$X \Rightarrow c$	c	$\bar{c}$	Σ
X	f_{11}	f_{10}	f_{1*}
$\bar{X}$	f_{01}	f_{00}	f_{0*}
Σ	f_{*1}	f_{*0}	f_{**}

Moreover, γ -frequent δ -free itemsets, bodies of δ -strong classification rules are known to have a minimal body property. Some constraints on γ and δ can help

¹ `AC_like` implementation is available at <http://liris.cnrs.fr/jeremy.besson/>

² Note the confidence of a δ -strong classification rule π is $f_{11}/f_{1*} \geq 1 - (\delta/\gamma)$.

to avoid some of the classification conflicts announced at the end of Section 2.1. Indeed, [16] has shown that setting $\delta \in [0; \lfloor \gamma/2 \rfloor[$ ensures that we can not have two classification rules $\pi_1 : I \Rightarrow^\delta c_i$ and $\pi_2 : I \Rightarrow^\delta c_j$ with $i \neq j$ s.t. $I \subseteq J$. This constraint also enforces confidence to be greater than $\frac{1}{2}$. Furthermore, we know that we can produce δ -strong classification rules that exhibit the discriminant power of emerging patterns if $\delta \in [0; \gamma \cdot (1 - \frac{|r_{c_i}|}{|r|})[$, r_{c_i} being the database restricted to objects of the majority class c_i [6]. One may say that the concept of γ -frequent δ -free itemsets ($\delta \neq 0$) can be considered as an interestingness measures (function of γ and δ) for feature selection.

2.4 Towards a New Space of Descriptors

Once γ -frequent (δ)-free itemsets have been mined, we can build a new representation of the original database using these new features. Each selected itemset I will generate a new attribute $NewAtt_I$ in the new database. One may encode $NewAtt_I$ to a binary attribute, i.e., for a given object t , $NewAtt_I$ equals 1 if $I \subseteq Items(t, r)$ else 0. In a relaxed setting and noise-tolerant way, we propose to compute $NewAtt_I$ as follows:

$$NewAtt_I(t) = \frac{|I \cap Items(t, r)|}{|I|}$$

This way, I is a multivalued ordinal attribute. It is obvious that for an object t , $NewAtt_I(t) \in \{0, 1, \dots, \frac{p-1}{p}, 1\}$ where $p = |I|$. Then, the value $NewAtt_I(t)$ is the proportion of items $i \in I$ that describe t . We think that multivalued encoding –followed by an entropy-based supervised discretization step³– should hold more information than binary encoding. Indeed, in the worst case, the split will take place between $\frac{p-1}{p}$ and 1, that is equivalent to binary case; in other better cases, split may take place between $\frac{j-1}{p}$ and $\frac{j}{p}$, $1 \leq j \leq p-1$ and this split leads to a better separation of data.

3 Experimental Validation

The frequency threshold γ and the accepted number of exceptions δ are important parameters for our Feature Construction (FC) proposal. Let us discuss how to set up sensible values for them. Extreme values for γ bring either (for lowest values) a huge amount of features –some of which are obviously irrelevant– or (for highest values) not enough features to correctly cover the training set. Furthermore, in both cases, these solutions are of limited interest in terms of Information Gain (see [4]). Then, δ varies from 0 to $\gamma \cdot (1 - \frac{|r_{c_i}|}{r})$ to capture discriminating power of emerging patterns. Once again, lowest values of δ lead to *strong* emerging patterns but a potentially low coverage proportion of data and features with high values of δ lacks of discriminating power.

³ The best split between 2 values is recursively chosen until no more information is gained.

Intuitively, a high coverage proportion implies a relatively good representation of data. In Fig. 2, we plotted proportion of the database coverage w.r.t. δ for a given frequency threshold. Results for **breast**, **cleve**, **heart** and **hepatic** data (from UCI repository) are reported. We easily observe that coverage proportion grows as δ grows. Then, it reaches a saturation point for δ_0 which is interesting: higher values of $\delta > \delta_0$ are less discriminant and lower values $\delta < \delta_0$ cover less objects. In our following experiments, we report (1) maximal accuracies over all γ and δ values (denoted **Max**), and (2) average accuracies of all γ values with $\delta = \delta_0$ (denoted **Av**).

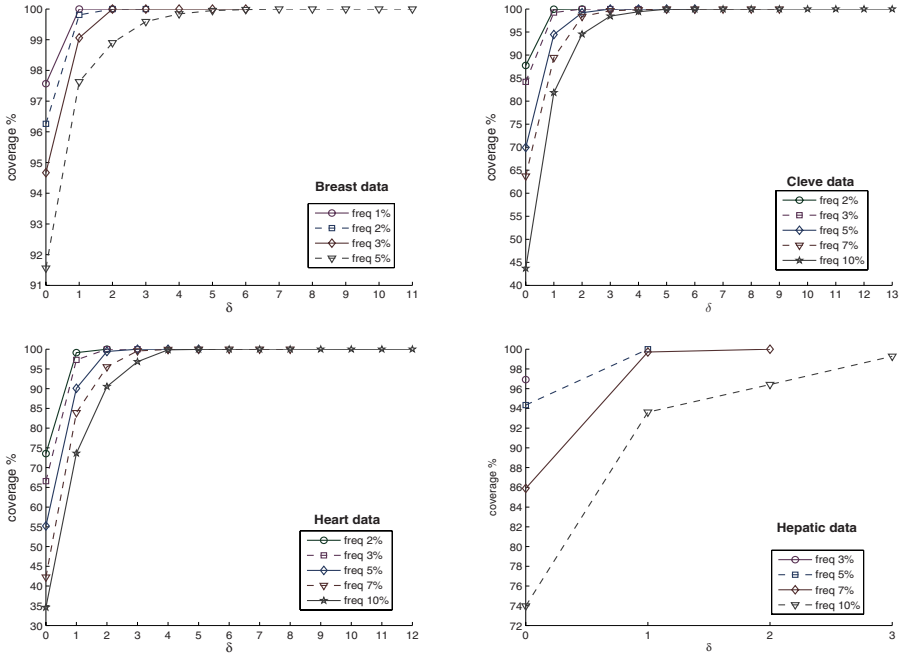


Fig. 2. Evolution of training database coverage proportion w.r.t. γ and δ

To validate our feature construction (FC) process, we used it on several data sets from UCI repository [20] and a real-world data set **meningitis**⁴. We have been using popular classification algorithms such as NB and C4.5 on both the original data and the new representation based on extracted features. As a result, our main objective criterion is the accuracy of the obtained classifiers.

Notice that before performing feature construction, we translated all attributes into binary ones. While the translation of nominal attributes is straightforward, we decided to discretize continuous attributes with the entropy-based method by Fayyad et al. [22]. Discretizations and classifier constructions have been performed with WEKA [23] (10-folds stratified cross validation).

⁴ **meningitis** concerns children hospitalized for acute bacterial or viral meningitis.

Table 3. Accuracy results improvement thanks to FC

databases	NB	FC & NB (Av/Max)	C4.5	FC & C4.5 (Av/Max)
breast	95.99	97.32/97.54	94.56	96.12/96.43
car	85.53	81.95/84.64	92.36	98.49/99.13
cleve	83.5	83.35/ 84.33	76.24	81.39/83.18
crx	77.68	85.91/86.46	86.09	83.95/ 86.33
diabetes	75.91	75.56/ 76.59	72.26	76.03/77.75
heart	84.07	83.62/ 84.81	80	84.56/85.55
hepatic	83.22	84.09/84.67	81.93	85.29/86.83
horse	78.8	81.09/83.74	85.33	83.35/ 85.40
iris	96	94.26/ 96	96	94.26/ 96.67
labor	94.74	93.5/ 95.17	78.95	83.07/87.17
lymph	85.81	83.35/85.46	76.35	81.08/83.46
meningitis	95.74	93.24/93.64	94.83	92.54/ 95.13
sonar	69.71	85.17/86.28	78.85	79.88/83.86
vehicle	45.03	59.72/62.88	71.04	70.70/71.28
wine	96.63	96.42/ 97.83	94.38	95.57/96.29

Table 4. Our FC Feature Construction proposal vs. state-of-the-art approaches

databases	BCEP	LB	FC&NB(Av/Max)	SJEP	CBA	CMAR	CPAR	FC&C4.5(Av/Max)
breast	—	96.86	97.32/97.54	96.96	96.3	96.4	96.0	96.12/96.43
car	—	—	81.95/84.64	—	88.90	—	92.65	98.49/99.13
cleve	82.41	82.19	83.35/84.33	82.41	82.8	82.2	81.5	81.39/ 83.18
crx	—	—	85.91/86.46	87.65	84.7	84.9	85.7	83.95/86.33
diabetes	76.8	76.69	75.56/76.59	76.18	74.5	75.8	75.1	76.03/ 77.75
heart	81.85	82.22	83.62/84.81	82.96	81.9	82.2	82.6	84.56/85.55
hepatic	—	84.5	84.09/ 84.67	83.33	81.8	80.5	79.4	85.29/86.83
horse	—	—	81.09/83.74	84.17	82.1	82.6	84.2	83.35/ 85.40
iris	—	—	94.26/96	—	94.7	94.0	94.7	94.26/ 96.67
labor	—	—	93.5/95.17	82	86.3	89.7	84.7	83.07/87.17
lymph	83.13	84.57	83.35/ 85.46	—	77.8	83.1	82.3	81.08/ 83.46
meningitis	—	—	93.24/93.64	—	91.79	—	91.52	92.54/95.13
sonar	78.4	—	85.17/86.28	85.10	77.5	79.4	79.3	79.88/83.86
vehicle	68.05	68.8	59.72/62.88	71.36	68.7	68.8	69.5	70.70/71.28
wine	—	—	96.42/97.83	95.63	95.0	95.0	95.5	95.57/ 96.29

We report in Tab. 3 the accuracy results obtained on both the original data and its new representation. NB, C4.5 classifiers built on the new representation often perform better (i.e., it lead to higher accuracies) than respective NB and C4.5 classifiers built from the original data. One can see that we have often (12 times among 15) a combination of γ and δ for which NB accuracies are improved by feature construction (column **Max**). And this is experimentally always the case for C4.5. Now considering average accuracies (column **Av**), improvement is still there w.r.t. C4.5 but it appears less obvious when using NB.

Then, we also compared our results with state-of-the-art classification techniques: **FC & NB** is compared with other bayesian approaches, **LB** [24] and **BCEP** [25]. When accessible, accuracies were reported from original papers within Tab. 4. Then, we have compared **FC & C4.5** with other associative classification approaches, namely **CBA** [7], **CMAR** [14], **CPAR** [15], and an EPs-based classifier **SJEP-classifier** [26]. Accuracy results for associative classifiers are taken from [14]. Others results are taken from the published papers. **FC** allows to often achieve better accuracies than the state-of-the-art classifiers, e.g., **FC & C4.5** wins 9 times over 15 against **CPAR**, 8 times over 13 against **CMAR**, 10 times over 15 against **CBA** when considering average accuracies (column **Av**). Considering optimal γ and δ values (column **Max**), it wins 10 times over 15 (see bold faced results).

4 Conclusion

We study the use of closedness-related condensed representations for feature construction. We pointed out that differences about “freeness or closedness” within existing approaches come from the way that condensed representations are mined : with or without class label, per class or in the whole database. We proposed a systematic framework to construct features. Our new features are built from mined (δ)-closure equivalence classes – more precisely from γ -frequent δ -free itemsets whose δ -closures involve a class attribute. Mining these types of itemsets differs from other approaches since (1) mined itemsets hold more information (such as emergence) and (2) there is no need for post-processing the set of features to select interesting features. We also proposed a new numeric encoding that is more suitable than binary encoding. Our **FC** process has been validated by means of an empirical evaluation. Using **C4.5** and **NB** on new representations of various datasets, we demonstrated improvement compared with original data features. We have also shown comparable accuracy results w.r.t. efficient state-of-the-art classification techniques. We have now a better understanding of critical issues w.r.t. feature construction when considering closedness related properties. One perspective of this work is to consider our **FC** process in terms of constraints over sets of patterns and its recent formalization in [27].

Acknowledgments. The authors wish to thank B. Crémilleux for exciting discussions and the data set **meningitis**. They also thank J. Besson for technical support during this study. Finally, this work is partly funded by EU contract IST-FET IQ FP6-516169.

References

1. Li, J., Li, H., Wong, L., Pei, J., Dong, G.: Minimum description length principle: generators are preferable to closed patterns. In: Proceedings AAAI 2006, pp. 409–415. AAAI Press, Menlo Park (2006)
2. Selmaoui, N., Leschi, C., Gay, D., Boulicaut, J.F.: Feature construction and delta-free sets in 0/1 samples. In: Todorovski, L., Lavrač, N., Jantke, K.P. (eds.) DS 2006. LNCS (LNAI), vol. 4265, pp. 363–367. Springer, Heidelberg (2006)

3. Garriga, G.C., Kralj, P., Lavrac, N.: Closed sets for labeled data. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 163–174. Springer, Heidelberg (2006)
4. Cheng, H., Yan, X., Han, J., Hsu, C.W.: Discriminative frequent pattern analysis for effective classification. In: Proceedings IEEE ICDE 2007, pp. 716–725 (2007)
5. Li, J., Liu, G., Wong, L.: Mining statistically important equivalence classes and delta-discriminative emerging patterns. In: Proceedings ACM SIGKDD 2007, pp. 430–439 (2007)
6. Gay, D., Selmaoui, N., Boulicaut, J.F.: Pattern-based decision tree construction. In: Proceedings ICDIM 2007, pp. 291–296. IEEE Computer Society Press, Los Alamitos (2007)
7. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings KDD 1998, pp. 80–86. AAAI Press, Menlo Park (1998)
8. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: Classification by aggregating emerging patterns. In: Arikawa, S., Furukawa, K. (eds.) DS 1999. LNCS (LNAI), vol. 1721, pp. 30–42. Springer, Heidelberg (1999)
9. Li, J., Dong, G., Ramamohanarao, K.: Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems* 3, 131–145 (2001)
10. Calders, T., Rigotti, C., Boulicaut, J.F.: A survey on condensed representations for frequent sets. In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) *Constraint-Based Mining and Inductive Databases*. LNCS (LNAI), vol. 3848, pp. 64–80. Springer, Heidelberg (2006)
11. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. *SIGKDD Explorations* 2, 66–75 (2000)
12. Boulicaut, J.F., Bykowski, A., Rigotti, C.: Approximation of frequency queries by means of free-sets. In: Zighed, A.D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 75–85. Springer, Heidelberg (2000)
13. Boulicaut, J.F., Bykowski, A., Rigotti, C.: Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery* 7, 5–22 (2003)
14. Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: Proceedings IEEE ICDM 2001, pp. 369–376 (2001)
15. Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In: Proceedings SIAM SDM 2003 (2003)
16. Boulicaut, J.F., Crémilleux, B.: Simplest rules characterizing classes generated by delta-free sets. In: Proceedings ES 2002, pp. 33–46. Springer, Heidelberg (2002)
17. Baralis, E., Chiusano, S.: Essential classification rule sets. *ACM Trans. on Database Systems* 29, 635–674 (2004)
18. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings ACM SIGKDD 1999, pp. 43–52 (1999)
19. Bayardo, R.: The hows, whys and whens of constraints in itemset and rule discovery. In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) *Constraint-Based Mining and Inductive Databases*. LNCS (LNAI), vol. 3848, pp. 1–13. Springer, Heidelberg (2006)
20. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
21. Soulet, A., Crémilleux, B., Rioult, F.: Condensed representation of emerging patterns. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 127–132. Springer, Heidelberg (2004)

22. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings IJCAI 1993, pp. 1022–1027 (1993)
23. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
24. Meretakakis, D., Wuthrich, B.: Extending naïve bayes classifiers using long itemsets. In: Proceedings ACM SIGKDD 1999, pp. 165–174 (1999)
25. Fan, H., Ramamohanarao, K.: A bayesian approach to use emerging patterns for classification. In: Proceedings ADC 2003, pp. 39–48. Australian Computer Society, Inc. (2003)
26. Fan, H., Ramamohanarao, K.: Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. *IEEE Trans. on Knowledge and Data Engineering* 18, 721–737 (2006)
27. De Raedt, L., Zimmermann, A.: Constraint-based pattern set mining. In: Proceedings SIAM SDM 2007 (2007)

Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Discovery

Petra Kralj

PETRA.KRALJ@IJS.SI

*Department of Knowledge Technologies
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia*

Nada Lavrač

NADA.LAVRAC@IJS.SI

*Department of Knowledge Technologies
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
and
University of Nova Gorica
Vipavska 13, 5000 Nova Gorica, Slovenia*

Geoffrey I. Webb

GEOFF.WEBB@INFOTECH.MONASH.EDU.AU

*Faculty of Information Technology
Monash University
Building 63, Clayton Campus, Wellington Road, Clayton
VIC 3800, Australia*

Editor: Leslie Pack Kaelbling

Abstract

This paper gives a survey of contrast set mining (CSM), emerging pattern mining (EPM), and subgroup discovery (SD) in a unifying framework named *supervised descriptive rule discovery*. While all these research areas aim at discovering patterns in the form of rules induced from labeled data, they use different terminology and task definitions, claim to have different goals, claim to use different rule learning heuristics, and use different means for selecting subsets of induced patterns. This paper contributes a novel understanding of these subareas of data mining by presenting a unified terminology, by explaining the apparent differences between the learning tasks as variants of a unique supervised descriptive rule discovery task and by exploring the apparent differences between the approaches. It also shows that various rule learning heuristics used in CSM, EPM and SD algorithms all aim at optimizing a trade off between rule coverage and precision. The commonalities (and differences) between the approaches are shown based on a selection of best known variants of CSM, EPM and SD algorithms. The paper also provides a critical survey of existing supervised descriptive rule discovery visualization methods.

Keywords: Descriptive Rules, Rule Learning, Contrast Set Mining, Emerging Patterns, Subgroup Discovery

1. Introduction

Symbolic data analysis techniques aim at discovering comprehensible patterns or models in data. They can be divided into techniques for *predictive induction*, where models, typically induced from class labeled data, are used to predict the class value of previously unseen examples, and *descriptive*

induction, where the aim is to find general humanly understandable patterns, typically induced from unlabeled data. Until recently, these techniques have been investigated by two different research communities: predictive induction mainly by the machine learning community, and descriptive induction mainly by the data mining community.

Data mining tasks where the goal is to find humanly interpretable differences between groups have been addressed by both communities independently. The groups can be interpreted as class labels, so the data mining community, using the association rule learning perspective, adapted association rule learners like Apriori by Agrawal et al. (1996) to perform a task named *contrast set mining* (Bay and Pazzani, 2001) and *emerging pattern mining* (Dong and Li, 1999). On the other hand, the machine learning community, which usually deals with class labeled data, was challenged by, instead of building sets of classification/prediction rules (e.g., Clark and Niblett, 1989; Cohen, 1995), to build individual rules for exploratory data analysis and interpretation, which is the goal of the task named *subgroup discovery* (Wrobel, 1997).

This paper gives a survey of contrast set mining (CSM), emerging pattern mining (EPM), and subgroup discovery (SD) in a unifying framework, named *supervised descriptive rule discovery*. Typical applications of supervised descriptive rule discovery include patient risk group detection in medicine, bioinformatics applications like finding sets of overexpressed genes for specific treatments in microarray data analysis, and identifying distinguishing features of different customer segments in customer relationship management. The main aim of these applications is to understand the underlying phenomena and not to classify new instances. Take another illustrative example, where a manufacturer wants to know in what circumstances his machines may break down; his intention is not to predict breakdowns, but to understand the factors that lead to them and how to avoid them.

The main contributions of this paper are as follows. It provides a survey of supervised descriptive rule discovery approaches addressed in different communities, and proposes a unifying supervised descriptive rule discovery framework, including a critical survey of visualization methods. The paper is organized as follows: Section 2 gives a survey of past research done in the main supervised descriptive rule discovery areas: contrast set mining, emerging pattern mining, subgroup discovery and other related approaches. Section 3 is dedicated to unifying the terminology, definitions and the heuristics. Section 4 addresses visualization as an important open issue in supervised descriptive rule discovery. Section 5 provides a short summary.

2. A Survey of Supervised Descriptive Rule Discovery Approaches

Research on finding interesting rules from class labeled data evolved independently in three distinct areas—contrast set mining, mining of emerging patterns and subgroup discovery—each area using different frameworks and terminology. In this section we provide a survey of these three research areas. We also discuss other related approaches.

2.1 An Illustrative Example

Let us illustrate contrast set mining, emerging pattern mining and subgroup discovery using data from Table 1, a very small, artificial sample dataset¹, adapted from Quinlan (1986). The dataset contains the results of a survey on 14 individuals, concerning the approval or disapproval of an

1. Thanks to Johannes Fürnkranz for providing this dataset.

Education	Marital Status	Sex	Has Children	Approved
primary	single	male	no	no
primary	single	male	yes	no
primary	married	male	no	yes
university	divorced	female	no	yes
university	married	female	yes	yes
secondary	single	male	no	no
university	single	female	no	yes
secondary	divorced	female	no	yes
secondary	single	female	yes	yes
secondary	married	male	yes	yes
primary	married	female	no	yes
secondary	divorced	male	yes	no
university	divorced	female	yes	no
secondary	divorced	male	no	yes

Table 1: A sample database.

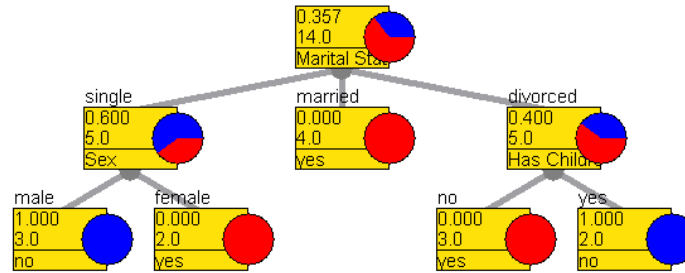


Figure 1: A decision tree, modeling the dataset shown in Table 1.

issue analyzed in the survey. Each individual is characterized by four attributes—Education (with values `primary` school, `secondary` school, or `university`), `MaritalStatus` (`single`, `married`, or `divorced`), `Sex` (`male` or `female`), and `HasChildren` (`yes` or `no`)—that encode rudimentary information about the sociodemographic background. The last column `Approved` is the designated *class* attribute, encoding whether the individual approved or disapproved the issue. Since there is no need for expert knowledge to interpret the results, this dataset is appropriate for illustrating the results of supervised descriptive rule discovery algorithms, whose task is to find interesting patterns describing individuals that are likely to approve or disapprove the issue, based on the four demographic characteristics.

The task of *predictive induction* is to induce, from a given set of *training examples*, a domain model aimed at predictive or classification purposes, such as the *decision tree* shown in Figure 1, or a *rule set* shown in Figure 2, as learned by C4.5 and C4.5rules (Quinlan, 1993), respectively, from the sample data in Table 1.

In contrast to predictive induction algorithms, *descriptive induction* algorithms typically result in rules induced from unlabeled examples. E.g., given the examples listed in Table 1, these algorithms would typically treat the class `Approved` no differently from any other attribute. Note, however, that in the learning framework discussed in this paper, i.e., in the framework of *supervised*

```

Sex = female → Approved = yes
MaritalStatus = single AND Sex = male → Approved = no
MaritalStatus = married → Approved = yes
MaritalStatus = divorced AND HasChildren = yes → Approved = no
MaritalStatus = divorced AND HasChildren = no → Approved = yes

```

Figure 2: A set of predictive rules, modeling the dataset shown in Table 1.

```

MaritalStatus = single AND Sex = male → Approved = no
Sex = male → Approved = no
Sex = female → Approved = yes
MaritalStatus = married → Approved = yes
MaritalStatus = divorced AND HasChildren = yes → Approved = no
MaritalStatus = single → Approved = no

```

Figure 3: Selected descriptive rules, describing individual patterns in the data of Table 1.

descriptive rule discovery, the discovered rules of the form $X \rightarrow Y$ are induced from class labeled data: the class labels are taken into account in learning of patterns of interest, constraining Y at the right hand side of the rule to assign a value to the class attribute.

Figure 3 shows six descriptive rules, found for the sample data using the Magnum Opus (Webb, 1995) software. Note that these rules were found using the default settings except that the critical value for the statistical test was relaxed to 0.25. These descriptive rules differ from the predictive rules in several ways. The first rule is redundant with respect to the second. The first is included as a strong pattern (*all* 3 single males do not approve) whereas the second is weaker but more general pattern (4 out of 7 males do not approve, which is not highly predictive, but accounts for 4 out of all 5 respondents who do not approve). Most predictive systems will include only one of these rules, but either may be of interest to someone trying to understand the data, depending upon the specific application. This particular approach to descriptive pattern discovery does not attempt to second guess which of the more specific or more general patterns will be the more useful.

Another difference between the predictive and the descriptive rule sets is that the descriptive rule set does not include the pattern that divorcees without children approve. This is because, while the pattern is highly predictive in the sample data, there are insufficient examples to pass the statistical test which assesses the probability that, given the frequency of respondents approving, the apparent correlation occurs by chance. The predictive approach often includes such rules for the sake of completeness, while some descriptive approaches make no attempt at such completeness, assessing each pattern on its individual merits.

Exactly which rules will be induced by a supervised descriptive rule discovery algorithm depends on the task definition, the selected algorithm, as well as the user-defined constraints concerning minimal rule support, precision, etc. In the following section, the example set of Table 1 is used to illustrate the outputs of emerging pattern and subgroup discovery algorithms (see Figures 4 and 5, respectively), while a sample output for contrast set mining is shown in Figure 3 above.

2.2 Contrast Set Mining

The problem of mining contrast sets was first defined by Bay and Pazzani (2001) as finding contrast sets as “conjunctions of attributes and values that differ meaningfully in their distributions across groups.” The example rules in Figure 3 illustrate this approach, including all conjunctions of attributes and values that pass a statistical test for productivity (explained below) with respect to attribute *Approved* that defines the ‘groups.’

2.2.1 CONTRAST SET MINING ALGORITHMS

The STUCCO algorithm (Search and Testing for Understandable Consistent Contrasts) by Bay and Pazzani (2001) is based on the Max-Miner rule discovery algorithm (Bayardo, 1998). STUCCO discovers a set of contrast sets along with their supports² on groups. STUCCO employs a number of pruning mechanisms. A potential contrast set X is discarded if it fails a statistical test for independence with respect to the group variable Y . It is also subjected to what Webb (2007) calls a test for *productivity*. Rule $X \rightarrow Y$ is productive iff

$$\forall Z \subset X : \text{confidence}(Z \rightarrow Y) < \text{confidence}(X \rightarrow Y) \quad (1)$$

where $\text{confidence}(X \rightarrow Y)$ is a maximum likelihood estimate of conditional probability $P(Y|X)$, estimated by the ratio $\frac{\text{count}(X,Y)}{\text{count}(X)}$, where $\text{count}(X, Y)$ represents the number of examples for which both X and Y are true, and $\text{count}(X)$ represents the number of examples for which X is true. Therefore a more specific contrast set must have higher confidence than any of its generalizations. Further tests for minimum counts and effect sizes may also be imposed.

STUCCO introduced a novel variant of the Bonferroni correction for multiple tests which applies ever more stringent critical values to the statistical tests employed as the number of conditions in a contrast set is increased. In comparison, the other techniques discussed below do not, by default, employ any form of correction for multiple comparisons, as result of which they have high risk of making *false discoveries* (Webb, 2007).

It was shown by Webb et al. (2003) that contrast set mining is a special case of the more general rule learning task. A contrast set can be interpreted as the antecedent of rule $X \rightarrow Y$, and group G_i for which it is characteristic—in contrast with group G_j —as the rule consequent, leading to rules of the form $\text{ContrastSet} \rightarrow G_i$. A standard descriptive rule discovery algorithm, such as an association-rule discovery system (Agrawal et al., 1996), can be used for the task if the consequent is restricted to a variable whose values denote group membership.

In particular, Webb et al. (2003) showed that when STUCCO and the general-purpose descriptive rule learning system Magnum Opus were each run with their default settings, but the consequent restricted to the contrast variable in the case of Magnum Opus, the contrasts found differed mainly as a consequence only of differences in the statistical tests employed to screen the rules.

Hilderman and Peckham (2005) proposed a different approach to contrast set mining called CIGAR (ContrastIng Grouped Association Rules). CIGAR uses different statistical tests to STUCCO or Magnum Opus for both independence and productivity and introduces a test for *minimum support*.

Wong and Tseng (2005) have developed techniques for discovering contrasts that can include negations of terms in the contrast set.

2. The support of a contrast set ContrastSet with respect to a group G_i , $\text{support}(\text{ContrastSet}, G_i)$, is the percentage of examples in G_i for which the contrast set is true.

In general, contrast set mining approaches require discrete data, which is in real world applications frequently not the case. A data discretization method developed specifically for set mining purposes is described by Bay (2000). This approach does not appear to have been further utilized by the contrast set mining community, except for Lin and Keogh (2006), who extended contrast set mining to time series and multimedia data analysis. They introduced a formal notion of a time series contrast set along with a fast algorithm to find time series contrast sets. An approach to quantitative contrast set mining without discretization in the preprocessing phase is proposed by Simeon and Hilderman (2007) with the algorithm Gen_QCSets. In this approach, a slightly modified equal width binning interval method is used.

Common to most contrast set mining approaches is that they generate all candidate contrast sets from discrete (or discretized) data and later use statistical tests to identify the interesting ones. Open questions identified by Webb et al. (2003) are yet unsolved: selection of appropriate heuristics for identifying interesting contrast sets, appropriate measures of quality for sets of contrast sets, and appropriate methods for presenting contrast sets to the end users.

2.2.2 SELECTED APPLICATIONS OF CONTRAST SET MINING

The contrast mining paradigm does not appear to have been pursued in many published applications. Webb et al. (2003) investigated its use with retail sales data. Wong and Tseng (2005) applied contrast set mining for designing customized insurance programs. Siu et al. (2005) have used contrast set mining to identify patterns in synchrotron x-ray data that distinguish tissue samples of different forms of cancerous tumor. Kralj et al. (2007b) have addressed a contrast set mining problem of distinguishing between two groups of brain ischaemia patients by transforming the contrast set mining task to a subgroup discovery task.

2.3 Emerging Pattern Mining

Emerging patterns were defined by Dong and Li (1999) as itemsets whose support increases significantly from one dataset to another. Emerging patterns are said to capture emerging trends in time-stamped databases, or to capture differentiating characteristics between classes of data.

2.3.1 EMERGING PATTERN MINING ALGORITHMS

Efficient algorithms for mining emerging patterns were proposed by Dong and Li (1999), and Fan and Ramamohanarao (2003). When first defined by Dong and Li (1999), the purpose of emerging patterns was “to capture emerging trends in time-stamped data, or useful contrasts between data classes.” Subsequent emerging pattern research has largely focused on the use of the discovered patterns for classification purposes, for example, classification by emerging patterns (Dong et al., 1999; Li et al., 2000) and classification by jumping emerging patterns³ (Li et al., 2001). An advanced Bayesian approach (Fan and Ramamohanara, 2003) and bagging (Fan et al., 2006) were also proposed.

From a semantic point of view, emerging patterns are association rules with an itemset in rule antecedent, and a fixed consequent: $ItemSet \rightarrow D_1$, for given dataset D_1 being compared to another dataset D_2 .

3. Jumping emerging patterns are emerging patterns with support zero in one dataset and greater than zero in the other dataset.

MaritalStatus = single AND Sex = male $\rightarrow$ Approved = no
 MaritalStatus = married $\rightarrow$ Approved = yes
 MaritalStatus = divorced AND HasChildren = yes $\rightarrow$ Approved = no

Figure 4: Jumping emerging patterns in the data of Table 1.

The measure of quality of emerging patterns is the *growth rate* (the ratio of the two supports). It determines, for example, that a pattern with a 10% support in one dataset and 1% in the other is better than a pattern with support 70% in one dataset and 10% in the other (as $\frac{10}{1} > \frac{70}{10}$). From the association rule perspective, $GrowthRate(ItemSet, D_1) = \frac{confidence(ItemSet \rightarrow D_1)}{1 - confidence(ItemSet \rightarrow D_1)}$. Thus it can be seen that growth rate provides an identical ordering to confidence.

Some researchers have argued that finding all the emerging patterns above a minimum growth rate constraint generates too many patterns to be analyzed by a domain expert. Fan and Ramamohanarao (2003) have worked on selecting the interesting emerging patterns, while Soulet et al. (2004) have proposed condensed representations of emerging patterns.

Boulesteix et al. (2003) introduced a CART-based approach to discover emerging patterns in microarray data. The method is based on growing decision trees from which the emerging patterns are extracted. It combines pattern search with a statistical procedure based on Fisher’s exact test to assess the significance of each emerging pattern. Subsequently, sample classification based on the inferred emerging patterns is performed using maximum-likelihood linear discriminant analysis.

Figure 4 shows all jumping emerging patterns found for the data in Table 1 when using a minimum support of 2. These were discovered using the Magnum Opus software, limiting the consequent to the variable *approved*, setting minimum confidence to 1.0 and setting minimum support to 2.

2.3.2 SELECTED APPLICATIONS OF EMERGING PATTERNS

Emerging patterns have been mainly applied to the field of bioinformatics, more specifically to microarray data analysis. Li et al. (2003) present an interpretable classifier based on simple rules that is competitive to the state of the art black-box classifiers on the acute lymphoblastic leukemia (ALL) microarray dataset. Li and Wong (2002) have focused on finding groups of genes by emerging patterns and applied it to the ALL/AML dataset and the colon tumor dataset. Song et al. (2001) used emerging patterns together with unexpected change and the added/perished rule to mine customer behavior.

2.4 Subgroup Discovery

The task of subgroup discovery was defined by Klösgen (1996) and Wrobel (1997) as follows: “Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically ‘most interesting’, e.g., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest”.

2.4.1 SUBGROUP DISCOVERY ALGORITHMS

Subgroup descriptions are conjunctions of features that are characteristic for a selected class of individuals (property of interest). A subgroup description can be seen as the condition part of a rule

```

Sex = female → Approved = yes
MaritalStatus = married → Approved = yes
MaritalStatus = divorced AND HasChildren = no → Approved = yes
Education = university → Approved = yes
MaritalStatus = single AND Sex = male → Approved = no

```

Figure 5: Subgroup descriptions induced by Apriori-SD from the data of Table 1.

SubgroupDescription $\rightarrow$ *Class*. Therefore, subgroup discovery can be seen as a special case of a more general rule learning task.

Subgroup discovery research has evolved in several directions. On the one hand, exhaustive approaches guarantee the optimal solution given the optimization criterion. One system that can use both exhaustive and heuristic discovery algorithms is Explora by Klösgen (1996). Other algorithms for exhaustive subgroup discovery are the SD-Map method by Atzmüller and Puppe (2006) and Apriori-SD by Kavšek and Lavrač (2006). On the other hand, adaptations of classification rule learners to perform subgroup discovery, including algorithm SD by Gamberger and Lavrač (2002) and algorithm CN2-SD by Lavrač et al. (2004b), use heuristic search techniques drawn from classification rule learning coupled with constraints appropriate for descriptive rules.

Relational subgroup discovery approaches have been proposed by Wrobel (1997, 2001) with algorithm Midos, by Klösgen and May (2002) with algorithm SubgroupMiner, which is designed for spatial data mining in relational space databases, and by Železný and Lavrač (2006) with the algorithm RSD (Relational Subgroup Discovery). RSD uses a propositionalization approach to relational subgroup discovery, achieved through appropriately adapting rule learning and first-order feature construction. Other non-relational subgroup discovery algorithms were developed, including an algorithm for exploiting background knowledge in subgroup discovery (Atzmüller et al., 2005a), and an iterative genetic algorithm SDIGA by del Jesus et al. (2007) implementing a fuzzy system for solving subgroup discovery tasks.

Different heuristics have been used for subgroup discovery. By definition, the interestingness of a subgroup depends on its unusualness and size, therefore the rule quality evaluation heuristics needs to combine both factors. Weighted relative accuracy (*WRAcc*, see Equation 4 in Section 3.3) is used by algorithms CN2-SD, Apriori-SD and RSD and, in a different formulation and in different variants, also by MIDOS and EXPLORA. Generalization quotient (q_g , see Equation 5 in Section 3.3) is used by the SD algorithm. SubgroupMiner uses the classical binominal test to verify if the target share is significantly different in a subgroup.

Different approaches have been used for eliminating redundant subgroups. Algorithms CN2-SD, Apriori-SD, SD and RSD use weighted covering (Lavrač et al., 2004b) to achieve rule diversity. Algorithms Explora and SubgroupMiner use an approach called subgroup suppression (Klösgen, 1996). A sample set of subgroup describing rules, induced by Apriori-SD with parameters *support* set to 15% (requiring at least 2 covered training examples per rule) and *confidence* set to 65%, is shown in Figure 5.

2.4.2 SELECTED APPLICATIONS OF SUBGROUP DISCOVERY

Subgroup discovery was used in numerous real-life applications. The applications in medical domains include the analysis of coronary heart disease (Gamberger and Lavrač, 2002) and brain ischaemia data analysis (Kralj et al., 2007b,a; Lavrač et al., 2007), as well as profiling examiners for sonographic examinations (Atzmüller et al., 2005b). Spatial subgroup mining applications include mining of census data (Klösigen et al., 2003) and mining of vegetation data (May and Ragia, 2002). There are also applications in other areas like marketing (del Jesus et al., 2007; Lavrač et al., 2004a) and analysis of manufacturing shop floor data (Jenkole et al., 2007).

2.5 Related Approaches

Research in some closely related areas of rule learning, performed independently from the above described approaches, is outlined below.

2.5.1 CHANGE MINING

The paper by Liu et al. (2001) on *fundamental rule changes* proposes a technique to identify the set of fundamental changes in two given datasets collected from two time periods. The proposed approach first generates rules and in the second phase it identifies changes (rules) that can not be explained by the presence of other changes (rules). This is achieved by applying statistical χ^2 test for homogeneity of support and confidence. This differs from contrast set discovery through its consideration of rules for each group, rather than itemsets. A change in the frequency of just one itemset between groups may affect many association rules, potentially all rules that have the itemset as either an antecedent or consequent.

Liu et al. (2000) and Wang et al. (2003) present techniques that identify differences in the decision trees and classification rules, respectively, found on two different data sets.

2.5.2 MINING CLOSED SETS FROM LABELED DATA

Closed sets have been proven successful in the context of compacted data representation for association rule learning. However, their use is mainly descriptive, dealing only with unlabeled data. It was recently shown that when considering labeled data, closed sets can be adapted for classification and discrimination purposes by conveniently contrasting covering properties on positive and negative examples (Garriga et al., 2006). The approach was successfully applied in potato microarray data analysis to a real-life problem of distinguishing between virus sensitive and resistant transgenic potato lines (Kralj et al., 2006).

2.5.3 EXCEPTION RULE MINING

Exception rule mining considers a problem of finding a set of rule pairs, each of which consists of an exception rule (which describes a regularity for fewer objects) associated with a strong rule (description of a regularity for numerous objects with few counterexamples). An example of such a rule pair is “using a seat belt is safe” (strong rule) and “using a seat belt is risky for a child” (exception rule). While the goal of exception rule mining is also to find descriptive rules from labeled data, in contrast with other rule discovery approaches described in this paper, the goal of exception rule mining is to find “weak” rules—surprising rules that are an exception to the general belief of background knowledge.

Suzuki (2006) and Daly and Taniar (2005), summarizing the research in exception rule mining, reveal that the key concerns addressed by this body of research include interestingness measures, reliability evaluation, practical application, parameter reduction and knowledge representation, as well as providing fast algorithms for solving the problem.

2.5.4 IMPACT RULES, BUMP HUNTING, QUANTITATIVE ASSOCIATION RULES

Supervised descriptive rule discovery seeks to discover sets of conditions that are related to deviations in the class distribution, where the class is a qualitative variable. A related body of research seeks to discover sets of conditions that are related to deviations in a target quantitative variable. Such techniques include Bump Hunting (Friedman and Fisher, 1999), Quantitative Association Rules (Aumann and Lindell, 1999) and Impact Rules (Webb, 2001).

3. A Unifying Framework for Supervised Descriptive Rule Induction

This section presents a unifying framework for contrast set mining, emerging pattern mining and subgroup discovery, as the main representatives of supervised descriptive rule discovery algorithms. This is achieved by unifying the terminology, the task definitions and the rule learning heuristics.

3.1 Unifying the Terminology

Contrast set mining (CSM), emerging pattern mining (EPM) and subgroup discovery (SD) were developed in different communities, each developing their own terminology that needs to be clarified before proceeding. Below we show that terms used in different communities are compatible, according to the following definition of compatibility.

Definition 1: Compatibility of terms. *Terms used in different communities are compatible if they can be translated into equivalent logical expressions and if they bare the same meaning, i.e., if terms from one community can replace terms used in another community.*

Lemma 1: *Terms used in CSM, EPM and SD are compatible.*

Proof The compatibility of terms is proven through a term dictionary, whose aim is to translate all the terms used in CSM, EPM and SD into the terms used in the rule learning community. The term dictionary is proposed in Table 2. More specifically, this table provides a dictionary of equivalent terms from contrast set mining, emerging pattern mining and subgroup discovery, in a unifying terminology of classification rule learning, and in particular of concept learning (considering class C_i as the concept to be learned from the positive examples of this concept, and the negative examples formed of examples of all other classes). ■

3.2 Unifying the Task Definitions

Having established a unifying view on the terminology, the next step is to provide a unifying view on the different task definitions.

CSM A contrast set mining task is defined as follows (Bay and Pazzani, 2001). Let $A_1, A_2, \dots, A_k$ be a set of k variables called attributes. Each A_i can take values from the set $\{v_{i1}, v_{i2}, \dots,$

Contrast Set Mining	Emerging Pattern Mining	Subgroup Discovery	Rule Learning
contrast set	itemset	subgroup description	rule condition
groups $G_1, \dots, G_n$	datasets D_1 and D_2	class/property C	class/concept C_i
attribute-value pair	item	logical (binary) feature	condition
examples in groups $G_1, \dots, G_n$	transactions in datasets D_1 and D_2	examples of C and $\bar{C}$	examples of $C_1 \dots C_n$
examples for which the contrast set is true	transactions containing the itemset	subgroup of instances	covered examples
support of contrast set on G_i	support of EP in dataset D_1	true positive rate	true positive rate
support of contrast set on G_j	support of EP in dataset D_2	false positive rate	false positive rate

Table 2: Table of synonyms from different communities, showing the compatibility of terms.

v_{im} . Given a set of user defined groups $G_1, G_2, \dots, G_n$ of data instances, a contrast set is a conjunction of attribute-value pairs, defining a pattern that best discriminates the instances of different user-defined groups. A special case of contrast set mining considers only two contrasting groups (G_1 and G_2). In such cases, we wish to find characteristics of one group discriminating it from the other and vice versa.

EPM An emerging patterns mining task is defined as follows (Dong and Li, 1999). Let $I = \{i_1, i_2, \dots, i_N\}$ be a set of items (note that an item is equivalent to a binary feature in SD, and an individual attribute-value pair in CSM). A transaction is a subset T of I . A *dataset* is a set D of transactions. A subset X of I is called an *itemset*. Transaction T contains an itemset X in a dataset D , if $X \subseteq T$. For two datasets D_1 and D_2 , emerging pattern mining aims at discovering itemsets whose support increases significantly from one dataset to another.

SD In subgroup discovery, subgroups are described as conjunctions of features, where features are of the form $A_i = v_{ij}$ for nominal attributes, and $A_i > value$ or $A_i \leq value$ for continuous attributes. Given the property of interest C , and the population of examples of C and $\bar{C}$, the subgroup discovery task aims at finding population subgroups that are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest C (Wrobel, 1997).

The definitions of contrast set mining, emerging pattern mining and subgroup discovery appear different: contrast set mining searches for discriminating characteristics of groups called contrast sets, emerging pattern mining aims at discovering itemsets whose support increases significantly from one dataset to another, while subgroup discovery searches for subgroup descriptions. By using the dictionary from Table 2 we can see that the goals of these three mining tasks are very similar, it is primarily the terminology that differs.

Definition 2: Compatibility of task definitions. *Definitions of different learning tasks are compatible if one learning task can be translated into another learning task without substantially changing the learning goal.*

Lemma 2: *Definitions of CSM, EPM and SD tasks are compatible.*

Proof To show the compatibility of task definitions, we propose a unifying table (Table 3) of task definitions, allowing us to see that emerging pattern mining task $EPM(D_1, D_2)$ is equivalent to $CSM(G_i, G_j)$. It is also easy to show that a two-group contrast set mining task $CSM(G_i, G_j)$ can be

Contrast Set Mining	Emerging Pattern Mining	Subgroup Discovery	Rule Learning
Given examples in G_1 vs. G_j from $G_1, \dots, G_i$	Given transactions in D_1 and D_2 from D_1 and D_2	Given in examples C from C and $\bar{C}$	Given examples in C_i from $C_1 \dots C_n$
Find $ContrastSet_{i_k} \rightarrow G_i$ $ContrastSet_{j_l} \rightarrow G_j$	Find $ItemSet_{1_k} \rightarrow D_1$ $ItemSet_{2_l} \rightarrow D_2$	Find $SubgrDescr_k \rightarrow C$	Find $\{RuleCond_{i_k} \rightarrow C_i\}$

Table 3: Table of task definitions from different communities, showing the compatibility of task definitions in terms of output rules.

directly translated into the following two subgroup discovery tasks: $SD(G_i)$ for $C = G_i$ and $\bar{C} = G_j$, and $SD(G_j)$ for $C = G_j$ and $\bar{C} = G_i$.

Having proved that the subgroup discovery task is compatible with a two-group contrast set mining task, it is by induction compatible with a general contrast set mining task, as shown below.

```

CSM( $G_1, \dots, G_n$ )
  for i=2 to n do
    for j=1, j≠i to n-1 do
      SD( $C = G_i$  vs.  $\bar{C} = G_j$ )

```

Note that in Table 3 of task definitions column ‘Rule Learning’ again corresponds to a concept learning task instead of the general classification rule learning task. In the concept learning setting, which is better suited for the comparisons with supervised descriptive rule discovery approaches, a distinguished class C_i is learned from examples of this class, and examples of all other classes $C_1, \dots, C_{i-1}, C_{i+1}, C_N$ are merged to form the set of examples of class $\bar{C}_i$. In this case, induced rule set $\{RuleCond_{i_k} \rightarrow C_i\}$ consists only of rules for distinguished class C_i . On the other hand, in a general classification rule learning setting, from examples of N different classes a set of rules would be learned $\{\dots, RuleCond_{i_k} \rightarrow C_i, RuleCond_{i_{k+1}} \rightarrow C_i, \dots, RuleCond_{j_l} \rightarrow C_j, \dots, Default\}$, consisting of sets of rules of the form $RuleCond_{i_k} \rightarrow C_i$ for each individual class C_i , supplemented by the default rule. ■

While the primary tasks are very closely related, each of the three communities has concentrated on different sets of issues around this task. The contrast set discovery community has paid greatest attention to the statistical issues of multiple comparisons that, if not addressed, can result in high risks of false discoveries. The emerging patterns community has investigated how supervised descriptive rules can be used for classification. The contrast set and emerging pattern communities have primarily addressed only categorical data whereas the subgroup discovery community has also considered numeric and relational data. The subgroup discovery community has also explored techniques for discovering small numbers of supervised descriptive rules with high coverage of the data.

3.3 Unifying the Rule Learning Heuristics

The aim of this section is to provide a unifying view on rule learning heuristics used in different communities. To this end we first investigate the rule quality measures.

actual	predicted		
	# of positives	# of negatives	
# of positives	$p = TP(X, Y) $	$\bar{p} = FN(X, Y) $	P
# of negatives	$n = FP(X, Y) $	$\bar{n} = TN(X, Y) $	N
	$p + n$	$\bar{p} + \bar{n}$	$P + N$

Table 4: Confusion matrix: $TP(X, Y)$ stands for true positives, $FP(X, Y)$ for false positives, $FN(X, Y)$ for false negatives and $TN(X, Y)$ for true negatives, as predicted by rule $X \rightarrow Y$.

Most rule quality measures are derived by analyzing the covering properties of the rule and the class in the rule consequent considered as positive. This relationship can be depicted by a confusion matrix (Table 4, see e.g., Kohavi and Provost, 1998), which considers that rule $R = X \rightarrow Y$ is represented as (X, Y) , and defines p as the number of true positives (positive examples correctly classified as positive by rule (X, Y)), n as the number of false positives, etc., from which other covering characteristics of a rule can be derived: true positive rate $TPR(X, Y) = \frac{p}{P}$ and false positive rate $FPR(X, Y) = \frac{n}{N}$.

CSM Contrast set mining aims at discovering contrast sets that best discriminate the instances of different user-defined groups. The support of contrast set X with respect to group G_i , $support(X, G_i)$, is the percentage of examples in G_i for which the contrast set is true. Note that *support of a contrast set with respect to group G* is the same as *true positive rate* in the classification rule and subgroup discovery terminology, i.e., $support(X, G_i) = \frac{count(X, G_i)}{|G_i|} = TPR(X, G_i)$. A derived goal of contrast set mining, proposed by Bay and Pazzani (2001), is to find contrast sets whose support differs meaningfully across groups, for δ being a user-defined parameter.

$$SuppDiff(X, G_i, G_j) = |support(X, G_i) - support(X, G_j)| \geq \delta. \quad (2)$$

EPM Emerging pattern mining aims at discovering itemsets whose support increases significantly from one dataset to another Dong and Li (1999), where *support* of itemset X in dataset D is computed as $support(X, D) = \frac{count(X, D)}{|D|}$, for $count(X, D)$ being the number of transactions in D containing X . Suppose we are given an ordered pair of datasets D_1 and D_2 . The *GrowthRate* of an itemset X from D_1 to D_2 , denoted as $GrowthRate(X, D_1, D_2)$, is defined as follows:

$$GrowthRate(X, D_1, D_2) = \frac{support(X, D_1)}{support(X, D_2)} \quad (3)$$

Definitions of special cases of $GrowthRate(X, D_1, D_2)$ are as follows, if $support(X, D_1) = 0$ then $GrowthRate(X, D_1, D_2) = 0$, if $support(X, D_2) = 0$ then $GrowthRate(X, D_1, D_2) = \infty$.

SD Subgroup discovery aims at finding population subgroups that are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest (Wrobel, 1997). There were several heuristics developed and used in the subgroup discovery community. Since they follow from the task definition, they try to maximize subgroup size and the distribution difference at the same time. Examples of such heuristics are

Contrast Set Mining	Emerging Pattern Mining	Subgroup Discovery	Rule Learning
$SuppDiff(X, G_i, G_j)$		$WRAcc(X, C)$	Piatetski-Shapiro heuristic leverage
	$GrowthRate(X, D_1, D_2)$	$q_g(X, C)$	odds ratio for $g = 0$ accuracy/precision, for $g = p$

Table 5: Table of relationships between the pairs of heuristics, and their equivalents in classification rule learning.

the *weighted relative accuracy* (Equation 4, see Lavrač et al., 2004b) and the *generalization quotient* (Equation 5, see Gamberger and Lavrač, 2002), for g being a user-defined parameter.

$$WRAcc(X, C) = \frac{p + n}{P + N} \cdot \left(\frac{p}{p + n} - \frac{P}{P + N} \right) \quad (4)$$

$$q_g(X, C) = \frac{p}{n + g} \quad (5)$$

Let us now investigate whether the heuristics used in CSM, EPM and SD are compatible, using the following definition of compatibility.

Definition 3: Compatibility of heuristics.

Heuristic function h_1 is compatible with h_2 if h_2 can be derived from h_1 and if for any two rules R and R' , $h_1(R) > h_1(R') \Leftrightarrow h_2(R) > h_2(R')$.

Lemma 3: *Definitions of CSM, EPM and SD heuristics are pairwise compatible.*

Proof The proof of Lemma 3 is established by proving two sub-lemmas, Lemma 3a and Lemma 3b, which prove the compatibility of two pairs of heuristics, whereas the relationships between these pairs is established through Table 5, and illustrated in Figures 6 and 7. ■

Lemma 3a: *The support difference heuristic used in CSM and the weighted relative accuracy heuristic used in SD are compatible.*

Proof Note that, as shown below, weighted relative accuracy (Equation 4) can be interpreted in terms of probabilities of rule antecedent X and consequent Y (class C representing the property of interest), and the conditional probability of class Y given X , estimated by relative frequencies.

$$WRAcc(X, Y) = P(X) \cdot (P(Y|X) - P(Y)) \quad (6)$$

From this equation we see that, indeed, when optimizing weighted relative accuracy of rule $X \rightarrow Y$, we optimize two contrasting factors: rule coverage $P(X)$ (proportional to the size of the subgroup), and distributional unusualness $P(Y|X) - P(Y)$ (proportional to the difference of the number of positive examples correctly covered by the rule and the number of positives in the original training set). It is straightforward to show that this measure is equivalent to the Piatetski-Shapiro measure, which evaluates the conditional (in)dependence of rule consequent and rule antecedent as follows:

$$PS(X, Y) = P(X \cdot Y) - P(X) \cdot P(Y)$$

Weighted relative accuracy, known from subgroup discovery, and support difference between groups, used in contrast set mining, are related as follows:⁴

$$\begin{aligned}
 WRAcc(X, Y) &= \\
 &= P(X) \cdot [P(Y|X) - P(Y)] = P(Y \cdot X) - P(Y) \cdot P(X) \\
 &= P(Y \cdot X) - P(Y) \cdot [P(Y \cdot X) + P(\bar{Y} \cdot X)] \\
 &= (1 - P(Y)) \cdot P(Y \cdot X) - P(Y) \cdot P(\bar{Y} \cdot X) \\
 &= P(\bar{Y}) \cdot P(Y) \cdot P(X|Y) - P(Y) \cdot P(\bar{Y}) \cdot P(X|\bar{Y}) \\
 &= P(\bar{Y}) \cdot P(Y) \cdot [P(X|Y) - P(X|\bar{Y})] \\
 &= P(Y) \cdot P(\bar{Y}) \cdot [TPR(X, Y) - FPR(X, Y)]
 \end{aligned}$$

Since the distribution of examples among classes is constant for any dataset, the first two factors $P(Y)$ and $P(\bar{Y})$ are constant within a dataset. Therefore, when maximizing the weighted relative accuracy, one is maximizing the second factor $TPR(X, Y) - FPR(X, Y)$, which actually is support difference when we have a two group contrast set mining problem. Consequently, for $C = G_1$, and $\bar{C} = G_2$ the following holds:

$$WRAcc(X, C) = WRAcc(X, G_1) = P(G_1) \cdot P(G_2) \cdot [support(X, G_1) - support(X, G_2)]$$

■

Lemma 3b: *The growth rate heuristic used in EPM and the generalization quotient heuristic used in SD are compatible.*

Proof Equation 3 can be rewritten as follows:

$$\begin{aligned}
 GrowthRate(X, D_1, D_2) &= \frac{support(X, D_1)}{support(C, D_2)} = \\
 &= \frac{count(X, D_1)}{count(X, D_2)} \cdot \frac{|D_2|}{|D_1|} = \frac{p}{n} \cdot \frac{N}{P}
 \end{aligned}$$

Since the distribution of examples among classes is constant for any dataset, the quotient $\frac{N}{P}$ is constant. Consequently, the growth rate is the generalization quotient with $g = 0$, multiplied by a constant. Therefore, the growth rate is compatible with the generalization quotient.

$$GrowthRate(X, C, \bar{C}) = q_0(X, C) \cdot \frac{N}{P}$$

■

The lemmas prove that heuristics used in CSM and EPM can be translated into heuristics used in SD and vice versa. In this way, we have shown the compatibility of CSM and SD heuristics, as well as the compatibility of EPM and SD heuristics. While the lemmas do not prove direct compatibility of CSM and EPM heuristics, they prove that heuristics used in SCM and EPM can be translated into two heuristics used in SD, both aiming at trading-off between coverage and distributional difference.

4. Peter A. Flach is acknowledged for having derived these equations.

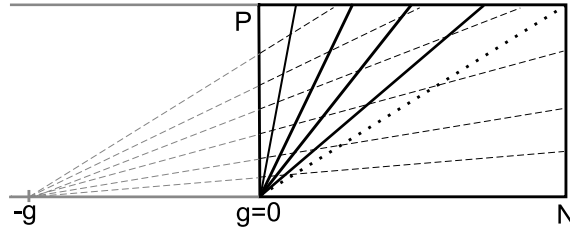


Figure 6: Isometrics for q_g . The dotted lines show the isometrics for a selected $g > 0$, while the full lines show the special case when $g = 0$, compatible to the EPM *growth rate* heuristic.

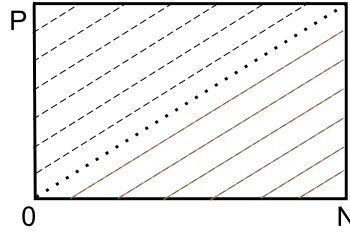


Figure 7: Isometrics for $WRAcc$, compatible to the CSM *support difference* heuristic.

Table 5 provides also the equivalents of these heuristics in terms of heuristics known from the classification rule learning community, details of which are beyond the scope of this paper (an interested reader can find more details on selected heuristics and their ROC representations in Fürnkranz and Flach, 2003).

Note that the growth rate heuristic from EPM, as a special case of the generalization quotient heuristic with $g = 0$, does not consider rule coverage. On the other hand, its compatible counterpart, the generalization quotient q_g heuristic used in SD, can be tailored to favor more general rules by setting the g parameter value, as for a general g value, the q_g heuristic provides a trade-off between rule accuracy and coverage. Figure 6⁵ illustrates the q_g isometrics, for a general g value, as well as for value $g = 0$.

Note also that standard rule learners (such as CN2 by Clark and Niblett, 1989) tend to generate very specific rules, due to using accuracy heuristic $Acc(X, Y) = \frac{p+\bar{n}}{p+N}$ or its variants: the Laplace and the m -estimate. On the other hand, the CSM support difference heuristic and its SD counterpart $WRAcc$ both optimize a trade-off between rule accuracy and coverage. The $WRAcc$ isometrics are plotted in Figure 7⁶.

3.4 Comparison of Rule Selection Mechanisms

Having established a unifying view on the terminology, definitions and rule learning heuristics, the last step is to analyze rule selection mechanisms used by different algorithms. The motivation for rule selection can be either to find only significant rules or to avoid overlapping rules (too many

5. This figure is due to Gamberger and Lavrač (2002).

6. This figure is due to Fürnkranz and Flach (2003).

too similar rules), or to avoid showing redundant rules to the end users. Note that rule selection is not always necessary and that depending on the goal, redundant rules can be valuable (e.g., classification by aggregating emerging patterns by Dong et al., 1999). Two approaches are commonly used: statistic tests and the (weighted) covering approach. In this section, we compare these two approaches.

Webb et al. (2003) show that contrast set mining is a special case of the more general rule discovery task. However, an experimental comparison of STUCCO, OPUS_AR and C4.5 has shown that standard rule learners return a larger set of rules compared to STUCCO, and that some of them are also not interesting to end users. STUCCO (see Bay and Pazzani (2001) for more details) uses several mechanisms for rule pruning. Statistical significance pruning removes contrast sets that, while significant and large, derive these properties only due to being specializations of more general contrast sets: any specialization is pruned that has a similar support to its parent or that fails a χ^2 test of independence with respect to its parent.

In the context of OPUS_AR, the emphasis has been on developing statistical tests that are robust in the context of the large search spaces explored in many rule discovery applications Webb (2007). These include tests for independence between the antecedent and consequent, and tests to assess whether specializations have significantly higher confidence than their generalizations.

In subgroup discovery the *weighted covering approach* (Lavrač et al., 2004b) is used with the aim of ensuring the diversity of rules induced in different iterations of the algorithm. In each iteration, after selecting the best rule, the weights of positive examples are decreased according to the number of rules covering each positive example $rule\_count(e)$; they are set to $w(e) = \frac{1}{rule\_count(e)}$. For selecting the best rule in consequent iterations, the SD algorithm (Gamberger and Lavrač, 2002) uses—instead of the unweighted q_g measure (Equation 5)—the weighted variant of q_g defined in Equation 7, while the CN2-SD (Lavrač et al., 2004b) and APRIORI-SD (Kavšek and Lavrač, 2006) algorithms use the weighted relative accuracy (Equation 4) modified with example weights, as defined in Equation 8, where $p' = \sum_{TP(X,Y)} w(e)$ is the sum of the weights of all covered positive examples, and P' is the sum of the weights of all positive examples.

$$q'_g(X, Y) = \frac{p'}{n + g} \quad (7)$$

$$WRAcc'(X, Y) = \frac{p' + n}{P' + N} \cdot \left(\frac{p'}{p' + n} - \frac{P}{P + N} \right) \quad (8)$$

Unlike in the sections on the terminology, task definitions and rule learning heuristics, the comparison of rule pruning mechanisms described in this section does not result in a unified view; although the goals of rule pruning may be the same, the pruning mechanisms used in different subareas of supervised descriptive rule discovery are—as shown above—very different.

4. Visualization

Webb et al. (2003) identify a need to develop appropriate methods for presenting contrast sets to end users, possibly through contrast set visualization. This open issue, concerning the visualization of contrast sets and emerging patterns, can be resolved by importing some of the solutions proposed in the subgroup discovery community. Several methods for subgroup visualization were developed by Wettschereck (2002); Wrobel (2001); Gamberger et al. (2002); Kralj et al. (2005); Atzmüller and Puppe (2005). They are here illustrated using the coronary heart disease dataset, originally analyzed

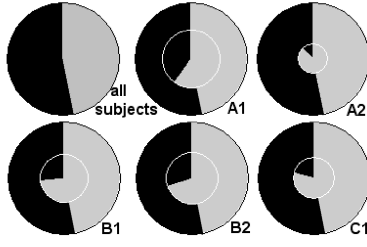


Figure 8: Subgroup visualization by pie charts.

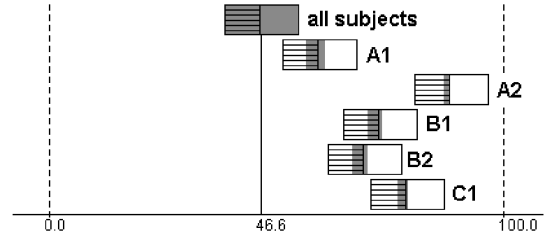


Figure 9: Subgroup visualization by box plots.

by Gamberger and Lavrač (2002). The visualizations are evaluated by considering their intuitiveness, correctness of displayed data, usefulness, ability to display contents besides the numerical properties of subgroups, (e.g., plot subgroup probability densities against the values of an attribute), and their extensibility to multi-class problems.

4.1 Visualization by Pie Charts

Slices of pie charts are the most common way of visualizing parts of a whole. They are widely used and understood. Subgroup visualization by pie chart, proposed by Wettschereck (2002), consists of a two-level pie for each subgroup. The base pie represents the distribution of individuals in terms of the property of interest of the entire example set. The inner pie represents the size and the distribution of individuals in terms of the property of interest in a specific subgroup. An example of five subgroups (subgroups A1, A2, B1, B2, C1), as well as the base pie “all subjects” are visualized by pie charts in Figure 8.

The main weakness of this visualization is the misleading representation of the relative size of subgroups. The size of a subgroup is represented by the radius of the circle. The faultiness arises from the surface of the circle which increases with the square of its radius. For example, a subgroup that covers 20% of examples is represented by a circle that covers only 4% of the whole surface, while a subgroup that covers 50% of examples is represented by a circle that covers 25% of the whole surface. In terms of usefulness, this visualization is not very handy since—in order to compare subgroups—one would need to compare sizes of circles, which is difficult. The comparison of distributions in subgroups is also not straightforward. This visualization also does not show the contents of subgroups. It would be possible to extend this visualization to multi-class problems.

4.2 Visualization by Box Plots

In subgroup visualization by box plots, introduced by Wrobel (2001), each subgroup is represented by one box plot (all examples are also considered as one subgroup and are displayed in the top box). Each box shows the entire population; the horizontally stripped area on the left represents the positive examples and the white area on the right-hand side of the box represents the negative examples. The grey area within each box indicates the respective subgroup. The overlap of the grey area with the hatched area shows the overlap of the group with the positive examples. Hence, the more to the left the grey area extends the better. The less the grey area extends to the right of the hatched area, the more specific a subgroup is (less overlap with the subjects of the negative class). Finally, the location of the box along the X-axis indicates the relative share of the target class within each subgroup: the more to the right a box is placed, the higher is the share of the target value

within this subgroup. The vertical line (in Figure 9 at value 46.6%) indicates the default accuracy, i.e., the number of positive examples in the entire population. An example box plot visualization of five subgroups is presented in Figure 9.

On the negative side, the intuitiveness of this visualization is relatively poor since an extensive explanation is necessary for understanding it. It is also somewhat illogical since the boxes that are placed more to the right and have more grey color on the left-hand side represent the best subgroups. This visualization is not very attractive since most of the image is white; the grey area (the part of the image that really represents the subgroups) is a relatively tiny part of the entire image. On the positive side, all the visualized data are correct and the visualization is useful since the subgroups are arranged by their confidence. It is also easier to contrast the sizes of subgroups compared to their pie chart visualization. However, this visualization does not display the contents of the data. It would also be difficult to extend this visualization to multi-class problems.

4.3 Visualizing Subgroup Distribution w.r.t. a Continuous Attribute

The distribution of examples w.r.t. a continuous attribute, introduced by Gamberger and Lavrač (2002) and Gamberger et al. (2002), was used in the analysis of several medical domains. It is the only subgroup visualization method that offers an insight of the visualized subgroups. The approach assumes the existence of at least one numeric (or ordered discrete) attribute of expert's interest for subgroup analysis. The selected attribute is plotted on the X-axis of the diagram. The Y-axis represents the target variable, or more precisely, the number of instances belonging to target property C (shown on the $Y+$ axis) or not belonging to C (shown on the $Y-$ axis) for the values of the attribute on the X-axis. It must be noted that both directions of the Y-axis are used to indicate the number of instances. The entire dataset and two subgroups A1 and B2 are visualized by their distribution over a continuous attribute in Figure 10.

This visualization method is not completely automatic, since the automatic approach does not provide consistent results. The automatic approach calculates the number of examples for each value of the attribute on the X-axis by moving a sliding window and counting the number of examples in that window. The outcome is a smooth line. The difficulty arises when the attribute from the X-axis appears in the subgroup description. In such a case, a manual correction is needed for this method to be realistic.

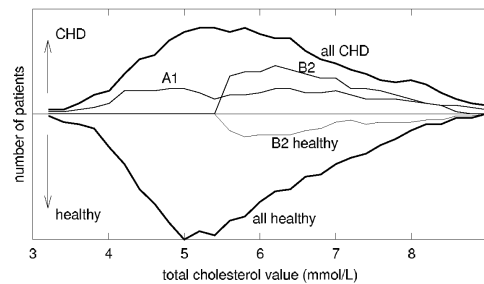


Figure 10: Subgroup visualization w.r.t. a continuous attribute. For clarity of the picture, only the positive ($Y+$) side of subgroup A1 is depicted.

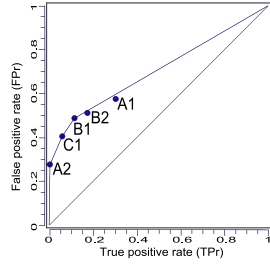


Figure 11: Representation of subgroups in the ROC space.

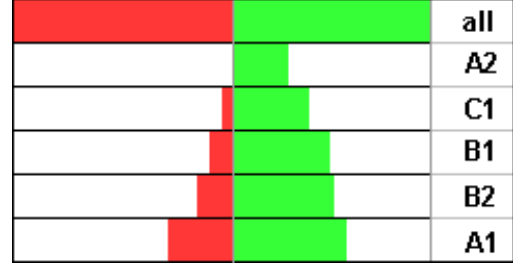


Figure 12: Subgroup visualization by bar charts.

This visualization method is very intuitive since it practically does not need much explanation. It is attractive and very useful to the end user since it offers an insight in the contents of displayed examples. However, the correctness displayed data is questionable. It is impossible to generalize this visualization to multi-class problems.

4.4 Representation in the ROC Space

The ROC (Receiver Operating Characteristics) (Provost and Fawcett, 2001) space is a 2-dimensional space that shows classifier (rule/rule set) performance in terms of its false positive rate (FPr) plotted on the X-axis, and true positive rate (TPr) plotted on the Y-axis. The ROC space is appropriate for measuring the success of subgroup discovery, since subgroups whose $\frac{TPr}{FPr}$ tradeoffs are close to the main diagonal (line connecting the points (0, 0) and (1, 1) in the ROC space) can be discarded as insignificant (Kavšek and Lavrač, 2006); the reason is that the rules with the $\frac{TPr}{FPr}$ ration on the main diagonal have the same distribution of covered positives and negatives ($TPr = FPr$) as the distribution in the entire dataset. An example of five subgroups represented in the ROC space is shown in Figure 11.

Even though the ROC space is an appropriate rule visualization, it is usually used just for the evaluation of discovered rules. The ROC convex hull is the line connecting the potentially optimal subgroups. The area under the ROC convex hull (AUC, area under curve) is a measure of quality of the resulting ruleset.⁷

This visualization method is not intuitive to the end user, but is absolutely clear to every machine learning expert. The displayed data is correct, but there is no content displayed. An advantage of this method compared to the other visualization methods is that it allows the comparison of outcomes of different algorithms at the same time. The ROC space is designed for two-class problems and is therefore inappropriate for multi-class problems.

4.5 Bar Chats Visualization

The visualization by bar charts was introduced by Kralj et al. (2005). In this visualization, the purpose of the first line is to visualize the distribution of the entire example set. The area on the right represents the positive examples and the area on the left represents the negative examples of the

7. Note that in terms of $\frac{TPr}{FPr}$ ratio optimality, two subgroups (A1 and B2) are suboptimal, lying below the ROC convex hull.

target class. Each following line represents one subgroup. The positive and the negative examples of each subgroup are drawn below the positive and the negative examples of the entire example set. Subgroups are sorted by the relative share of positive examples (precision).

An example of five subgroups visualized by bar charts is shown in Figure 12. It is simple, understandable and shows all the data correctly. This visualization method allows simple comparison between subgroups and is therefore useful. It is relatively straight-forward to understand and can be extended to multi-class problems. It does not display the contents of data, though.

4.6 Summary of Subgroup Visualization Methods

In this section we (subjectively) compare the five different subgroup visualization methods by considering their intuitiveness, correctness of displayed data, usefulness, ability to display contents besides the numerical properties of subgroups, (e.g., plot subgroup probability densities against the values of an attribute), and their extensibility to multi-class problems. The summary of the evaluation is presented in Table 6.

	Pie chart	Box plot	Continuous attribute	ROC	Bar chart
Intuitiveness	+	-	+	+/-	+
Correctness	-	+	-	+	+
Usefulness	-	+	+	+	+
Contents	-	-	+	-	-
Multi-class	+	-	-	-	+

Table 6: Our evaluation of subgroup visualization methods.

Two visualizations score best in Table 6 of our evaluation of subgroup visualization methods: the visualization of subgroups w.r.t. a continuous attribute and the bar chart visualization. The visualization of subgroups w.r.t. a continuous attribute is the only visualization that directly shows the contents of the data; its main shortcomings are the doubtful correctness of the displayed data and its difficulty to be extended to multi-class problems. It also requires a continuous or ordered discrete attribute in the data. The bar chart visualization combines the good properties of the pie chart and the box plot visualization. In Table 6 it only fails in displaying the contents of the data. By using the two best visualizations, one gets a very good understanding of the mining results.

To show the applicability of subgroup discovery visualizations for supervised descriptive rule discovery, the bar visualizations of results of contrast set mining, jumping emerging patterns and subgroup discovery on the survey data analysis problem of Section 2 are shown in Figures 13, 14 and 15, respectively.

5. Conclusions

Patterns in the form of rules are intuitive, simple and easy for end users to understand. Therefore, it is not surprising that members of different communities have independently addressed supervised descriptive rule induction, each of them solving similar problems in similar ways and developing vocabularies according to the conventions of their respective research communities.

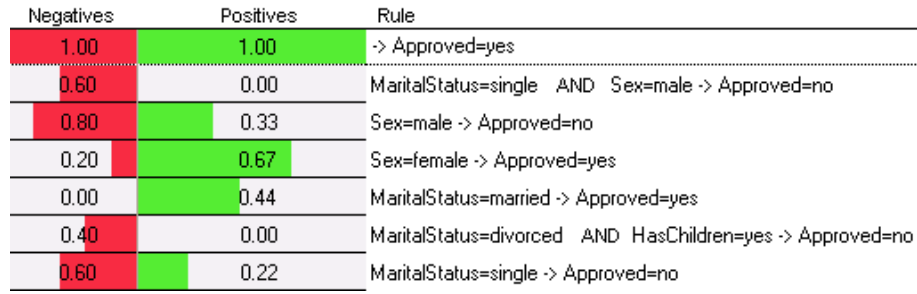


Figure 13: Bar visualization of contrasting sets of Figure 3.

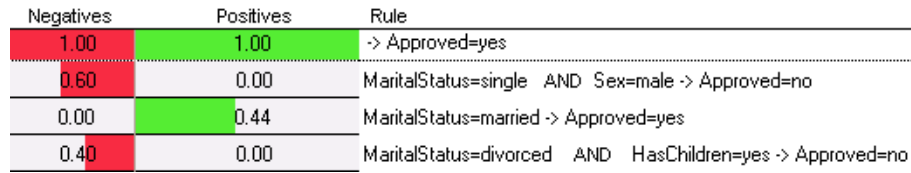


Figure 14: Bar visualization of jumping emerging patterns of Figure 4.

This paper sheds a new light on previous work in this area by providing a systematic comparison of the terminology, definitions, goals, algorithms and heuristics of contrast set mining (CSM), emerging pattern mining (EPM) and subgroup discovery (SD) in a unifying framework called supervised descriptive rule discovery. We have also shown that the heuristics used in CSM and EPM can be translated into two well-known heuristics used in SD, both aiming at trading-off between coverage and distributional difference. In addition, the paper presents a critical survey of existing visualization methods, and shows that some methods used in subgroup discovery can be easily adapted for use in CSM and EPM.

Acknowledgements

The work of Petra Kralj and Nada Lavrač was funded by the project Knowledge Technologies (grant no. P2-0103) funded by the Slovene National Research Agency, and co-funded by the European 6FP project IQ - *Inductive Queries for Mining Patterns and Models* (IST-FP6-516169). Geoff Webb's contribution to this work has been supported by Australian Research Council Grant DP0772238.

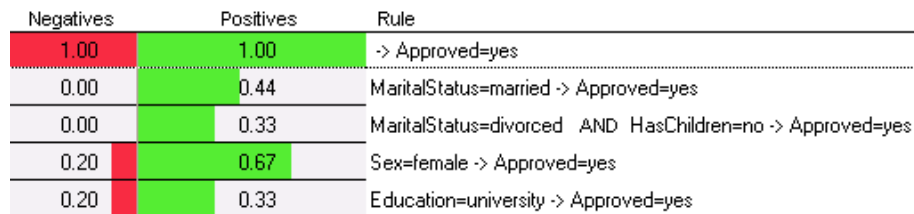


Figure 15: Bar visualization of subgroups of Figure 5 of individuals who have approved the issue.

References

- Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.
- Martin Atzmüller and Frank Puppe. SD-Map - a fast algorithm for exhaustive subgroup discovery. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, pages 6–17, 2006.
- Martin Atzmüller and Frank Puppe. Semi-automatic visual subgroup mining using VIKAMINE. *Journal of Universal Computer Science (JUCS), Special Issue on Visual Data Mining*, 11(11): 1752–1765, 2005.
- Martin Atzmüller, Frank Puppe, and Hans-Peter Buscher. Exploiting background knowledge for knowledge-intensive subgroup discovery. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 647–652, 2005a.
- Martin Atzmüller, Frank Puppe, and Hans-Peter Buscher. Profiling examiners using intelligent subgroup mining. In *Proceedings of the 10th Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-05)*, pages 46–51, 2005b.
- Yonatan Aumann and Yehuda Lindell. A statistical theory for quantitative association rules. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 261–270, 1999.
- Stephen D. Bay. Multivariate discretization of continuous variables for set mining. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, pages 315–319, 2000.
- Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.
- Roberto J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, pages 85–93, 1998.
- Anne-Laure Boulesteix, Gerhard Tutz, and Korbinian Strimmer. A CART-based approach to discover emerging patterns in microarray data. *Bioinformatics*, 19(18):2465–2472, 2003.
- Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
- William W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pages 115–123, 1995.
- Olena Daly and David Taniar. Exception rules in data mining. In *Encyclopedia of Information Science and Technology (II)*, pages 1144–1148. 2005.

- María José del Jesus, Pedro González, Francisco Herrera, and Mikel Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *IEEE Transactions on Fuzzy Systems*, 15(4):578–592, 2007.
- Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 43–52, 1999.
- Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by aggregating emerging patterns. In *Proceedings of the 2nd International Conference on Discovery Science (DS-99)*, pages 30–42, 1999.
- Hongjian Fan and Kotagiri Ramamohanara. A bayesian approach to use emerging patterns for classification. In *Proceedings of the 14th Australasian Database Conference (ADC-03)*, pages 39–48, 2003.
- Hongjian Fan and Kotagiri Ramamohanarao. Efficiently mining interesting emerging patterns. In *Proceeding of the 4th International Conference on Web-Age Information Management (WAIM-03)*, pages 189–201, 2003.
- Hongjian Fan, Ming Fan, Kotagiri Ramamohanarao, and Mengxu Liu. Further improving emerging pattern based classifiers via bagging. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-06)*, pages 91–96, 2006.
- Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.
- Johannes Fürnkranz and Peter A. Flach. An analysis of rule evaluation metrics. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 202–209, 2003.
- Dragan Gamberger and Nada Lavrač. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.
- Dragan Gamberger, Nada Lavrač, and Dietrich Wettschereck. Subgroup visualization: A method and application in population screening. In *Proceedings of the 7th International Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-02)*, pages 31–35, 2002.
- Gemma C. Garriga, Petra Kralj, and Nada Lavrač. Closed sets for labeled data. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, pages 163 – 174, 2006.
- Robert J. Hilderman and Terry Peckham. A statistically sound alternative approach to mining contrast sets. In *Proceedings of the 4th Australia Data Mining Conference (AusDM-05)*, pages 157–172, 2005.
- Jože Jenkole, Petra Kralj, Nada Lavrač, and Alojzije Sluga. A data mining experiment on manufacturing shop floor data. In *Proceedings of the 40th International Seminar on Manufacturing Systems (CIRP-07)*, 2007. 6 pages.

- Branko Kavšek and Nada Lavrač. APRIORI-SD: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20(7):543–583, 2006.
- Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. *Advances in Knowledge Discovery and Data Mining*, pages 249–271, 1996.
- Willi Klösgen and Michael May. Spatial subgroup mining integrated in an object-relational spatial database. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-02)*, pages 275–286, 2002.
- Willi Klösgen, Michael May, and Jim Petch. Mining census data for spatial effects on mortality. *Intelligent Data Analysis*, 7(6):521–540, 2003.
- Ron Kohavi and Foster Provost, editors. *Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Glossary of Terms*, 1998.
- Petra Kralj, Nada Lavrač, and Blaž Zupan. Subgroup visualization. In *8th International Multiconference Information Society (IS-05)*, pages 228–231, 2005.
- Petra Kralj, Ana Rotter, Nataša Toplak, Kristina Gruden, Nada Lavrač, and Gemma C. Garriga. Application of closed itemset mining for class labeled data in functional genomics. *Informatica Medica Slovenica*, (1):40–45, 2006.
- Petra Kralj, Nada Lavrač, Dragan Gamberger, and Antonija Krstačić. Contrast set mining for distinguishing between similar diseases. In *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME-07)*, pages 109–118, 2007a.
- Petra Kralj, Nada Lavrač, Dragan Gamberger, and Antonija Krstačić. Contrast set mining through subgroup discovery applied to brain ischaemia data. In *Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining : (PAKDD-07)*, pages 579–586, 2007b.
- Nada Lavrač, Bojan Cestnik, Dragan Gamberger, and Peter A. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning Special issue on Data Mining Lessons Learned*, 57(1-2):115–143, 2004a.
- Nada Lavrač, Branko Kavšek, Peter A. Flach, and Ljupčo Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004b.
- Nada Lavrač, Petra Kralj, Dragan Gamberger, and Antonija Krstačić. Supporting factors to improve the explanatory potential of contrast set mining: Analyzing brain ischaemia data. In *Proceedings of the 11th Mediterranean Conference on Medical and Biological Engineering and Computing (MEDICON-07)*, pages 157–161, 2007.
- Jinyan Li and Limsoon Wong. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics*, 18(10):1406–1407, 2002.
- Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *In proceedings of the 14th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*, pages 191–200, 2000.

- Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems*, 3(2):1–29, 2001.
- Jinyan Li, Huiqing Liu, James R. Downing, Allen Eng-Juh Yeoh, and Limsoon Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. *Bioinformatics*, 19(1):71–78, 2003.
- Jessica Lin and Eamonn Keogh. Group SAX: Extending the notion of contrast sets to time series and multimedia data. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, pages 284–296, 2006.
- Bing Liu, Wynne Hsu, Heng-Siew Han, and Yiyuan Xia. Mining changes for real-life applications. In *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK-2000)*, pages 337–346, 2000.
- Bing Liu, Wynne Hsu, and Yiming Ma. Discovering the set of fundamental rule changes. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 335–340, 2001.
- Michael May and Lemonnia Ragia. Spatial subgroup discovery applied to the analysis of vegetation data. In *Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management (PAKM-2002)*, pages 49–61, 2002.
- Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.
- J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
- J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- Mondelle Simeon and Robert J. Hilderman. Exploratory quantitative contrast set mining: A discretization approach. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Vol.2 (ICTAI-07)*, pages 124–131, 2007.
- K.K.W. Siu, S.M. Butler, T. Beveridge, J.E. Gillam, C.J. Hall, A.H. Kaye, R.A. Lewis, K. Mannan, G. McLoughlin, S. Pearson, A.R. Round, E. Schultke, G.I. Webb, and S.J. Wilkinson. Identifying markers of pathology in SAXS data of malignant tissues of the brain. *Nuclear Instruments and Methods in Physics Research A*, 548:140–146, 2005.
- Hee S. Song, Jae K. Kimb, and Soung H. Kima. Mining the change of customer behavior in an internet shopping mall. *Expert Systems with Applications*, 21(3):157–168, 2001.
- Arnaud Soulet, Bruno Crmilleux, and Francois Rioult. Condensed representation of emerging patterns. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-04)*, pages 127–132, 2004.
- Einoshin Suzuki. Data mining methods for discovering interesting exceptions from an unsupervised table. *Journal of Universal Computer Science*, 12(6):627–653, 2006.
- Filip Železný and Nada Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62:33–63, 2006.

- Ke Wang, Senqiang Zhou, Ada W.-C. Fu, and Jeffrey X. Yu. Mining changes of classification by correspondence tracing. In *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM-03)*, pages 95–106, 2003.
- Geoffrey I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.
- Geoffrey I. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.
- Geoffrey I. Webb. Discovering associations with numeric variables. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 383–388, 2001.
- Geoffrey I. Webb, Shane M. Butler, and Douglas Newlands. On detecting differences between groups. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pages 256–265, 2003.
- Dietrich Wettschereck. A KDDSE-independent PMML visualizer. In *Proceedings of 2nd Workshop on Integration Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-02)*, pages 150–155, 2002.
- Tzu-Tsung Wong and Kuo-Lung Tseng. Mining negative contrast sets from data with discrete attributes. *Expert Systems with Applications*, 29(2):401–407, 2005.
- Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the 1st European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, 1997.
- Stefan Wrobel. Inductive logic programming for knowledge discovery in databases. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, chapter 4, pages 74–101. 2001.

Cluster-Grouping: From Subgroup Discovery to Clustering^{*}

Albrecht Zimmermann and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan
200A, 3001 Leuven, P.O. Box 2402, Belgium
{Albrecht.Zimmermann,Luc.DeRaedt}@cs.kuleuven.be

Abstract. We introduce the problem of **cluster-grouping** and show that it is a subtask in several important data mining tasks, such as *subgroup discovery*, mining correlated patterns, *clustering* and *classification*. The algorithm CG for solving *cluster-grouping* problems is then introduced, and it is incorporated as a component in several existing and novel algorithms for tackling *subgroup discovery*, *clustering* and *classification*. The resulting systems are empirically compared to state-of-the-art systems such as CN2, CBA, Ripper, Autoclass and COBWEB. The results indicate that the CG algorithm is useful as a generic local pattern mining component in a wide variety of data mining and machine learning algorithms.

Keywords: correlated pattern mining, subgroup discovery, associative classification, clustering.

1 Introduction

The representation of conjunctive rules occupies a central position in the field of symbolic machine learning and data mining. It is used to represent local patterns in the data and sets of such rules form the output of a wide variety of systems, tackling diverse tasks ranging from association rule mining, correlated pattern mining, subgroup discovery, rule learning to conceptual clustering. [1, 29, 23, 10, 33] These systems all possess a local pattern mining or rule learning component, which raises the question as to whether there exists a unified or universal local pattern mining approach that can be used across these systems. The key contribution of the present paper is that we answer this question positively, by first, introducing the task of *cluster-grouping*, based on an extension of the work of [29], and secondly, that we develop an algorithm, CG, employing sophisticated pruning techniques for solving this task. The algorithm guarantees to find the best pattern (or rule) with regard to a correlation measure (e.g. χ^2), and forms an alternative to the often heuristic (beam-search) methods employed in machine learning.

As evidence for the wide applicability of the CG algorithm, we use it together with different wrappers to tackle the tasks of correlated pattern mining, subgroup

^{*} This paper integrates and extends an abstract published at ECML 2004 [38] as well as related work, published at DS 2004 [39], and as a book contribution [40].

discovery, classification and conceptual clustering. The resulting systems are then empirically evaluated on a large number of UCI data sets [3] and compared to state-of-the-art machine learning and data mining systems such as CN2-SD, RIPPER, CBA, and COBWEB [23, 10, 25, 13]. This also results in number of novel systems. Especially interesting are the novel CBC system, for realizing associative classification using correlated patterns rather than pure association rules as CBA [25] and CMAR [24] do, and the novel CG-CLUS system, based on a divisive decision-tree like algorithm, for realizing conceptual clustering. The tests in the CG-CLUS trees are based on conjunctive descriptions. The results of the experiments provide evidence for the key claims of this work – that the *cluster-grouping* task and CG algorithm are useful as a component accross a wide variety of data mining and machine learning algorithms – but also indicate that using the CG algorithm instead of the common beam-search are often advantageous for what concerns efficiency and performance.

We proceed as follows. In the next section we introduce the concept of local pattern mining. In Section 3, we present the underlying principles of CG, the algorithm we develop for addressing the local pattern mining task. In Section 4 we introduce the general mechanism for combining local patterns into a global model. Additionally, we show how different data mining tasks can be cast in this description, describe influential systems based on existing paradigms and experimentally compare CG-based systems to existing solutions. In Section 5 we refer to related work before we conclude in the last section.

2 Local Pattern Mining

Throughout the paper, we use attribute-value representations, and hence, employ conjunctions of attribute-value pairs to describe patterns. Such patterns are called *local* patterns if they describe instances that show an unexpectedly high density of certain attribute values compared to a background model. In the case of CG, the background model is supplied by observed distributions of target attributes' values. A local pattern is considered *interesting* if the difference in density exceeds a given threshold. More formally, let $\mathcal{A} = \{A_1, \dots, A_d\}$ be a tuple of attributes and $\mathcal{V}[A] = \{v_1, \dots, v_p\}$ the domain of A . A tuple $\langle v^1, \dots, v^d \rangle$ with $v^i \in \mathcal{V}[A_i]$ is called an *instance*. A multiset $\mathcal{E} = \{e_1, \dots, e_n\}$ of instances is called a *data set*.

Definition 1 (*Condition*) A *condition* l is an attribute-value-pair $A = v$ with $v \in \mathcal{V}[A]$. An instance $\langle v_1, \dots, v_d \rangle$ is **covered** by a condition l of the form $A_i = v$ iff $v_i = v$.

Definition 2 (*Pattern*) A *pattern* p is a conjunction of conditions, $l_1 \wedge \dots \wedge l_i$. An instance e is covered by p iff it is covered by all its conditions.

We define certain attributes $A_i^T \in \mathcal{A}$ as *target attributes* and define target conditions. The background model against which the quality of an observed pattern is evaluated is given by these conditions' distribution in the data set.

Table 1. Contingency table for p w.r.t. $\{A^T\}$

	$A^T = v_1$	$A^T = v_2$	
p	$\text{sup}(p \wedge A^T = v_1) = y_1^+$	$\text{sup}(p \wedge A^T = v_2) = y_1^-$	$\text{sup}(p) = y_1^+ + y_1^-$
$\neg p$	$\text{sup}(\neg p \wedge A^T = v_1) = m_1 - y_1^+$	$\text{sup}(\neg p \wedge A^T = v_2) = n - m_1 - y_1^-$	$\text{sup}(\neg p) = n - (y_1^+ + y_1^-)$
	$\text{sup}(A^T = v_1) = m_1$	$\text{sup}(A^T = v_2) = n - m_1$	n

We consider patterns to be *interesting* if the distribution of the values of the target attributes deviates unexpectedly from the background distribution. To quantify the quality of a given pattern, different *interestingness* measures can be used, such as accuracy or confidence, or correlation measures, such as χ^2 , *Information Gain*, and *Category Utility*. Accuracy measures the purity of the described population w.r.t. a given target condition, while correlation measures quantify the deviation between assumed distributions of target conditions and the actual distribution in the subspace of instances defined by the patterns.

Definition 3 (*Support*) For a pattern p , we define

$$\text{sup}(p) = |\{e \in \mathcal{E} \mid e \text{ is covered by } p\}|$$

the support of p . The support of a pattern w.r.t. a single target $A^T = v_1$ is

$$\text{sup}(p \wedge A^T = v_1) = |\{e \in \mathcal{E} \mid e \text{ is covered by } p \text{ and } A^T = v_1\}|$$

To facilitate the use of correlation measures, occurrence counts are often organized in contingency tables. A contingency table for a pattern and a single binary-valued target attribute is shown in Table 1.

We use the following notation to refer to occurrence counts of patterns:

Definition 4 (*Occurrence Counts*) For a given pattern p , target attributes $A_1^T, \dots, A_d^T$ and a given data set $\mathcal{E}$ we define:

$$\mathbf{n} = |\mathcal{E}|, \mathbf{m}_i = \text{sup}(A_i^T = v_1), \mathbf{y}_i^+ = \text{sup}[p \wedge (A_i^T = v_1)], \mathbf{y}_i^- = \text{sup}[p \wedge \neg(A_i^T = v_1)]$$

Note that the sum of the cells in a row (column) is equal to the margins of the table, that is the rightmost (down-most) entry in a row (column). Correlation measures compare for a given cell the product of the corresponding margins to the cell count, thus comparing *expected* (under an independence assumption between patterns and target attribute values) to *observed* frequency, and score the difference. Consider for instance the upper left cell of Table 1: the value of the cell itself, the *observed* value, is y^+ . The coverage of the pattern on the entire data is $y^+ + y^-$, as seen on the upper right margin, and the size of $A^T = v_1$ is m_1 , as seen in the lower left margin. This leads to a straight-forward *expected* value for the upper left cell: $m_1 \cdot (y^+ + y^-)/n$. To compare these two values, one can for instance subtract them from each other, squared so that both higher and lower than expected behavior is treated symmetrically: $(y_1^+ - m_1(y_1^+ + y_1^-)/n)^2$.

In the χ^2 measure this term would then be discounted with the expected value, giving a complete term of:

$$\frac{(y_1^+ - m_1(y_1^+ + y_1^-)/n)^2}{m_1(y_1^+ + y_1^-)/n}$$

Table 2. Pseudo-Contingency table for p w.r.t $\{A_1^T, A_2^T\}$

	$A_1^T = v_1$	$A_1^T = v_2$	$A_2^T = v_1$	$A_2^T = v_2$	
p	y_1^+ $= \text{sup}[p \wedge (A_1^T = v_1)]$	y_1^- $= \text{sup}[p \wedge (A_1^T = v_2)]$	y_2^+ $= \text{sup}[p \wedge (A_2^T = v_1)]$	y_2^- $= \text{sup}[p \wedge (A_2^T = v_2)]$	$\text{sup}(p) = y_1^+ + y_1^-$ $= y_2^+ + y_2^-$
$\neg p$	$m_1 - y_1^+$ $= \text{sup}(\neg p \wedge A_1^T = v_1)$	$n - m_1 - y_1^-$ $= \text{sup}(\neg p \wedge A_1^T = v_2)$	$m_2 - y_2^+$ $= \text{sup}(\neg p \wedge A_2^T = v_1)$	$n - m_2 - y_2^-$ $= \text{sup}(\neg p \wedge A_2^T = v_2)$	$\text{sup}(\neg p)$ $= n - (y_1^+ + y_1^-)$
	$\text{sup}(A_1^T = v_1) = m_1$	$n - m_1 = \text{sup}(A_1^T = v_2)$	$m_2 = \text{sup}(A_2^T = v_1)$	$n - m_2 = \text{sup}(A_2^T = v_2)$	n

Increasing the number of involved target attributes usually leads to an increase of dimension of the contingency table to capture all dependencies among the conditions. Our focus is on the effect that pattern presence has on the A_i^T , the target attributes defining the background model. This means that we can disregard dependencies between those target attributes, decreasing the computational complexity of mining processes by instead using *pseudo-contingency* tables such as the one in Table 2. The main difference with regard to a regular high-dimensional contingency table, a so-called multi-way table, is that the margin of a row is not equal to the sum of row-cells anymore. A correlation measure still compares the product of the margins to the cell count.

Definition 5 (Stamp Point [29]) The **stamp point** of a pattern p w.r.t. a data set $\mathcal{E}$, and a set of target attributes $\{A_1^T, \dots, A_d^T\}$, is the tuple of occurrence counts $\langle y_1^+, y_1^-, \dots, y_d^+, y_d^- \rangle$.

Consider an interestingness measure such as accuracy, χ^2 , *Category Utility*, *Information Gain*, or *Weighted Relative Accuracy* defined on a pseudo-contingency table. Since n and the m_i are constant for a given data set, a given interestingness measure $\sigma(p)$ is a function of $2d$ variables

$$\sigma : \mathbb{N}^{2d} \mapsto \mathbb{R},$$

mapping the stamp point $sp(p)$ to a real number.

We can now introduce the *cluster-grouping* problem, which – as we shall argue – can be used in a wide variety of data mining and machine learning problems.

Definition 6 (Cluster-Grouping Problem)**Given:**

- a pattern language $\mathcal{L}$, defining the attribute-value pairs to be used in patterns,
- a data set $\mathcal{E}$,
- an interestingness measure σ ,
- an interestingness threshold τ and/or maximum number of patterns k , and
- a set of target attributes $\{A_1^T, \dots, A_d^T\}$

Find:*A k -theory*

$$Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau, \{A_1^T, \dots, A_d^T\}) = \arg_k \max_{p \in \mathcal{L}} \{\sigma(sp(p)) \geq \tau\}$$

The k -theory consists of the k best patterns expressible in $\mathcal{L}$ according to σ w.r.t. the background model induced by the A_i^T on data set $\mathcal{E}$.

In the next section, we develop a branch-and-bound algorithm for solving *cluster-grouping* problems that is guaranteed to find optimal solutions. This contrasts with some heuristic approaches to solving instances of the *cluster-grouping* problem (such as beam-search) that are sometimes encountered in machine learning, cf. [8, 13].

3 Upper Bound on Convex Correlation Measures

Based on the convexity of correlation measures it is possible to calculate an upper bound on the future value of σ for specializations of a given pattern. This upper bound can be used to prune away parts of the search space known not to produce interesting solutions, and to focus the search on promising parts of the search space. The main insight underlying this technique is that convex functions attain their maximal values at the extreme points of their domain. To our knowledge, this idea was introduced by Morishita *et al.* [29].

3.1 Pattern behaviour in coverage space

Coverage spaces, introduced in [19], can be used to visualize a pattern's or (collection of patterns') coverage behaviour. To this end, a pattern is represented by the number of positives $P = y^+$ and negatives $N = y^-$ it covers (its stamp point w.r.t. a single target attribute), as shown in Figure 1(a). The most general pattern is situated at the upper right corner $(m_1, n - m_1)$, since all instances are covered. When a pattern is specialized (i.e. extended with additional conditions), its stamp point moves to the left and/or downwards, as its coverage decreases.

The diagonal in the diagram corresponds to a proportion of covered positives and covered negatives that is equal to that of the entire data set. At this diagonal, correlation measures evaluate to 0. The farther away from the diagonal a stamp point lies, the more significant it is w.r.t. the background distribution. For a given

pattern p with stamp point $sp(p) = \langle y^+, y^- \rangle$, the point in coverage space that can be reached by any specialization p' and is farthest away from the diagonal, and therefore most significant, is either $(y^+, 0)$ or $(0, y^-)$, the extreme points.

Theorem 1. *The upper bound of a specialization of pattern p w.r.t. a convex correlation measure σ is*

$$ub_\sigma(p) = \max\{\sigma(y^+, 0), \sigma(0, y^-)\}$$

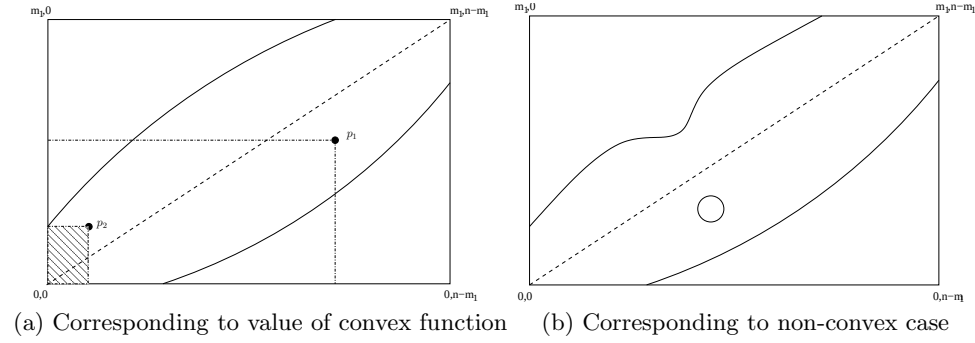


Fig. 1. Coverage space with isometric lines

A pattern evaluation measure induces so-called isometrics in coverage space – curves that connect all coverage points having the same value for the measure. Consider the coverage space shown in Figure 1(a). The two elliptic lines correspond to a χ^2 threshold. A point between one of the isometrics and the diagonal refers to a pattern that does not pass the threshold, such as the patterns shown in the figure. The right pattern, p_1 , has two upper bounds that lie above the threshold, which implies that it is worthy of specialization. The left pattern's (p_2) upper bounds both lie inside the isometrics and therefore no specialization of this pattern can be better than the threshold value, and thus the pattern (and all its specializations) should be pruned away.

3.2 Convexity

Upper bound pruning only works correctly for *convex* functions.

Definition 7 (Convexity) *A function $f : D \mapsto \mathbb{R}$ is convex iff $D \subseteq \mathbb{R}^d$ is a convex set and $\forall x_1, x_2 \in D, \lambda \in [0, 1] : f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda)f(x_2)$.*

This means that, given two points x_1, x_2 , all points x that lie on the line connecting x_1 and x_2 must have a value $f(x)$ that lies on or below the line connecting $f(x_1)$ and $f(x_2)$. Isometrics are just projections of a three-dimensional graph's area onto the two-dimensional plane denoting its domain. If “islands”

and “dents” exist, such as the ones shown in Figure 1(b), the upper bound technique cannot be utilized since a future point might lie in one of the “islands”, thus attaining a higher value than the threshold without lying outside the curves. At the same time, the existence of “islands” and “dents” corresponds to a violation of the convexity criterion.

Functions such as χ^2 , *WRAcc*, *Information Gain*, and *Category Utility* are convex. For the proofs of the convexity of χ^2 and *Information Gain* we refer the reader to [29], while the proofs for *WRAcc* and *Category Utility* can be found in Appendix A.

3.3 Extension to Arbitrary Dimensions

If the mining process considers two or more independent target attributes, as we do, the interestingness measure is additive, meaning that the correlation measure can be evaluated separately for each of the independent target attributes, those finally summed up and averaged and/or normalized.

Definition 8 *A function σ over patterns p with $sp(p) = \langle y_1^+, y_1^-, \dots, y_d^+, y_d^- \rangle$ over attributes $\mathcal{A}_t$ (the stamp point) is additive if*

$$\sigma(p, \mathcal{A}_t) = \sigma(y_1^+, y_1^-, \dots, y_d^+, y_d^-) = c \sum_{i=1}^d \sigma(y_i^+, y_i^-)$$

for some constant c .

Since a sum of convex functions is a convex function itself, and a possible averaging factor c has no effect on convexity, the upper bound technique can be used on the entire sum. However, computing an upper bound is not so easy. There is a naïve upper bound that simply maximizes each summand:

$$ub_\sigma(p) = c \sum_{i=1}^d \max\{\sigma(y_i^+, 0), \sigma(0, y_i^-)\}$$

As shown in the Table 2, however, $\forall i : y_i^+ + y_i^- = sup(p)$, which in turn leads to

$$x = sup(p) = y_1^+ + y_1^- = y_2^+ + y_2^- = \dots = y_d^+ + y_d^-$$

This constraint is potentially violated when each summand is maximized independently of the others.

To illustrate this effect, consider the left-hand side of Figure 2. Shown are overlaid coverage spaces for two different target attributes, with a pattern’s coverage denoted by a dark circle. Note that both the upper right corners of the coverage spaces and the dark circles lie on an isometric with a 135 degree slope – all points lying on this line have the same sum $m_i + (n - m_i)$ (the size-isometric), $y_i^+ + y_i^-$ (the coverage-isometric), respectively. The maximal values, visualized by being farthest from the background-distribution-diagonal, that can be reached

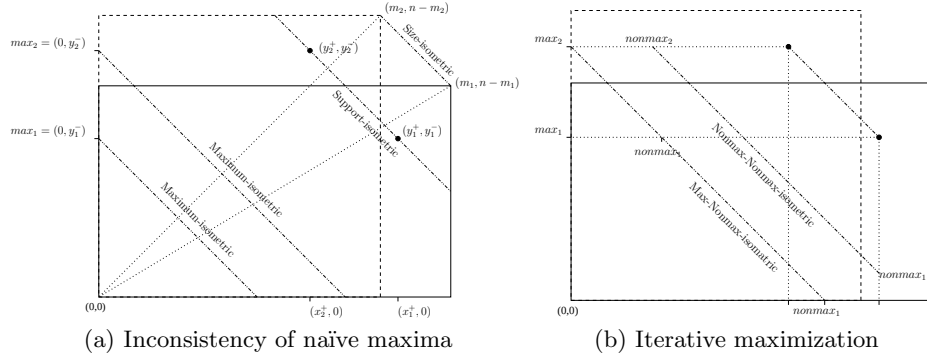


Fig. 2. Coverage spaces for two target attributes

by specializations of the pattern w.r.t. the two target attributes are denoted by max_1 and max_2 (denoted by maximum-isometric), respectively. The equal-sum-isometrics passing through those two values are not the same however, meaning that the respective maximum values would be reached by specializations with different coverage.

Given that the calculation of a non-naïve upper bound for an arbitrary number of target attributes is crucial to the success of our technique, we will give in the next paragraphs an algorithmic description of how to calculate this upper bound, use this algorithm and the isometrics to give an intuition as to why the upper bound is correct and tighter and finally prove its tighter evaluation.

Algorithm 1 Multi-dimensional upper bound calculation

Given: current pattern p , corresponding stamp point $\langle y_1^+, y_1^-, \dots, y_d^+, y_d^- \rangle$

Return: upper bound on $\phi(p')$, with p' a specialization of p

```

ub = 0
for 1 ≤ x ≤ sup(p) - 1 do
  for 1 ≤ i ≤ d do
    ymax+ = min{x, yi+}, ymin- = x - ymax+
    ymax- = min{x, yi-}, ymin+ = x - ymax-
    ubi = max{σ(ymax+, ymin-), σ(ymin+, ymax-)}
  end for
  ub = max{ub, ∑i=1d ubi}
end for
return ub

```

As mentioned above, the main problem with the naïve technique lies in the fact that conflicting support isometrics could be induced by extreme points maximizing σ independently for each target attribute.

The upper bound calculation we use, shown as Algorithm 1, instead calculates an upper bound for every target attribute separately *under a support constraint*

$$ub_i = \max_{y_{i,max}^+ + y_{i,min}^- = y_{i,min}^+ + y_{i,max}^- = x} \{\sigma(y_{i,max}^+, y_{i,min}^-), \sigma(y_{i,min}^+, y_{i,max}^-)\}$$

and then maximizes the sum of these upper bounds over a range of possible supports of specializations of the current pattern

$$ub(\sigma) = \max_{1 \leq x \leq sup(p)-1} \sum_{i=1}^d ub_i$$

To do this, the algorithm iterates over all possible supports of specializations, which lie between 1 (0 would correspond to a pattern covering nothing) and the current pattern support-1 (since identical support corresponds to a more specific pattern with the same informative value) - the outermost for-loop in Algorithm 1.

As can be seen in the right-hand side of Figure 2, such an isometric can correspond to a maximal value of σ for one of the attributes while corresponding to a non-maximal value for another one. The isometric can also correspond to non-maximal values for **both** attributes (the nonmax-nonmax-isometric). What still holds for the purpose of maximizing σ for a single attribute is that those points should be extreme points (furthest away from the background-distribution diagonal). To achieve this, two points $\langle y_{max}^+, y_{min}^- \rangle, \langle y_{min}^+, y_{max}^- \rangle$ are created. Since the support of a pattern, and the current value of y^+ both impose upper bounds on the maximal value of y_{max}^+ , the smaller of the two is chosen. Additionally, since the isometric is fixed, $y_{max}^+ + y_{min}^-$ have to equal x , the specified support. Therefore, y_{min}^- is set to $x - y_{max}^+$. Analogous reasoning holds for $\langle y_{min}^+, y_{max}^- \rangle$.

σ is evaluated on both of these extreme points for an attribute, and the larger value chosen. Finally, the values for all attributes are added up and in this way an upper bound for the score of a hypothetical specialization of support x calculated. By maximizing over all possible future supports, an upper bound for *any* possible specialization of the current pattern is derived.

The preceding discussion explains why this is a *correct* upper bound: maximizing the contribution to σ for each target attribute under a certain support constraint, and doing this for all possible future supports ensures that no future score can exceed this bound. What is left to show is that this bound is *tighter* than the naïve one.

As mentioned in Section 3, convex functions attain their maximal values at the extreme points of their domain. Given the domain induced on a coverage space by y_i^+, y_i^- , it must therefore hold that for all $\langle y_{i,max}^+, y_{i,min}^- \rangle, \langle y_{i,min}^+, y_{i,max}^- \rangle$ defined according to Algorithm 1

$$ub_i = \max\{\sigma(y_{i,max}^+, y_{i,min}^-), \sigma(y_{i,min}^+, y_{i,max}^-)\} \leq \max\{\sigma(y_i^+, 0), \sigma(0, y_i^-)\}$$

This in turn implies

$$\max_{1 \leq x \leq sup(p)-1} \sum_{i=1}^d ub_i \leq \sum_{i=1}^d \max\{\sigma(y_i^+, 0), \sigma(0, y_i^-)\}$$

3.4 The CG-Algorithm

In this section we present an algorithm for solving the *cluster-grouping* problem, called CG. For reasons of readability we show a version for finding patterns having a single highest score value ($k = 1$).

Algorithm 2 The CG algorithm that computes $Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau, \{A_1^T, \dots, A_d^T\}) = \arg_k \max_{p \in \mathcal{L}} \{\sigma(sp(p)) \geq \tau\}$.

$\mathcal{E}$ - data set, σ - correlation measure, τ_{user} - user-defined minimum threshold on σ

```

1:  $P := \{\top\}, \tau := \tau_{user}, S := \emptyset$ 
2: while  $P \neq \emptyset$  do
3:    $p_{mp} := \arg \max_{p \in P} \{ub(p)\}$ 
4:    $C := \rho(p_{mp})$ 
5:   for all  $c_i \in C$  do
6:     compute  $sp(c_i)$ , calculate  $\sigma(c_i)$ 
7:      $ub_\sigma(c_i) := UpperBound(sp(c_i))$ 
8:      $\tau := \max\{\tau, \sigma(c_i)\}$ 
9:   end for
10:   $S := \{s \in S \mid \sigma(s) = \tau\} \cup \{c \in C \mid \sigma(c) = \tau\}$ 
11:   $S := S \setminus \{s \in S \mid \exists s' \in S : s' \prec s \wedge sp(s') = sp(s)\}$ 
12:   $P := \{p \in P \mid ub_\sigma(p) \geq \tau\} \cup \{c \in C \mid ub_\sigma(c) \geq \tau\}$ 
13: end while
14: return  $S$ 

```

The *cluster-grouping* algorithm CG (listed as Algorithm 2) is essentially a branch-and-bound algorithm. Starting from the most general pattern (denoted by $\top$), in each iteration the pattern $p_{mp} \in P$ with the highest upper bound is specialized. We use an optimal refinement operator ρ :

Definition 9 (Optimal Refinement Operator) Let $\mathcal{L}$ be a set of conditions, $\prec$ a total order on the literals in $\mathcal{L}$, $\tau \in \mathbb{R}$.

$$\rho(p) = \{b \wedge l_i \mid l_i \in \mathcal{L}, ub_\sigma(l_i) \geq \tau, \forall l \in r : l \prec l_i\}$$

is an *optimal refinement operator*.

Optimal ρ ensures that each pattern will be created and evaluated only once during a run of the algorithm. Since only conditions are added whose upper bound exceeds the threshold, the resulting specializations may have a score that exceeds or matches the current threshold.

The created specializations are then evaluated on the data set and the σ -scores and upper bounds are calculated. If possible, the threshold is raised, additionally all specializations whose upper bound is not larger than the threshold τ are pruned. In Algorithm 2, this threshold is either the best score seen so far or a user-defined threshold, whichever is larger. The algorithm can be easily modified so that k best patterns are found, by using the k th-best score as threshold.

Specializations whose scores match the current threshold are added to the set of solutions S only if the solution set does not already include a generalization having the same stamp point. The rationale behind this is that literals not included in the more general pattern do not change the coverage and therefore do not add information. Finally, the set of promising patterns P is pruned using the threshold and all specializations whose upper bound exceeds τ are added.

4 CG applied to several data mining tasks

In the previous sections we have introduced the *cluster-grouping* task and developed the CG algorithm for solving it. *Cluster-grouping* is typically not a goal in itself but rather – as we shall argue – an important step for building global models. A key contribution of our work is that we show that *cluster-grouping* is a useful component for machine learning and data mining systems tackling a wide variety of tasks such as correlated pattern mining, subgroup discovery, classification, and clustering. Many such systems can be decomposed into **two** main components:

- A *local* pattern mining algorithm to find patterns describing/ predicting the behavior of a subset of the data
- A *control* structure, or “wrapper”, that, depending on the local miner’s result, manipulates the data and/or restarts the local mining process, possibly with a different parameter setting

In this section, we will show that the *cluster-grouping* task and algorithm can be used as the local pattern mining component together with a wrapper for correlated pattern mining, subgroup discovery, classification and clustering. The resulting systems will also be empirically evaluated and compared to state-of-the-art systems. This empirical comparison is meant to provide insight into both the effectiveness and the efficiency of the *cluster-grouping* algorithm for the above mentioned tasks. Whereas the criterion for effectiveness depends on the specific task considered, the efficiency of the algorithms will be measured by the number of patterns evaluated during the search, rather than cpu-time or used memory, because these values are implementation-dependent. We now turn our attention to the different subtasks: correlated pattern mining, subgroup discovery, classification and clustering.

4.1 Correlated Pattern Mining

Problem Description *Correlated pattern mining* [4, 29] is motivated by the observation that association rules with very high confidence may still carry only little information. If every single person shopping in a grocery store bought bread and every second person bought milk then an association rule $milk \Rightarrow bread$ would have a support of 0.5 and a confidence of 1.0 but still be useless. Therefore, using a correlation measure, grounded in statistical principles, rather than frequency will typically result in more interesting relationships. Reformulated,

while classical association rule mining assumes frequent patterns to be interesting, correlated pattern mining looks for local patterns for which the distribution of the target item significantly deviates from the distribution in the entire data set.

Correlated Pattern Mining Using Cluster-Grouping Morishita and Sese [29] model *correlated pattern mining* in the following way – the attribute of interest is restricted to a single, fixed item and the quality of patterns is quantified using the χ^2 -statistic to compare expected and observed occurrence counts. *Correlated pattern mining* can be modeled as a *cluster-grouping* problem with:

- $\mathcal{L} = \{I = \text{true} \mid I \in \mathcal{I}\}$, with $\mathcal{I} = \{I_1, \dots, I_z\}$, the set of items, and $\forall I \in \mathcal{I} : \mathcal{V}[I] = \{\text{false}, \text{true}\}$,
- $\mathcal{E}$ a transaction database,
- σ a (convex) correlation measure such as χ^2 ,
- τ_{user} the user threshold, and
- $\mathcal{A}_t = \{I_0\}$

Inclusion in the actual solution set is based either on whether a pattern belongs among the k best patterns according to the correlation measure used or on a p-value for the measure. This gives it a more sound statistical interpretation than the setting of a support threshold. The “wrapper” for this approach actually only performs a single call to CG with certain parameters.

Algorithm 3 The correlated pattern mining algorithm

$$S = Th_{\infty}(\mathcal{L}, \chi^2, \mathcal{E}, \tau_{user}, I_0)$$

Denoting the local pattern mining step in Algorithm 3 for *correlated pattern mining* the generic notation $Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau_{user}, \{A_1^T, \dots, A_d^T\})$ from Definition 6 is used, where the target attribute has been restricted to the item I_0 . Furthermore, S denotes the set of computed solutions.

Experimental Evaluation Since the patterns mined by Morishita and Sese are special cases of *cluster-grouping* patterns and the pruning technique is based on the same principles, it follows that the CG algorithm is applicable to *correlated pattern mining* and also that it will produce exactly the same solutions as Morishita and Sese’s approach. We therefore include no experiments on this task.

4.2 Subgroup Discovery

Problem Description In *subgroup discovery*, the goal is to find groups of instances in the data that show unexpected behavior with regard to a target

attribute. For instance, a higher than expected frequency of lung cancer, as compared to the overall population, in people living in areas with high air pollution or a lower than expected number of cardiac arrests in persons whose diet is rich in olive oil. Again, *subgroup discovery* can be viewed as an attempt to find patterns for which the distribution of values for a given target attribute deviates from their distribution in a different context (e.g. the entire data set or a particular subset [18]).

Subgroup Discovery using Cluster-Grouping Lavrač *et al.* [23] show how a rule learning algorithm such as *CN2* [8], used together with a function measuring positive correlation such as *Weighted Relative Accuracy* (*WRAcc*) [22], can be employed to find subgroups. The resulting system is CN2-SD, which we will use here as a representative subgroup discovery system. The local pattern mining component of CN2-SD can be modeled as a *cluster-grouping* problem with:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A} \setminus \{A_t\}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *Weighted Relative Accuracy*
- $k = 1$
- $\mathcal{A}_t = \{A_t = v_i\}$

Given that *WRACC* is an asymmetrical measure that rewards higher-than-expected occurrence of a value, and penalizes lower-than-expected occurrence, the target attributes for *subgroup discovery* are derived by turning the actual target attribute into a binary one denoting presence of a value.

The System CN2-SD performs beam search within a “wrapper” that re-weights instances that have already been covered to reduce their importance. The complete mining system is shown in Algorithm 4, where $\tau = -\infty$ implies that the k best patterns, regardless of their score, are included.

The local pattern mining step in CG-SD that computes Th_1 is based on an incomplete search strategy, beam search, which does not guarantee that the k best patterns are found. However, this step can also be performed by CG, we call the resulting algorithm CG-SD.

Experimental Evaluation To compare the complete search method of CG-SD with the heuristic approach of CN2-SD, we set up experiments to answer the following questions:

- Q1 Does CN2-SD find all subgroups found by CG-SD?
- Q2 Is CN2-SD more efficient than CG-SD?

The evaluation is performed in two settings:

- A) without the wrapper, where we search for a single top-scoring subgroup, and

Algorithm 4 The general weighted covering algorithm

```
 $S = \emptyset$ 
 $\forall e_i \in \mathcal{E} : \text{weight}(e_i) = 1$ 
repeat
  Find  $Th_1(\mathcal{L}, WRAcc, \mathcal{E}, -\infty, A_t = v_i)$ 
  for all  $e_i \in E$  do
    if  $e_i$  is covered by  $Th_1$  then
       $\text{weight}(e_i) = \left(\frac{1}{\text{weight}(e_i)} + 1\right)^{-1}$ 
    end if
  end for
  if  $Th_1 \notin S$  then
     $S = S \cup Th_1$ 
  end if
until  $\forall e_i \in \mathcal{E} : \text{weight}(e_i) \leq 1$ 
```

Table 3. Comparison for induction of a single subgroup per class value, setting A. The first column lists the data set, the last columns the number of candidate pattern evaluated by CG-SD, corresponding to 100%, columns 2–4 the corresponding percentage-values for different settings of CN2-SD.

Dataset	CN2-SD ₂₀	CN2-SD ₁₀	CN2-SD ₅	CG-SD
Balance-2-Class	644.00%	436.00%	278.00%	50 (100%)
Breast-W	3443.04%	1791.14%	948.10%	79 (100%)
Breast-W-equal	3061.36%	1609.09%	856.82%	88 (100%)
Car	1722.22%	898.08%	481.61%	261 (100%)
Colic	10569.26%	5336.49%	2723.65%	296 (100%)
Colic-equal	10699.64%	5394.31%	2772.95%	281 (100%)
Credit-G	2106.84%	1062.73%	541.76%	1492 (100%)
Credit-G-equal	2036.56%	1028.06%	523.89%	1436 (100%)
Diabetes	2445.24%	1329.76%	705.95%	84 (100%)
Diabetes-equal	1014.78%	550.25%	291.63%	203 (100%)
Heart-H	3682.01%	1876.19%	976.72%	189 (100%)
Heart-Statlog	2639.30%	1342.36%	696.07%	229 (100%)
Heart-Statlog-equal	2416.27%	1227.38%	637.70%	252 (100%)
Krkopt	1463.92%	765.52%	413.20%	2697 (100%)
Mfeat-Morpho	2090.53%	1244.44%	672.43%	243 (100%)
Mfeat-Morpho-equal	2086.42%	1249.38%	676.95%	243 (100%)
Nursery	3283.92%	1692.60%	888.75%	311 (100%)
Segment	7784.20%	3949.24%	2015.46%	595 (100%)
Tic-Tac-Toe	1717.58%	879.69%	461.33%	256 (100%)
Voting Record	7201.55%	3655.04%	1883.72%	129 (100%)
Zoo	13206.91%	6714.63%	3400.96%	1982 (100%)
Pendigits	5523.76%	2800.83%	1313.24%	846 (100%)
Mushroom	11928.74%	5997.13%	3074.71%	522 (100%)
Average	4155.07%	2119.10%	1090.17%	

B) with the wrapper, where we look for an incrementally constructed set of subgroups.

To answer the questions posed above, we perform experiments on a number of UCI data sets, which were selected such that a large range of data cardinality and dimensionality were covered. The implementation of CG is currently limited to nominal data. Therefore, numerical attributes have been discretized for the experiments, and we only chose data with discrete classes. Two unsupervised discretization approaches were chosen. In the naïve version, the mean value of an attribute is computed and taken as threshold, leading to two nominal values. For some data sets this leads to very unbalanced value distributions. For these sets, we also chose the threshold in such a way that two roughly equally distributed nominal values result. These data sets are denoted by a trailing “-equal” in the name.

Table 4. Comparison of a complete *subgroup discovery* run, setting B. First column lists the data set, last columns the number of candidate pattern evaluated by CG-SD, corresponding to 100%, columns 2-4 the corresponding percentage-values for different settings of CN2-SD.

Dataset	CN2-SD ₂₀	CN2-SD ₁₀	CN2-SD ₅	CG-SD
Balance-2-Class	537.37%	356.48%	214.86%	471 (100%)
Breast-W	1588.41% •	865.23% •	442.74% •	179625 (100%)
Breast-W-equal	1475.91% •	800.42% •	504.28% •	117251 (100%)
Car	689.71%	350.11%	184.15%	61609 (100%)
Colic	1109.36% •	563.09% •	285.68% •	395291 (100%)
Colic-equal	892.98% •	458.43% •	218.20% •	476363 (100%)
Credit-G	231.50% •	113.24% •	55.72% •	543376 (100%)
Credit-G-equal	152.01% •	66.88% •	32.22% •	684859 (100%)
Diabetes	1948.31% •	1061.99% •	486.26% •	8030 (100%)
Diabetes-equal	316.79% •	169.46% •	88.00% •	13836 (100%)
Heart-H	1223.74% •	617.57% •	391.68% •	22415 (100%)
Heart-Statlog	1263.71%	911.69% •	479.43% •	5509 (100%)
Heart-Statlog-equal	1178.30%	595.25%	304.50%	6692 (100%)
Krko	655.85% •	337.64% •	182.00% •	394671 (100%)
Mfeat-Morpho	1724.76%	1017.61%	530.07%	11775 (100%)
Mfeat-Morpho-equal	1465.44%	864.24%	450.13%	12052 (100%)
Nursery	2082.05%	1066.15%	552.82%	975 (100%)
Segment	5610.12%	2835.89%	1410.76%	19293 (100%)
Tic-Tac-Toe	771.75%	391.34%	201.31%	2818 (100%)
Voting Record	1500.65% •	2454.68% •	2540.43% •	3643096 (100%)
Zoo	13206.91%	6714.63%	3400.96%	1982 (100%)
Pendigits	2705.73% •	1443.45% •	733.19% •	279217 (100%)
Mushroom	10002.78%	4870.87% •	2377.40% •	8900 (100%)
Average	2210.15%	1150.94%	588.10%	

• denotes that a non-optimal subgroup, that is a subgroup having a lower score than the highest possible, has been found

Experimental Setting The experimental settings are the following:

- The attribute of interest is the class label
- Beam sizes for CN2-SD are 5,10,20¹
- σ is *WRAcc*

Results Tables 3 and 4 report the number of candidate patterns evaluated by CG-SD – which corresponds to 100% – and the corresponding percentage-values for different settings of CN2-SD. Additionally, the tables specify whether CG-SD found a subgroup description better than the one induced by CN2-SD during one of the iterations.

For setting **A**), the single subgroup case, CG-SD always evaluates far less candidate patterns than the beam search, for all settings of beam size. For this setting, CN2-SD did find the highest-scoring subgroup for each data set.

For setting **B**), the result of the single-subgroup run carries over to the suggested setting of beam size 20, even though the difference is not as pronounced as before. While for some data sets CN2-SD needs less candidate patterns than CG-SD for beam sizes 5 and 10, the heuristic technique also fails to find the top-scoring subgroups for these settings. On average, CG-SD needs 5 to 20 times less candidate evaluations, and this factor correlates strongly with the used beam-size. Note, finally, that even a beam size of 20 does not ensure that all highest-scoring subgroups are found by CN2-SD!

As a consequence, the answers to **Q1** and **Q2** are negative for both settings **A**) and **B**), and CN2-SD is neither as effective as CG-SD, nor more efficient. Therefore, branch-and-bound *cluster-grouping* should be preferred over beam-search for subgroup discovery.

4.3 Finding Rules for Classification

Problem Description *Classification* is related to *subgroup discovery*, because rules for classification concern groups whose class distribution differs from the default one. Once rules describing such groups are found, they can be used to predict the value of the class attribute. The main difference with *subgroup discovery* is that rule-based *classification* aims at inducing a set of rules that, taken together, correctly predict the *entire* set of training instances.

Classification and Cluster-Grouping: Since the goal of classification can be re-interpreted as finding rules that separate two classes from one another, measures such as χ^2 and *Information Gain* can be used as well as – to a certain degree – accuracy to solve the task in the *cluster-grouping* framework. The problem then has the following characteristics:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A} \setminus \{A_t\}, v \in \mathcal{V}[A]\}$

¹ 20 was suggested by a reviewer, 5 and 10 evaluated as well as to not bias the efficiency estimation against CN2-SD

- $\mathcal{E}$ a data set
- σ is *accuracy*, χ^2 , *Information Gain* or *Category Utility*
- $k = 1, \tau_{user}$
- $\mathcal{A}_t = \{C\}$, where C is the class attribute

Note, that when accuracy is used, its asymmetry would require the same binarization of the target attribute as for the *subgroup discovery* setting.

The System Rule-based classifiers often rely on the covering paradigm [19]. Our focus lies essentially on the sequential covering paradigm in which patterns are mined, covered instances removed, and the process iterated on the remainder of the data set. Mining can either be done in a greedy way, e.g. optimizing some measure’s score using beam search [8], or exhaustively, by setting thresholds on e.g. the support and confidence of interesting patterns and performing the covering step as post-processing [25, 24].

This gives rise to two possible “wrappers”, which are shown in Algorithms 5 and 6: sequential covering and complete mining.

Algorithm 5 The sequential covering algorithm.

```

 $S = \emptyset$ 
repeat
   $\mathcal{E} = \mathcal{E} \setminus \{e | e \text{ covered by } Th_1(\mathcal{L}, \sigma, \mathcal{E}, \tau_{user}, C = c_1)\}$ 
   $S = S \cup Th_1$ 
until  $\mathcal{E} = \emptyset \vee Th_1 = \emptyset$ 
return  $S$ 

```

Algorithm 6 The complete mining algorithm.

```

 $S = \text{post-process}(Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau_{user}, C = c_1))$ 

```

By instantiating the inner loop of the sequential covering algorithm (Algorithm 5) with CG we derive CN2-CG, by using a beam search maximizing χ^2 , CN2 $_{\chi^2}$. We empirically compare those two techniques below, also to RIPPER [10], one of the most sophisticated sequential covering algorithms that use beam search.

For the complete mining algorithm, sketched in Algorithm 6, one choice is to use *minimum frequency* to estimate significance; the resulting system has been introduced as CBA [25] – *classification based on association*. Alternatively, by using CG instead for mining the significant patterns, we obtain the novel CBC – *classification based on correlation* algorithm, which we empirically evaluate below.

Experimental Evaluation As demonstrated in the section on *subgroup discovery*, beam size is important for both the quality of found solutions, and the efficiency of the mining technique, when comparing $CN2_{\chi^2}$ to CN2-CG. At the same time, we are also interested in comparing the performance to RIPPER. This yields to the following reformulations of Q1 and Q2 for the sequential covering approaches:

Q1' How does the quality of the rules found by $CN2_{\chi^2}$, CN2-CG and RIPPER compare?

Q2' Is $CN2_{\chi^2}$ more efficient than CN2-CG?

However, for measuring the quality of the discovered solutions in the classification setting, we employ classification accuracy rather than the correlation measures used for *subgroup discovery*. We are also interested in a comparison to the state-of-the-art system for classification RIPPER.

For the comparison of the complete mining algorithms, the following questions result:

Q3 How does the quality of CBA's classifiers compare to those of CBC?

Q4 Is CBA more efficient than CBC?

Experimental setup The experimental setting for the sequential covering approach are as follows:

- Beam sizes for $CN2_{\chi^2}$ are 5, 10, 20.
- Minimum significance threshold for $CN2_{\chi^2}$ and CN2-CG is 3.84.
- RIPPER is run as WEKA's [17] JRIP implementation with default parameters and pruned classifiers evaluated.
- $CN2_{\chi^2}$ - and CN2-CG-classifiers are unpruned.

For the exhaustive techniques, the following parameter settings are used:

- WEKA's APRIORI implementation is used in CBA, with 1% minimum support and 50% minimum confidence.
- Minimum significance threshold of CBC is 3.84.
- Maximum number of mined rules is $50,000^2$
- CBC mines only the 1000 most accurate rules, a restriction motivated by an observation in [31], that the rules chosen for the final classifier fall well within the 1000 highest-ranked rules.

The data sets used were again discretized. The discretization scheme was more sophisticated than in the *subgroup discovery* experiments, using Fayyad and Irani's supervised discretization method [12]. The discretization algorithm was run on the training folds, with the resulting intervals used on test data, such as not to introduce bias into the data.

Table 5. Average accuracy and standard deviation for CN2 $_{\chi^2}$, CN2-SD, RIPPER, CBA, and CBC. The left-most column lists data sets, columns 2-4 accuracy estimates for sequential covering approaches (2 & 3 annotated with statistical t-test comparison to RIPPER), CN2 $_{\chi^2}$ results are also annotated with the width of the beam giving rise to the result, columns 5 & 6 list exhaustive techniques (column 6 annotated with t-test comparison to CBA).

Dataset	CN2 $_{\chi^2}$	CN2-CG	RIPPER	CBA	CBC
Balance (2 Class)	86.8 $\pm$ 3.9 (5) $\circ$	86.8 $\pm$ 3.9 $\circ$	80 $\pm$ 3.4	79.18 $\pm$ 4.59	79.18 $\pm$ 4.59
Breast-Cancer	81.5 $\pm$ 8.4 (10)	80.4 $\pm$ 0.77	71.7 $\pm$ 0.72	68.19 $\pm$ 8.48	66.77 $\pm$ 9.28
Breast-W	96.4 $\pm$ 2.4 (5)	96.4 $\pm$ 2.4	95.7 $\pm$ 2.1	94.71 $\pm$ 1.9	95.71 $\pm$ 1.34
Colic	82.9 $\pm$ 5.5 (5)	88.9 $\pm$ 5.9	83.9 $\pm$ 7.7	81.27 $\pm$ 8.07	76.91 $\pm$ 6.51
Credit-A	86.5 $\pm$ 2.5 (20)	85.8 $\pm$ 2.1	85.4 $\pm$ 2.5	85.65 $\pm$ 4.35	84.06 $\pm$ 4.48
Credit-G	79.4 $\pm$ 6 (10) $\circ$	79.4 $\pm$ 6 $\circ$	69.4 $\pm$ 5.4	71.4 $\pm$ 2.63	69.8 $\pm$ 4.89
Diabetes	77.4 $\pm$ 5.4 (10)	75.1 $\pm$ 6.2	76 $\pm$ 3.9	75.92 $\pm$ 4.14	75.78 $\pm$ 4.23
Heart-H	83.3 $\pm$ 7.5 (5)	81.6 $\pm$ 6.3	79.2 $\pm$ 7.4	83.33 $\pm$ 6.69	82.66 $\pm$ 5.82
Kr-vs-Kp	94.3 $\pm$ 1.4 (5) $\bullet$	94.3 $\pm$ 1.4 $\bullet$	99.3 $\pm$ 0.4	80.72 $\pm$ 1.75	95.63 $\pm$ 1.29 $\circ$
Mushroom	98.5 $\pm$ 0.3 (5) $\bullet$	98.5 $\pm$ 0.3 $\bullet$	1	99.53 $\pm$ 0.19	1 $\circ$
Spambase	91.4 $\pm$ 1.4 (10)	89 $\pm$ 1.4 $\bullet$	92.7 $\pm$ 1.1	86.39 $\pm$ 1.62	86.09 $\pm$ 1.61
Tic-Tac-Toe	84.6 $\pm$ 2.2 (5) $\bullet$	83.1 $\pm$ 2.2 $\bullet$	97.1 $\pm$ 1.2	1	1
Voting Record	95.3 $\pm$ 3.2 (5)	96.2 $\pm$ 3	95.6 $\pm$ 2.8	94.25 $\pm$ 3.1	93.1 $\pm$ 3.58

$\circ$ denotes statistical wins at the 99% level, base-line being RIPPER, and CBA, respectively

$\bullet$ denotes statistical losses at the 99% level, base-line being RIPPER, and CBA, respectively

Results CN2 $_{\chi^2}$, CN2-CG, and RIPPER give rise to solutions of similar quality, cf. Table 5, with RIPPER being significantly better than CN2 $_{\chi^2}$ three times (four times *vs* CN2-CG), χ^2 -based optimization being significantly better twice. With CN2 $_{\chi^2}$ outperforming CN2-CG once, one can conclude w.r.t. **Q1'** that the quality of the heuristically derived classifiers is at least equal to the ones found using CN2-CG. It should be noted however that selecting the right beam size is non-trivial, mirroring the results of the *subgroup discovery* experiments.

Two of the cases in which RIPPER finds the better solution are large data sets with rules that have high accuracy on only small subsets (*Kr-vs-Kp*, *Tic-Tac-Toe*). This indicates that significance measures at some point tend to penalize low-frequency rules too much. On the other hand, the χ^2 -based approaches significantly outperform RIPPER on the *Balance* and *Breast-Cancer* data sets, both of which are rather small, more likely leading to overfitted rules, which are discounted by the significance estimate of χ^2 . In contrast, all data sets on which RIPPER performs well are large - giving a large enough sample to counteract the overfitting stemming from accuracy maximization. Thus there seems to be a slight advantage provided from the sophisticated pruning techniques of RIPPER which has less of an effect on small data sets though.

² An exception is the *Kr-vs-KP* data set where 90,000 rules are needed for CBA to find rules with confidence of at least 90%

The comparison of CBC with CBA shows that using χ^2 rather than support for quantifying the significance of rules (and ranking them thus) gives better results in that CBC never performs significantly worse than CBA and in two cases is significantly better, answering **Q3** positively regarding CBC effectiveness. In the case of the *Mushroom* data set the ordering of the rule set before pruning is decidedly different between the two approaches and thus different rules are selected for the final classifier. In the *Kr-vs-Kp* scenario, limiting the mining process to the 90,000 most significant rules according to support excludes many high-confidence rules. Even at 200,000, the highest confidence is at just 0.92, while for CBC rules with confidence 1.0 are found within the 50,000 most significant rules according to χ^2 .

Table 6. Average number of patterns mined by the CN2 $_{\chi^2}$, and CN2-CG, number of patterns mined and used by RIPPER

Dataset	CN2 $_{\chi^2}$	CN2-CG	RIPPER	
	# mined	# mined	# mined	# used
Balance (2 Class)	6 $\pm$ 1	5.4 $\pm$ 2.12	8.7 $\pm$ 2.35	5.2 $\pm$ 1.22
Breast-Cancer	27.1 $\pm$ 6.3	24.6 $\pm$ 6.1	11.2 $\pm$ 3.97	3.1 $\pm$ 0.74
Breast-W	12.8 $\pm$ 1.3	11.4 $\pm$ 0.8	19.8 $\pm$ 2.04	6.6 $\pm$ 0.97
Colic	16.5 $\pm$ 2.9	8 $\pm$ 0.9	12.9 $\pm$ 2.23	3.6 $\pm$ 0.7
Credit-A	16.2 $\pm$ 2.4	14.6 $\pm$ 1.84	25.5 $\pm$ 2.42	5.8 $\pm$ 1.81
Credit-G	31.1 $\pm$ 6.1	30 $\pm$ 3.37	15.3 $\pm$ 2.79	5.5 $\pm$ 2.12
Diabetes	10 $\pm$ 2.3	11.6 $\pm$ 3.37	20.1 $\pm$ 3.14	5.2 $\pm$ 0.91
Heart-H	9 $\pm$ 2	8.8 $\pm$ 1.75	10.8 $\pm$ 1.39	3.5 $\pm$ 0.85
Kr-vs-Kp	3	2	19.2 $\pm$ 1.13	15.4 $\pm$ 1.27
Mushroom	4.3 $\pm$ 0.5	3	8.8 $\pm$ 0.79	8.7 $\pm$ 0.68
Spambase	10.6 $\pm$ 1	5.8 $\pm$ 0.92	53.9 $\pm$ 3.28	27.3 $\pm$ 3.23
Tic-Tac-Toe	6.3 $\pm$ 1.3	8.8 $\pm$ 0.4	12.1 $\pm$ 2.02	10.6 $\pm$ 1.27
Voting Record	4.8 $\pm$ 0.6	3.2 $\pm$ 0.42	8.8 $\pm$ 0.63	2.9 $\pm$ 1.2

Since the quality of found rules for the heuristic (CN2 $_{\chi^2}$) and complete (CN2-CG) χ^2 maximization is very similar, **Q2'** focusses on whether one of the two techniques is more efficient. Note that the introduction of beam search mainly attempts to make the search space manageable by focusing on certain subspaces. Upper bound pruning on the other hand, uses a different kind of restriction, also with the aim of focussing on the relevant parts of the search space. Table 6 shows that CN2-CG often though not always selects fewer rules than CN2 $_{\chi^2}$, apparently capturing the underlying regularities better.

The number of candidate patterns evaluated, shown in Table 7, does not give a clear answer to question **Q2'**. On several occasions CN2-CG is better, most pronounced for the *Colic* data set, while e.g. on *Credit-G* CN2 $_{\chi^2}$ finds effective rules far quicker. A possible explanation is that effective rules are found early, the upper bound is not tight enough though, thus exploring large parts of the search space without gaining anything.

Table 7. Number of candidate patterns evaluated by CN2_{χ²} (for beam size giving the best solution) and CN2-CG. Column three lists number of pattern evaluated by CN2-CG, equating 100%, column two the corresponding percentage value for CN2_{χ²}

Dataset	CN2 _{χ²}	CN2-CG
Balance (2 Class)	139.83%	261.60 (100%)
Breast-Cancer	108.00%	46153.30 (100%)
Breast-W	101.69%	6817.00 (100%)
Colic	2277.10%	2288.70 (100%)
Credit-A	10.76%	1003999.50 (100%)
Credit-G	1.35%	17061266.50 (100%)
Diabetes	40.60%	13030.90 (100%)
Heart-H	36.27%	17809.40 (100%)
Kr-vs-Kp	5.46%	240431.00 (100%)
Mushroom	90.66%	28758.60 (100%)
Spambase	16.14%	2828763.30 (100%)
Tic-Tac-Toe	102.56%	2975.40 (100%)
VotingRecord	39.17%	11726.50 (100%)
Average	228.43%	

The final question to be answered is **Q4**, namely whether CBA is more efficient than CBC. Table 8 shows again no clear-cut advantage for either technique. On average CBA mines slightly fewer patterns than CBC .

Again, large data sets on which accurate rules have small coverage, and data sets with minority classes make upper-bound pruning less effective. More specifically, basing *associative classification* mining on CG compares worst on *Kr-vs-Kp*, *Spambase*, and *Tic-Tac-Toe*. We have seen however that *Kr-vs-Kp* gives also CBA trouble and subsequent experiments in which the number of mined patterns is set to 1.8 million still does not give classifiers comparing well with the CG-solution while exceeding its number of evaluated candidate patterns significantly.

To summarize, the experiments show that using statistically well-founded measures improves the prediction accuracy of heuristic methods on small data sets and generally improves upon the accuracy of frequency-based *associative classification* methods. While the efficiency of CG is on average as good as or better than the alternative approaches¹, for particular data sets existing methods can outperform CG. These findings suggest 1) that the robustness of sequential covering algorithms such as RIPPER or CN2 that use beam search, a heuristic technique, may be improved improved by using CG 2) that it may be advantageous to replace the use of support and confidence in *associative classification* techniques such as CBA [25] and CMAR [24] by using correlation measures grounded in statistical theory, 3) that also decision tree approaches, who choose an optimal pattern based on a single attribute, may profit from using *cluster-grouping* instead, cf. also our approach to clustering below and the Tree² approach of [5].

Table 8. Number of candidate pattern evaluated by the complete mining algorithms The last column lists number of patterns for CBC, equating 100%, column two shows the corresponding percentage value for CBA

Dataset	CBA	CBC
Balance (2 Class)	156.01%	99.80 (100%)
Breast-Cancer	127.44%	8179.20 (100%)
Breast-W	52.58%	12913.80 (100%)
Colic	136.29%	73682.90 (100%)
Credit-A	98.49%	65226.50 (100%)
Credit-G	39.14%	155020.90 (100%)
Diabetes	126.52%	3875.80 (100%)
Heart-H	172.57%	14680.00 (100%)
Kr-vs-Kp	15.92%	687715.50 (100%)
Mushroom	95.86%	53615.80 (100%)
Spambase	15.13%	445856.30 (100%)
Tic-Tac-Toe	38.14%	24511.10 (100%)
Voting Record	65.41%	88996.10 (100%)
Average	87.65%	

Related Work To the best of our knowledge, all rule-based approaches to classification mine local patterns in some way and build classifiers from them. Decision trees solve the problem of finding the optimal splitting pattern by essentially limiting the number of conditions to one. There has been work on multi-variate splitting criteria [30] - there, similar decisions on the induction mechanism have to be made as in sequential covering. Sequential covering algorithms like RIPPER or CN2 use beam search, a heuristic technique, inside the covering loop and their robustness could be improved by using CG. *Associative classification* techniques such as CBA [25] and CMAR [24] mine patterns based on user-specified values for support and confidence. Both approaches rely on the declaration of parameters by the user. The exhaustive algorithms have parameters which are rather difficult to decide upon but which can have an important effect on the resulting set [9]. Additionally, the result set often is made up of a very large number of rules, making interpretation by the user difficult. Basing the choice of cut-off value on statistical theory and pushing the significance test inside the mining step should improve both efficiency and effectiveness of such techniques.

4.4 Conceptual Clustering

Problem Description In *clustering*, the goal is to partition the instances of a data set into usually disjoint subsets (clusters) that exhibit high *intra-cluster similarity* and high *inter-cluster dissimilarity*. For the numerical case clusters can be represented for instance by centroids or medoids and the similarity quantified by a vector norm such as the L1 or L2 norm.

The goal in *conceptual clustering* is to perform the *clustering* task for instances described by nominal attributes for which a similarity measure is often

harder to define. In general, instances are considered similar if they agree on the values of many attributes. One measure for judging the quality of a set of clusters is *Category Utility* [20] but others have been defined in the literature as well. An additional goal is to produce a description of the found clusters in terms of the conceptual language used to describe the instances.

Clustering and Cluster-Grouping Clusters are often arranged into a hierarchy, with clusters closer to the root of the clustering tree (or *dendogram*) described by more general concepts. Such a dendogram can be obtained using either a divisive or an agglomerative approach. Here, we focus on a divisive approach, which bears some similarities to decision tree induction, in which clusters are repeatedly divided into subclusters according to some criterion. According to Hoepner et al. [21] clusters can be considered as deviations in distribution from a default (or background) distribution w.r.t. certain attributes. Therefore, *cluster-grouping* can be used to identify patterns that capture the deviating areas and can be used to split the clusters. Using CG assures the best split without restarts of the clustering algorithm and allows the induction of conjunctive descriptions for clusters. Maximizing e.g. *Category Utility* – with binary attributes only – is a *cluster-grouping* task with the following characteristics:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A}, v \in \mathcal{V}[A]\}$, where $\mathcal{A} = \{A_1, \dots, A_d\}, \forall A \in \mathcal{A} : \mathcal{V}[A] = \{true, false\}$
- $\mathcal{E}$ a data set
- σ is *Category Utility*
- $k = 1$
- $\mathcal{A}_t = \{A_i \mid A_i \in \mathcal{A}\}$

Since the goal, as mentioned above, is similarity in as many attributes as possible, *all* attributes are considered targets, with the symmetric measure *CU* leading to the induction of patterns in whose coverage space either the occurrence of *true* or false values will be higher than expected.

The System A dendogram is a decision tree-like structure, with cluster membership decided by the concepts in the nodes. Therefore the wrapper for a CG-based clustering algorithm is a bit more involved, compared to the other wrappers, mirroring recursive decision tree algorithms. Algorithm 8, which we term CG-CLUS, thus calls a function SPLIT that recursively constructs a tree, whose inner nodes denote patterns (or their negations). This is to the best of our knowledge the first time that correlation based patterns are used in divisive clustering approach.

A second approach is *cluster mining* [33], where a clustering algorithm is used to find a clustering, each cluster treated as a single class, and conjunctive concepts learned on them. Afterwards, all instances matching a concept are considered to be in one cluster, possibly producing overlapping clusters.

Algorithm 7 SPLIT

Input: Data set $\mathcal{E}$, Leaf node t
if $Th_1(\mathcal{L}, \text{CU}, \mathcal{E}, -\infty) \neq \emptyset$ **then**
 $\mathcal{E}_1 = \mathcal{E} \setminus \{e | e \text{ covered by } Th_1(\mathcal{L}, \text{CU}, \mathcal{E}, -\infty)\}$
 $\mathcal{E}_2 = \mathcal{E} \setminus \mathcal{E}_1$
 Create left child of t , t_l , containing $Th_1(\mathcal{L}, \text{CU}, \mathcal{E}, -\infty)$
 Create right child of t , t_r , containing $\neg Th_1(\mathcal{L}, \text{CU}, \mathcal{E}, -\infty)$
 SPLIT($\mathcal{E}_1, t_l$)
 SPLIT($\mathcal{E}_2, t_r$)
end if

Algorithm 8 CG-CLUS

$T = \emptyset$
SPLIT($\mathcal{E}, T$)
return T

To evaluate CG-CLUS, we shall compare it to COBWEB [13], the arguably best-known conceptual clustering technique, and a cluster mining technique using AUTOCLASS [7] and RIPPER. COBWEB iteratively processes instances, using four operators: assigning an instance to an existing dendogram node, creating a new node, splitting an existing node, or merging two existing nodes.

Experimental Evaluation COBWEB’s direct assignment and iterative processing gives it great flexibility in assembling clusters but also makes it vulnerable to ordering effects in data. In addition, by using conditional probability vectors instead of conjunctions to describe the clusters, it has fewer restrictions which instances to cluster together. Thus, a question pertaining to COBWEB is:

Q5 Do COBWEB’s clusterings have higher CU than the ones of CG-CLUS? This question is meant to provide an insight into the effectiveness of CG-CLUS.

AUTOCLASS is based on Bayesian principles and thus not directly optimizing CU. On the other hand, it has greater flexibility than CG-CLUS in assigning instances *directly* to clusters – not indirectly via the found description. In addition, decoupling the processes of forming clusters and finding a description gives the actual concept formation greater flexibility than CG-CLUS possesses. Since both COBWEB and a *cluster-mining* approach have greater flexibility (and make conjunctive concept formation a non-integral part of the mining process), two further questions are:

Q6 How similar are CG-CLUS’ and COBWEB’s/AUTOCLASS’ clusterings?

Q7 How complex are conjunctive descriptions of COBWEB’s/AUTOCLASS’ clusters, compared to CG-CLUS’ ones, and how much information about the underlying instances is recovered?

Experimental setup To compare the agreement of two clusterings, we use the *Rand* index. It is the fraction of pairwise grouping decisions on which the two clusterings agree. Let $\mathcal{E} = \{e_1, \dots, e_n\}$ be a data set and $\mathcal{C}_1, \mathcal{C}_2$ two clusterings

of $\mathcal{E}$. For each pair of instances e_i, e_j , $\mathcal{C}_l$ either assigns them to the same cluster or to different clusters. Let *pos* be the number of decision where e_i, e_j are in the same cluster in both clusterings and *neg* the number of decisions where they belong to different clusters in both $\mathcal{C}_l$. The *Rand Index* is defined as:

$$Rand(\mathcal{C}_1, \mathcal{C}_2) = \frac{pos + neg}{n * (n - 1) / 2}$$

If the number of clusters in a clustering is different, the *Rand* index will obviously show a dissimilarity. Therefore we attempted to form a number of clusters corresponding to the number of class values on each data set in our experiments.

To obtain a given number of clusters from a COBWEB dendrogram, there are two possibilities – the user selects certain nodes in the tree, disregarding the structure underneath them, or the growth of the dendrogram is limited.

In the first case, the fact that COBWEB often constructs dendograms in which every instance is sorted into its own cluster, makes this a non-trivial procedure. Also, selecting all nodes from the same level of the tree does not guarantee a good solution *CU*-wise.

Instead, in the WEKA implementation, a minimum *CU*-gain can be set which determines whether new nodes in the dendrogram are introduced, or existing nodes split. By starting out with a lenient threshold, systematically tightening it when more than the desired number of clusters is formed and relaxing it when the tightening proved to be too strict, it is possible to approximate the desired number of clusters.

Unfortunately, this method does not always guarantee obtaining the target number of clusters, since COBWEB sometimes forms just one cluster or tens/hundreds depending on a 0.0001 difference in the threshold value. Instead of arbitrarily merging clusters, we used the COBWEB-solution whose number of clusters is closest to the actual number of classes, unless this number is 1, i.e. all instances were sorted into the same cluster. After determining the COBWEB-clustering, we attempted to construct the same number of clusters using CG-CLUS.

AUTOCLASS can be supplied with the number of clusters it should create. For each data set, AUTOCLASS performed 250 restarts with 200 iterations each. The assumed model was single multinomial for all attributes. We used the best clustering found for comparison with the CG-based approach.

As for our technique, CG-CLUS, to obtain the desired number k of clusters, the $k - 1$ best splits are used. Since a good *CU* score on a small subset is easier to achieve than on a larger one, patterns' scores are weighted with the proportion of instances of the complete set that they were derived on. The resulting dendrogram is decision tree-like in the way data is split on patterns and their impact discounted on the population size.

Owing to the need for binary attributes, discretization was performed as in the *subgroup discovery* experiments, and nominal attributes binarized.

Results To answer **Q5** and **Q6**, we report CU -values and the Rand-index for CG-CLUS- and COBWEB-clusterings for a variety of data sets in Table 9. For the data sets for which hundreds or even thousands of clusters were formed by COBWEB we did **not** attempt to form the same number of clusters using CG. Instead we report on the *Category Utility* of the “correct” solution of CG-CLUS (i.e. the clustering having as clusters the classes in the data), the average CU for COBWEB and no value for the *Rand*-index.

Table 9. CU of the CG-CLUS clusterings, and CU of COBWEB’s solution, averaged over 10 runs, Rand-index of the two clusterings

Dataset	CU_{CG}	CU_{CW}	Rand
Credit-G	0.4408	0.0239 ± 0.002	N/A
Credit-G-Equal	0.4753	0.1161 ± 0.0293	N/A
Kr-vs-Kp	0.5343 ± 0.0040	0.5782 ± 0.0012	0.7817 ± 0.0029
KrkOpt	0.1536 ± 0.0072	0.1369 ± 0.002	0.7396 ± 0.028
Letter	0.1742 ± 0.0075	0.1342 ± 0.0151	0.7629 ± 0.0275
Letter-Equal	0.1677 ± 0.0053	0.1439 ± 0.0063	0.8759 ± 0.0001
Mfeat-Fourier	0.4743	0.4487 ± 0.1334	N/A
Mfeat-Fourier-Equal	0.7183	0.1855 ± 0.0289	N/A
Mfeat-Karhunen	0.457	0.0203	N/A
Nursery	0.3555	0.0846 ± 0.0273	N/A
Optdigits	0.4609 ± 0.0029	0.5234 ± 0.0229	0.7865 ± 0.0148
Optdigits-Equal	0.6865 ± 0.0203	0.7936 ± 0.0565	0.8509 ± 0.0069
Pendigits	0.4336 ± 0.0091	0.4015 ± 0.0074	0.8519 ± 0.0004
Segment	0.5878 ± 0.0122	0.5438 ± 0.0505	0.7994 ± 0.1029
Segment-Equal	0.7925 ± 0.0083	0.7916 ± 0.0063	0.8984 ± 0.0039
Waveform	0.9791	1.1624 ± 0.0229	0.7822 ± 0.0192

The resulting *Category Utilities* show that far from always giving rise to superior scores by using the more flexible clustering scheme, the quality of COBWEB’s solution is clearly affected by ordering effects in the data. If the right ordering of instances exists, COBWEB constructs very good solutions, if not, CU values are rather low or the dendrogram is huge. When threshold differences of 0.0001 make the difference between a single cluster and a dendrogram having hundreds of leaves it is difficult for the user to make an informed decision on which clusters to merge. For the data sets where a reasonable number of clusters was constructed, COBWEB’s average CU is larger than that of CG-CLUS four times, less six times, while at the same time exhibiting similarities of the clusterings in excess of 0.7. This means that **Q5** has to be answered negatively, COBWEB does not always translate the greater flexibility of its assignment mechanism into better CU values. Also, the solutions are rather similar, giving the answer to **Q6**.

Table 10 is used to give insight into question **Q7**. It lists the number of classes in the data, the average number of clusters in COBWEB’s clusterings over ten runs, the number of rules learned by RIPPER (unpruned) on these clusters and their accuracy. It should be noted that CG-CLUS builds a tree of conjunctive

descriptions (and their negations) for clusters – which have 100% accuracy – and can easily be constrained to form as many clusters as classes exist in the data.

Table 10. Classes per data set, average number of clusters formed by COBWEB, number of conjunctive rules learned on the clustering using RIPPER, recovery rate (that is training set accuracy of learned rules)

Data sets	# of Classes	# of Clusters	# of Rules	Recovery rate
Credit-G	2	349.5 ± 180.96	38.9 ± 8.86	$65.44\% \pm 17.61$
Credit-G-Equal	2	202.5 ± 191.02	42.7 ± 8.3	$78.34\% \pm 18.8$
Kr-vs-Kp	2	2.5 ± 0.7	15.7 ± 14.47	$99.74\% \pm 0.3$
KrkOpt	18	13.8 ± 5.2	14.5 ± 5.98	$99.99\% \pm 0.003$
Letter	26	18.7 ± 13.01	139.6 ± 40.65	$98.71\% \pm 0.64$
Letter-Equal	26	24.5 ± 5.23	211.2 ± 24.04	$97.41\% \pm 0.61$
Mfeat-Fourier	10	54.9 ± 60.58	51.9 ± 24.82	$95.13\% \pm 3.84$
Mfeat-Fourier-Equal	10	58 ± 24.35	24.6 ± 2.87	$96.99\% \pm 1.2$
Mfeat-Karhunen	10	663.7 ± 173.25	86.3 ± 19.59	$64.28\% \pm 8.75$
Nursery	5	1480.8 ± 958.07	1102.3 ± 710.23	$97.22\% \pm 0.92$
Opdigits	10	8.5 ± 1.65	103.4 ± 16.59	$93.33\% \pm 1.81$
Opdigits-Equal	10	8.6 ± 1.17	111.2 ± 17.21	$93.47\% \pm 1$
Pendigits	10	7.2 ± 0.63	60.5 ± 7.01	$99.73\% \pm 0.08$
Segment	7	4.5 ± 0.7	6.6 ± 1.17	$99.99\% \pm 0.01$
Segment-Equal	7	6.2 ± 0.42	38.9 ± 3.24	$99.42\% \pm 0.13$
Waveform	3	3	52.4 ± 2.75	$97.31\% \pm 1.04$

The experiments show that COBWEB hardly forms a number of clusters that corresponds to the number of underlying classes. In addition, most of the time far more rules than classes will be learned on the data, which do not always capture the clusters very well. So the conjunctive descriptions found using COBWEB’s clusterings are at the same time rather complex and not always reliable.

Regarding the cluster mining solution using AUTOCLASS and RIPPER, Table 11 lists the number of classes per data set (both CG-CLUS and AUTOCLASS form the same number of clusters), the number of rules learned by RIPPER and their accuracy on AUTOCLASS’ solution, as well as the *Rand*-index of the two clusterings.

The similarity of the clusterings produced by the two methods is generally very high, with three exceptions, thus answering **Q6**. While AUTOCLASS’ solutions give rise to smaller descriptions than COBWEB’s do, and AUTOCLASS’ rules achieve a far higher accuracy on the underlying clusters, they still exceed the number of clusters (and thus conjunctive descriptions in CG-CLUS’ tree) by far. This means that while being rather close in actual composition, the description of AUTOCLASS’ solution is far more complex.

To summarize, while being less flexible in forming clusters, the novel system CG-CLUS finds clusterings that are highly similar to the solutions of two more flexible schemes that are well-established in the literature. The mining process itself guarantees high intra-cluster similarity in a single run of the algorithm and in addition, the formulation of conjunctive cluster descriptions of low complexity.

Table 11. Classes per data set, number of descriptive rules found by RIPPER on the AUTOCLASS-solution and their accuracy, and similarity of found clusterings

Data sets	# of Classes	# of Rules	Recovery rate	Rand
Credit-G	2	7	100%	0.5012
Credit-G-Equal	2	25%	99.1	0.558
Kr-vs-Kp	2	2	100%	0.9185
KrkOpt	18	31	100%	0.9663
Letter	26	268	98.86%	0.9016
Letter-Equal	26	277	97.16%	0.9402
Mfeat-Fourier	10	92	99.15%	0.8673
Mfeat-Fourier-Equal	10	64	99.4%	0.8481
Mfeat-Karhunen	10	89	99%	0.8729
Nursery	5	5	100%	0.6543
Optdigits	10	117	95.53%	0.8656
Optdigits-Equal	10	115	96.89%	0.8887
Pendigits	10	73	99.66%	0.9019
Segment	7	9	99.91%	0.8345
Segment-Equal	7	10	99.65%	0.9002
Waveform	3	50	98.2%	0.7877

Related work The field of *conceptual clustering* is too vast to exhaustively discuss everything relating to our work so we will restrict ourselves mainly to the papers mentioned in Section 4.4. One of the earliest approaches to inducing conjunctive descriptions of conceptual clusters is the CLUSTER/2 system [27]. While CLUSTER/2 works bottom-up in a heuristic manner, our technique induces concepts top-down and gives guarantees w.r.t. the quality of found solutions. The *conceptual cluster mining* task, introduced by Perkowitz *et al.* [33], is similar to *cluster-grouping* w.r.t. to clustering. Their goal is to induction clusters that are cohesive but also describable by a simple concept. To this end, they use their PAGEGATHER system for clustering webpages and RIPPER to learn the concept separating each cluster from all others. The final solution consists of all subsets of instances that correspond to the learned concepts. Since both the clustering and the rule learning algorithm could be instantiated differently, the *conceptual cluster mining* framework is rather general. The approach does have potential drawbacks as discussed in Section 4.4. In [32], an incremental branch-and-bound clusterer for the formation of hierarchies was introduced. Since addition of new observations can have a severe effect on the existing hierarchy, re-insertion of instances and clusters is performed during the formation process. To restrict the number of evaluation steps needed, the set of nodes that could act as parents in the hierarchy to the instance or cluster to be inserted is limited. To this end, an upper bound on the best value a node can give is calculated based on the evaluation of picking this node’s parents as parents of the new instance or cluster. The measure we used for the quality of cluster descriptions in this work is *Category Utility*, with the probably best-known clusterer using this measure being COBWEB. Due to its incremental instance processing, the ordering of instances has an effect on the solution. To address this effect, Fisher [14] explores

several re-distribution and re-clustering techniques for greedily improving an existing clustering. We find the optimally discriminating patterns in the first run instead. A second issue addressed in Fisher’s work is related to the effect we observed in Section 4.4, namely that COBWEB on certain data sets tends to create a large amount of clusters, gaining only a small increase in *Category Utility*. The solution discussed in [14] is similar to post-pruning in decision tree learning in that certain branches of the clustering hierarchy are removed during validation on a separate data set. Third, Fisher discusses possible shortcomings of *Category Utility* as a quality measure for clusterings. Assuming that a clustering is used for classification afterwards, he suggests properties such as number of leaves, maximum path length, branching factor and classification cost, e.g. number of attributes to be evaluated, for measuring the quality of a clustering tree. All these parameters could be affected by a user, given a suitable wrapper around CG. Finally, possible alternatives to *Category Utility* mentioned in this work could be used in CG if they are convex.

5 Related Work

Work that is related to the overall *cluster-grouping*-framework can be roughly grouped into two categories: work exploring the relation between local pattern mining and diverse data mining or machine learning tasks, and algorithmically related techniques.

5.1 Local pattern mining for machine learning

The task of correlating pattern mining has been introduced by Brin *et al.* [4]. Their solution to the problem is somewhat different than Morishita and Sese’s one in that they mine for all pattern for whom *all* pairs of items correlate. By restricting their definition in that way, they can derive an anti-monotone criterion that can be used for pruning. It also means that they have more flexibility w.r.t. possible rules. The caveat is, however, that there might be correlations which will only emerge if a combination of items is related to a target item. Those will not be found by their technique.

The relation between local pattern mining and classical machine learning tasks has been explored in recent years, notably at the 2004 Dagstuhlseminar: Detecting Local Patterns [28]. Höppner [21] discusses the relation between local pattern mining and clustering and arrives at an algorithm finding clusters characterized by local patterns whose interestingness is measured w.r.t. a background distribution. Found partitions are then successively refined further. The relationship between local and global models w.r.t. classification rule learning is the topic of [18]. This work is more concerned with filtering and combining mined patterns to build a classifier, though.

A short discussion of the unification of supervised (*subgroup discovery*, *classification*) and unsupervised learning (*conceptual clustering*) can be found in [15]. The authors mention that supervised learning aims at predicting a single

attribute, unsupervised learning all attributes, and that tasks between those two goals could be imagined but do not seem to pursue this further, as we do.

5.2 Related algorithms

Similar ideas to the ones discussed in *cluster-grouping* have been explored in the BRUTE system [34]. It performs a bounded, exhaustive search to find the k best so-called *nuggets*. These nuggets are high-accuracy rules, essentially local patterns that have a high predictive power for a potentially small set of instances. In this regard they are similar to rules describing subgroups. Instead of a minimum interestingness threshold, BRUTE asks the user to specify a minimum search depth and possibly also minimum number of positives covered and a beam size.

Webb *et al.* [37] presented an algorithm for mining the best k frequent patterns according to an additional interestingness measure, specifically *leverage* which is calculated in the same way as *WRacc*. The approach is similar to CG, employing a dynamic threshold for pruning. The pruning rules are specifically tailored to leverage, in contrast to the technique used here. The author identifies finding additional constraints and according pruning rules as a future research direction. The addition of a frequency constraint seems to conflict with our intuition that a pattern should be interesting w.r.t. statistical considerations.

The CG algorithm is a substantial extension of the Morishita and Sese algorithm for *correlated pattern mining*, APRIORISMP [29]. Morishita and Sese have also adapted the basic APRIORISMP [35] to cope with multiple numerical attributes in the consequent part of rules. By performing clustering of numerical target values using the convex interclass variance criterion, they *are* defining a further *cluster-grouping* task. The most important difference with the work of Morishita and Sese [35] is thus that they only look for the k best rules achieving this clustering effect and did not study the application of these rules to hierarchical *conceptual clustering*, *subgroup discovery*, or *classification*, which is the most important contribution of the present work.

A similar technique has been developed independently by Bay and Pazzani [2]. Their name for patterns that discriminate strongly between several values of a designated attribute is *contrast sets*. Bay and Pazzani also use the convexity of χ^2 to derive an upper bound for patterns regarding a multi-valued target attribute. The main difference with Morishita and Sese’s work lies in the fact that the latter derive a general upper-bound framework applicable to all convex correlation measures is developed, which we further generalized into the *cluster-grouping* framework.

The *cluster-grouping* problem is also related to feature selection in *conceptual clustering* and to semi-flexible prediction [36, 6]. Talavera’s [36] motivation for feature selection in *conceptual clustering* is somewhat related to our motivation insofar as he is aiming for better comprehensibility, exclusion of irrelevant features and more efficient *clustering* processes (both when creating and using the clusters). There is also some similarity in where in the algorithm the feature are selected, since it is recomputed for each node in the hierarchical *clustering* tree. This is called *local* or *dynamic* selection. The main differences with our work

are two-fold: Firstly, Talavera’s work still retains COBWEB’s representation and only achieves better comprehensibility by reducing the number of considered attributes. Secondly, in his approach each attribute is scored *before* the actual *clustering* step, whereas CG performs feature selection as part of the *clustering* process itself.

Cardie [6] defines semi-flexible prediction as learning to predict a *set* of features known *a priori* as opposed to inflexible prediction (classification) and flexible prediction (clustering). Her approach involves automated feature selection for each attribute to be predicted separately. These features are then used in subsequent independent prediction of the attributes. In contrast, we attempt to predict a disjunction of attributes from a shared set of antecedents instead.

Finally, *cluster-grouping* is in many aspects related to the confirmatory induction setting in the *Tertius* system by Flach *et al.* [16]. As in CG, several target attributes are considered. It is interesting to note in this context that the rule head is treated as a single target while CG treats each condition separately. Flach’s work diverges from the general correlation setting in which correlation is symmetric and instead focuses on the number of counter-instances to a given rule, thus considering only directed associations. Using an *optimistic* estimate (an upper bound) they prune non-promising candidates and find and rank optimal rules. Focusing on counter-instances only allows more flexibility regarding the rule head, that is, the set of conditions need not be fixed.

6 Conclusions and Future Work

We have introduced the problem of *cluster-grouping* and argued that it is a sub-problem in a wide variety of popular machine learning and data mining tasks, such as *correlated pattern mining*, *subgroup discovery*, *classification*, and *conceptual clustering*.

A key contribution of this paper is the formulation of the CG algorithm for tackling the *cluster-grouping* task. We have also argued that it can be used as a universal local pattern mining component in systems tackling important machine learning and data mining tasks. Furthermore, using the CG algorithm has several advantages that often help to alleviate some of the problems with existing systems:

1. CG *always* outputs the k best solutions according to the interestingness function σ . This contrasts with current approaches to the *subgroup discovery*, *classification*, and *conceptual clustering* settings, where the quality of the discovered solutions depends on user-set parameters such as beam-size, and which do not guarantee optimality of the found descriptions.
2. An effective pruning technique uses the best σ values seen so far to dynamically remove those parts of the search space that cannot lead to solutions. This procedure often considers fewer candidate rules than a beam search technique (cf. also the experiments in Section 4.2) or complete enumeration techniques as *associative classification* or brute-force search.

3. The optimization with regard to interestingness measures is based on statistical theory. Additionally, setting a parameter k to limit the size of the solution set is – arguably – more intuitive than the specification of a beam size or minimum support threshold.

We have shown that our approach is an extension of Morishita’s and Sese’s work that allows one to apply the underlying ideas to more flexible target definitions and thus additional problem settings. We have provided experimental evidence that CG is well-suited for rule-based *subgroup discovery* (CG-SD), use in classification (CN2-CG,CBC), and conceptual clustering (CG-CLUS). Different variants of existing and novel algorithms were implemented and experimentally compared to state-of-the-art techniques for solving these tasks. In most of cases the CG based approach improved upon alternative techniques in efficiency or performance. Especially worth mentioning are two novel algorithms, CBC and CG-CLUS, which target associative classification and divisive clustering respectively. CBC is a natural alternative to systems such as CMAR and CBA that derive association rules using support and confidence. The CG-CLUS algorithm is competitive with one of the best-known *conceptual clustering* algorithms, COBWEB, and computes rule-sets that are easier to interpret.

Further research will proceed in several directions. First, as can be seen in the experiments, the effectiveness of the pruning step depends strongly on the tightness of the upper bound calculated. Therefore, it is desirable to tighten future support estimates and therefore attainable values of σ . Second, the technique should in principle be usable in the formation of multi-variate decision trees [30]. For such a setting it would be necessary to extend the upper-bound techniques to multi-valued target attributes. For the classification setting this extension could take the form of learning rules involving error-correcting output codes [11, 26].

Another direction is the application to other learning areas. We have already employed the basic principles of CG in a different domain, tree-structured data [5] and the *cluster-grouping* paradigm could also be extended into the area of *inductive logic programming*.

Acknowledgments

We sincerely thank Shinichi Morishita and Jun Sese for useful discussion and feedback. We also thank our fellow researchers Kristian Kersting, Björn Bringmann, and Ulrich Rückert for their thorough reviews, constructive comments and helpful suggestions.

This work was partly supported by the EU IST project cInQ (consortium on discovering knowledge with Inductive Queries), contract no. IST-2000-26469, and the EU IST project IQ (Inductive Querying), contract no. IST-FET FP6-516169.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases*, pages 487–499, Santiago de Chile, Chile, September 1994. Morgan Kaufmann.
2. Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining constraint sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.
3. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
4. Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *SIGMOD Conference*, pages 265–276. ACM Press, 1997.
5. Björn Bringmann and Albrecht Zimmermann. Tree² - decision trees for tree structured data. In Alípio Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 46–58. Springer, 2005.
6. Claire Cardie. Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32, Amherst, Massachusetts, USA, June 1993. Morgan Kaufmann.
7. Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. In John E. Laird, editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, Ann Arbor, Michigan, USA, June 1988. Morgan Kaufmann.
8. Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
9. Frans Coenen and Paul Leng. Obtaining best parameter values for accurate classification. In Jiawei Han, Benjamin W. Wah, Vijay Raghavan, Xindong Wu, and Rajeev Rastogi, editors, *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 597–600, Houston, Texas, USA, November 2005. IEEE.
10. William W. Cohen. Fast effective rule induction. In Armand Friediris and Stuart J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, Tahoe City, California, USA, July 1995. Morgan Kaufmann.
11. Thomas G. Dietterich and Ghulam Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *AAAI*, pages 572–577, Anaheim, California, USA, July 1991. AAAI Press/The MIT Press.
12. Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, Chambéry, France, August 1993. Morgan Kaufmann.
13. Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.
14. Douglas H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research (JAIR)*, 4:147–178, 1996.
15. Douglas H. Fisher and Gilford Hapanyengwi. Database management and analysis tools of machine learning. *Journal of Intelligent Information Systems*, 2:5–38, 1993.
16. Peter A. Flach and Nicolas Lachiche. Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning*, 42(1/2):61–95, 2001.
17. Eibe Frank and Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

18. Johannes Fürnkranz. From local to global patterns: Evaluation issues in rule learning algorithms. In Morik et al. [28], pages 20–38.
19. Johannes Fürnkranz and Peter A. Flach. ROC 'n' rule learning-towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, 2005.
20. Mark A. Gluck and James E. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, pages 283–287, Irvine, California, USA, 1985. Lawrence Erlbaum Associates.
21. Frank Höppner. Local pattern detection and clustering. In Morik et al. [28], pages 53–70.
22. Nada Lavrač, Peter A. Flach, and Blaz Zupan. Rule evaluation measures: A unifying view. In Sašo Džeroski and Peter A. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, pages 174–185, Bled, Slovenia, June 1999. Springer.
23. Nada Lavrac, Branko Kavsek, Peter A. Flach, and Ljupco Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.
24. Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376, San José, California, USA, 2001. IEEE Computer Society.
25. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 80–86, New York City, New York, USA, August 1998. AAAI Press.
26. Francesco Masulli and Giorgio Valentini. Effectiveness of error correcting output codes in multiclass learning problems. In Josef Kittler and Fabio Roli, editors, *Proceedings on the First International Workshop on Multiple Classifier Systems*, pages 107–116, Cagliari, Italy, June 2000. Springer.
27. Ryszard S. Michalski and Robert E. Stepp. Learning from observation: Conceptual clustering. *Machine Learning, An Artificial Intelligence Approach*, 1:331–363, 1983.
28. Katharina Morik, Jean-François Boulicaut, and Arno Siebes, editors. *Local Pattern Detection, International Seminar, Revised Selected Papers*, Dagstuhl Castle, Germany, April 2004. Springer.
29. Shinichi Morishita and Jun Sese. Traversing itemset lattices with statistical metric pruning. In *Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 226–236, Dallas, Texas, USA, May 2000. ACM.
30. Sreerama K. Murthy. *On Growing Better Decision Trees from Data*. PhD thesis, John Hopkins University, Baltimore, Maryland, USA, 1997.
31. Stefan Mutter, Mark Hall, and Frank Frank. Using classification to evaluate the output of confidence-based association rule mining. In Geoffrey I. Webb and Xinghuo Yu, editors, *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 538–549, Cairns, Australia, December 2004. Springer.
32. Arthur J. Nevins. A branch and bound incremental conceptual clusterer. *Machine Learning*, 18(1):5–22, 1995.
33. Mike Perkowitz and Oren Etzioni. Adaptive web sites: Conceptual cluster mining. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 264–269, Stockholm, Sweden, July 1999. Morgan Kaufmann.

34. Patricia J. Riddle, Richard Segal, and Oren Etzioni. Representation design and brut-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence*, 8(1):125–147, 1994.
35. Jun Sese and Shinichi Morishita. Itemset classified clustering. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Proceedings of the 8th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 398–409, Pisa, Italy, September 2004. Springer.
36. Luis Talavera. Dynamic feature selection in incremental hierarchical clustering. In Ramon López de Mántaras and Enric Plaza, editors, *Proceedings of the 11th European Conference on Machine Learning*, pages 392–403, Barcelone, Catalonia, Spain, May 2000. Springer.
37. Geoffrey I. Webb and Songmao Zhang. K-optimal rule discovery. *Data Min. Knowl. Discov.*, 10(1):39–79, 2005.
38. Albrecht Zimmermann and Luc De Raedt. Cluster-grouping: From subgroup discovery to clustering. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning*, pages 575–577, Pisa, Italy, September 2004. Springer.
39. Albrecht Zimmermann and Luc De Raedt. Corclass: Correlated association rule mining for classification. In Einoshin Suzuki and Setsuo Arikawa, editors, *Proceedings of the 7th International Conference on Discovery Science*, pages 60–72, Padova, Italy, October 2004. Springer.
40. Albrecht Zimmermann and Luc De Raedt. Inductive querying for discovering subgroups and clusters. In *Constraint-Based Mining and Inductive Databases*, pages 380–399, 2004.

A Convexity Proofs

A.1 Convexity of *Weighted Relative Accuracy* (*WRAcc*)

Let A be an attribute, $v \in \mathcal{V}[A]$ a possible value of A , $\mathcal{E}$ a data set, r a rule of the form $b \rightsquigarrow A = v$, with $x = \sup(b)$, $y = \sup(x \rightsquigarrow A = v)$, $m = \sup(A = v)$, $n = |\mathcal{E}|$.

Then the usual definition of *WRAcc*:

$$P(b)(P(A = v|b) - P(A = v)) \text{ can be redefined as: } WRAcc(x, y) = \frac{x}{n} \left(\frac{y}{x} - \frac{m}{n} \right)$$

To prove the convexity of *WRAcc*, we directly check the convexity criterion:

$$\begin{aligned} WRAcc(\lambda(x_1, y_1) + (1 - \lambda)(x_2, y_2)) &= \frac{\lambda x_1 + (1 - \lambda)x_2}{n} \left(\frac{\lambda y_1 + (1 - \lambda)y_2}{\lambda x_1 + (1 - \lambda)x_2} - \frac{m}{n} \right) \\ &= \frac{\lambda y_1 + (1 - \lambda)y_2}{n} - \frac{m(\lambda x_1 + (1 - \lambda)x_2)}{n^2} = \frac{\lambda y_1}{n} - \frac{\lambda m x_1}{n^2} + \frac{(1 - \lambda)y_2}{n} - \frac{(1 - \lambda)m x_2}{n^2} \\ &= \frac{\lambda x_1}{n} \left(\frac{y_1}{x_1} - \frac{m}{n} \right) + \frac{(1 - \lambda)x_2}{n} \left(\frac{y_2}{x_2} - \frac{m}{n} \right) = \lambda WRAcc(x_1, y_1) + (1 - \lambda) WRAcc(x_2, y_2) \end{aligned}$$

Since the two terms are equal *WRAcc* is not strictly convex function.

A.2 Convexity of *Category Utility* (*CU*)

As shown in 3.3, *CU* can be decomposed into a sum of partial *CUs*. If reformulated in the stamp point notation, *CU* becomes:

$$\begin{aligned} CU(\langle x, y_1, \dots, y_d \rangle) &= \sum_{i=1}^d CU(x, y_i) \\ &= \sum_{i=1}^d \frac{1}{2} \frac{x}{n} \left(\left(\frac{y_i}{x} \right)^2 - \left(\frac{m_i}{n} \right)^2 + \left(\frac{x - y_i}{x} \right)^2 - \left(\frac{n - m_i}{n} \right)^2 \right) \\ &\quad + \frac{1}{2} \frac{n - x}{n} \left(\left(\frac{m_i - y_i}{n - x} \right)^2 - \left(\frac{m_i}{n} \right)^2 + \left(\frac{n - m_i - (x - y_i)}{n - x} \right)^2 - \left(\frac{n - m_i}{n} \right)^2 \right) \end{aligned}$$

Since a sum of convex functions is itself is again convex it is enough to prove the convexity of partial *CU*. Additionally to directly checking the convexity property there is another way to prove convexity whose presentation takes up less space. For a twice differentiable function to be convex, its Hessian has to be positive semi-definite. The Hessian is the matrix of the function's second partial derivatives and for the two-dimensional case has the form:

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

A matrix is positive semi-definite if the determinants of all its leading principal minors are ≥ 0 . This implies that we have to show that:

$$\frac{\partial^2 f}{\partial x^2} \geq 0, \frac{\partial^2 f}{\partial y^2} \geq 0, \text{ and } \frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \frac{\partial^2 f}{\partial x \partial y} \frac{\partial^2 f}{\partial y \partial x} \geq 0$$

A partial CU in stamp point notation is a sum that can be decomposed further into the part corresponding to the instances covered by the rule body and the instances not covered by the rule body. Again it holds that if those two terms are convex, the entire partial CU is convex. The corresponding Hessians are:

$$\begin{pmatrix} \frac{2y^2}{nx^3} & \frac{-2y}{nx^2} \\ \frac{-2y}{nx^2} & \frac{2}{nx} \end{pmatrix} \begin{pmatrix} \frac{2(y-m)^2}{(n-x)^3n} & \frac{2y-2m}{(n-x)^2n} \\ \frac{2y-2m}{(n-x)^2n} & \frac{2}{2(n-x)n} \end{pmatrix}$$

The $\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2}$ are obviously greater or than zero, so all that is left is checking the determinants of the whole matrices. For the “positive” part (the part that corresponds to covered instances) this determinant is:

$$\frac{4y^2}{n^2x^4} - \frac{4y^2}{n^2x^4} = 0 \text{ and for the “negative” part: } \frac{4(y-m)^2}{(n-x)^4n^2} - \frac{4(y-m)^2}{(n-x)^4n^2} = 0$$

So the Hessian of CU is positive semidefinite and thus CU is a convex function.

A Parameter-Free Associative Classification Method

Loïc Cerf¹, Dominique Gay², Nazha Selmaoui², and Jean-François Boulicaut¹

¹ INSA-Lyon, LIRIS CNRS UMR5205,
F-69621 Villeurbanne, France
{loic.cerf, jean-francois.boulicaut}@liris.cnrs.fr
² Université de la Nouvelle-Calédonie, ERIM EA 3791,
98800 Nouméa, Nouvelle-Calédonie
{dominique.gay, nazha.selmaoui}@univ-nc.nc

Abstract. In many application domains, classification tasks have to tackle multiclass imbalanced training sets. We have been looking for a CBA approach (Classification Based on Association rules) in such difficult contexts. Actually, most of the CBA-like methods are one-vs-all approaches (OVA), i.e., selected rules characterize a class with what is relevant for this class and irrelevant for the union of the other classes. Instead, our method considers that a rule has to be relevant for one class and irrelevant for every other class taken separately. Furthermore, a constrained hill climbing strategy spares users tuning parameters and/or spending time in tedious post-processing phases. Our approach is empirically validated on various benchmark data sets.

Key words: Classification, Association Rules, Parameter Tuning, Multiclass

1 Introduction

Association rule mining [1] has been applied not only for descriptive tasks but also for supervised classification based on labeled transactional data [2–8]. An association rule is an implication of the form $X \Rightarrow Y$ where X and Y are different sets of Boolean attributes (also called items). When Y denotes a single class value, it is possible to look at the predictive power of such a rule: when the conjunction X is observed, it is sensible to predict that the class value Y is true. Such a shift between descriptive and predictive tasks needs for careful selection strategies [9]. [2] identified it as an associative classification approach (also denoted CBA-like methods thanks to the name chosen in [2]). The pioneering proposal in [2] is based on the classical objective interestingness measures for association rules – *frequency* and *confidence* – for selecting candidate classification rules. Since then, the selection procedure has been improved leading to various CBA-like methods [2, 4, 6, 8, 10]. Unfortunately, support-confidence-based methods show their limits on imbalanced data sets. Indeed, rules with high confidence can also be negatively correlated. [11, 12] propose new methods based on correlation measure to overcome this weakness. However, when

considering a n -class imbalanced context, even a correlation measure is not satisfactory: a rule can be positively correlated with two different classes what leads to conflicting rules. The common problem of these approaches is that they are OVA (one-vs-all) methods, i.e., they split the classification task into n two-class classification tasks (positives vs negatives) and, for each sub-task, look for rules that are relevant in the *positive* class and irrelevant for the union of the other classes. Notice also that the popular emerging patterns (EPs introduced in [13]) and the associated EPs-based classifiers (see e.g. [14] for a survey) are following the same principle. Thus, they can lead to conflicting EPs.

In order to improve state-of-the-art approaches for associative classification when considering multiclass imbalanced training sets, our contribution is twofold. First, we propose an OVE (one-vs-each) method that avoids some of the problems observed with typical CBA-like methods. Indeed, we formally characterize the association rules that can be used for classification purposes when considering that a rule has to be relevant for one class and irrelevant for every other class (instead of being irrelevant for their union). Next, we designed a constrained hill climbing technique that automatically tunes the many parameters (frequency thresholds) that are needed. The paper is organized as follows: Section 2 provides the needed definitions. Section 3 discusses the relevancy of the rules extracted thanks to the algorithm presented in Section 4. Section 5 describes how the needed parameters are automatically tuned. Section 6 provides our experimental study on various benchmark data sets. Section 7 briefly concludes.

2 Definitions

Let $\mathcal{C}$ be the set of classes and n its cardinality. Let $\mathcal{A}$ be the set of Boolean attributes. An object o is defined by the subset of attributes that holds for it, i.e., $o \subseteq \mathcal{A}$. The data in Table 1 illustrate the various definitions. It provides 11 classified objects $(o_k)_{k \in 1 \dots 11}$. Each of them is described with some of the 6 attributes $(a_l)_{l \in 1 \dots 6}$ and belongs to one class $(c_i)_{i \in 1 \dots 3}$. This is a toy labeled transactional data set that can be used to learn an associative classifier that may predict the class value among the three possible ones.

2.1 Class Association Rule

A *Class Association Rule* (CAR) is an ordered pair $(X, c) \in 2^{\mathcal{A}} \times \mathcal{C}$. X is the *body* of the CAR and c its *target class*.

Example 1. In Tab. 1, $(\{a_1, a_5\}, c_3)$ is a CAR. $\{a_1, a_5\}$ is the body of this CAR and c_3 its target class.

2.2 Per-class Frequency

Given a class $d \in \mathcal{C}$ and a set $\mathcal{O}_d$ of objects belonging to this class, the *frequency* of a CAR (X, c) in d is $|\{o \in \mathcal{O}_d | X \subseteq o\}|$. Since the frequency of (X, c) in d

Table 1. Eleven classified objects.

		$\mathcal{A}$						$\mathcal{C}$		
		a_1	a_2	a_3	a_4	a_5	a_6	c_1	c_2	c_3
$\mathcal{O}_{c_1}$	o_1	.		.	.			.		
	o_2	.	.	.	.	.		.		
	o_3	.	.		.			.		
	o_4	.	.	.		.		.		
	o_5	.	.	.		.		.		
$\mathcal{O}_{c_2}$	o_6		.		.		.	.		
	o_7	.	.		.			.		
	o_8	.	.			.		.		
$\mathcal{O}_{c_3}$	o_9	.				.			.	
	o_{10}	.				.			.	
	o_{11}					.			.	

does not depend on c , it is denoted $f_d(X)$. Given $d \in \mathcal{C}$ and a related frequency threshold $\gamma \in \mathbb{N}$, (X, c) is *frequent* (resp. *infrequent*) in d iff $f_d(X)$ is at least (resp. strictly below) γ .

Example 2. In Tab. 1, the CAR $(\{a_1, a_5\}, c_3)$ has a frequency of 3 in c_1 , 1 in c_2 and 2 in c_3 . Hence, if a frequency threshold $\gamma = 2$ is associated to c_3 , $(\{a_1, a_5\}, c_3)$ is frequent in c_3 .

With the same notations, the *relative frequency* of a CAR (X, c) in d is $\frac{f_d(X)}{|\mathcal{O}_d|}$.

2.3 Interesting Class Association Rule

Without any loss of generality, consider that $\mathcal{C} = \{c_i | i \in 1 \dots n\}$. Given $i \in 1 \dots n$ and $(\gamma_{i,j})_{j \in 1 \dots n} \in \mathbb{N}^n$ (n per-class frequency thresholds pertaining to each of the n classes), a CAR (X, c_i) is said *interesting* iff:

1. it is frequent in c_i , i.e., $f_{c_i}(X) \geq \gamma_{i,i}$
2. it is infrequent in every other class, i.e., $\forall j \neq i, f_{c_j}(X) < \gamma_{i,j}$
3. any more general CAR is frequent in at least one class different from c_i , i.e., $\forall Y \subset X, \exists j \neq i | f_{c_j}(Y) \geq \gamma_{i,j}$ (*minimal body* constraint).

Example 3. In Tab. 1, assume that the frequency thresholds $\gamma_{3,1} = 4$, $\gamma_{3,2} = 2$, and $\gamma_{3,3} = 2$ are respectively associated to c_1 , c_2 , and c_3 . Although it is frequent in c_3 and infrequent in both c_1 and c_2 , $(\{a_1, a_5\}, c_3)$ is not an interesting CAR since $\{a_5\} \subset \{a_1, a_5\}$ and $(\{a_5\}, c_3)$ is neither frequent in c_1 nor in c_2 .

3 Relevancy of the Interesting Class Association Rules

3.1 Selecting Better Rules

Constructing a CAR-based classifier means selecting relevant CARs for classification purposes. Hence, the space of CARs is to be split into two: the *relevant*

CARs and the *irrelevant* ones. Furthermore, if $\preceq$ denotes a relevancy (possibly partial) order on the CARs, there should not be a rule r from the relevant CARs and a rule s from the irrelevant CARs s.t. $r \preceq s$. If this never happens, we say that the frontier between relevant and irrelevant CARs is *sound*. Notice that [3] uses the same kind of argument but conserves a one-vs-all perspective.

Using the “Global Frequency + Confidence” Order. The influential work from [2] has been based on a frontier derived from the conjunction of a global frequency (sum of the per-class frequencies for all classes) threshold and a confidence (ratio between the per-class frequency in the target class and the global frequency) threshold. Let us consider the following partial order $\preceq_1$: $\forall (X, Y) \in (2^A)^2, \forall c \in \mathcal{C}$,

$$(X, c) \preceq_1 (Y, c) \Leftrightarrow \forall d \in \mathcal{C}, \begin{cases} f_c(X) \leq f_d(Y) & \text{if } c = d \\ f_c(X) \geq f_d(Y) & \text{otherwise.} \end{cases}$$

Obviously, $\preceq_1$ is a sensible relevancy order. However, as emphasized in the example below, the frontier drawn by the conjunction of a global frequency threshold and a confidence threshold is not sound w.r.t. $\preceq_1$.

Example 4. Assume a global frequency threshold of 5 and a confidence threshold of $\frac{3}{5}$. In Tab. 1, the CAR $(\{a_3\}, c_1)$ is not (globally) frequent. Thus it is on the irrelevant side of the frontier. At the same time, $(\{a_4\}, c_1)$ is both frequent and with a high enough confidence. It is on the relevant side of the frontier. However, $(\{a_3\}, c_1)$ correctly classifies more objects of $\mathcal{O}_1$ than $(\{a_4\}, c_1)$ and it applies on less objects outside $\mathcal{O}_1$. So $(\{a_4\}, c_1) \preceq_1 (\{a_3\}, c_1)$.

Using the “Emergence” Order. *Emerging patterns* have been introduced in [13]. Here, the frontier between relevancy and irrelevancy relies on a growth rate threshold (ratio between the relative frequency in the target class and the relative frequency in the union of all other classes). As emphasized in the example below, the low number of parameters (one growth rate threshold for each of the n classes) does not support a fine tuning of this frontier.

Example 5. Assume a growth rate threshold of $\frac{8}{5}$. In Tab. 1, the CAR $(\{a_1\}, c_1)$ has a growth rate of $\frac{3}{2}$. Thus it is on the irrelevant side on the frontier. At the same time, $(\{a_2\}, c_1)$ has a growth rate of $\frac{8}{5}$. It is on the relevant side of the frontier. However $(\{a_1\}, c_1)$ correctly classifies more objects of $\mathcal{O}_1$ than $(\{a_2\}, c_1)$ and more clearly differentiates objects in $\mathcal{O}_1$ from those in $\mathcal{O}_2$.

Using the “Interesting” Order. The frontier drawn by the growth rates is sound w.r.t. $\preceq_1$. So is the one related to the so-called interesting CARs. Nevertheless, the latter can be more finely tuned so that the differentiation between two classes is better performed. Indeed, the set of interesting CARs whose target class is c_i is parametrized by n thresholds $(\gamma_{i,j})_{j \in 1 \dots n}$: one frequency threshold

$\gamma_{i,i}$ and $n - 1$ infrequency thresholds for each of the $n - 1$ other classes (instead of one for all of them). Hence, to define a set of interesting CARs targeting every class, n^2 parameters enable to finely draw the frontier between relevancy and irrelevancy.

In practice, this quadratic growth of the number of parameters can be seen as a drawback for the experimenter. Indeed, in the classical approaches presented above, this growth is linear and finding the proper parameters already appears as a dark art. This issue will be solved in Section 5 thanks to an automatic tuning of the frequency thresholds.

3.2 Preferring General Class Association Rules

The minimal body constraint avoids redundancy in the set of interesting CARs. Indeed, it can easily be shown that, for every CAR (X, c) , frequent in c and infrequent in every other class, it exists a body $Y \subseteq X$ s.t. (Y, c) is interesting and $\forall Z \subset Y, (Z, c)$ is not. Preferring shorter bodies means focusing on more general CARs. Hence, the interesting CARs are more prone to be applicable to new unclassified objects. Notice that the added-value of the minimal body constraint has been well studied in previous approaches for associative classification (see, e.g., [5, 7]).

4 Computing and Using the Interesting Class Association Rules

Let us consider n classes $(c_i)_{i \in 1 \dots n}$ and let us assume that Γ denotes a $n \times n$ matrix of frequency thresholds. The i^{th} line of Γ pertains to the subset of interesting CARs whose target class is c_i . The j^{th} column of Γ pertains to the frequency thresholds in c_j . Given Γ and a set of classified objects, we discuss how to efficiently compute the complete set of interesting CARs.

4.1 Enumeration

The complete extraction of the interesting CARs is performed one target class after another. Given a class $c_i \in \mathcal{C}$, the enumeration strategy of the candidate CARs targetting c_i is critical for performance issues. The search space of the CAR bodies, partially ordered by $\subseteq$, has a lattice structure. It is traversed in a breadth-first way. The two following properties enable to explore only a small part of it without missing any interesting CAR:

1. If (Y, c_i) is not frequent in c_i , neither is any (X, c_i) with $Y \subseteq X$.
2. If (Y, c_i) is an interesting CAR, any (X, c_i) with $Y \subset X$ does not have a minimal body.

Such CAR bodies Y are collected into a prefix tree. When constructing the next level of the lattice, every CAR body in the current level is enlarged s.t. it does not become a superset of a body in the prefix tree. In this way, entire sublattices, which cannot contain bodies of interesting CARs, are ignored.

4.2 Algorithm

Algorithm 1 details how the extraction is performed. **parents** denotes the current level of the lattice (i.e., a list of CAR bodies). **futureParents** is the next level. **forbiddenPrefixes** is the prefix tree of forbidden subsets from which is computed **forbiddenAtts**, the list of attributes that are not allowed to enlarge **parent** (a CAR body in the current level) to give birth to its **children** (bodies in the next level).

```

forbiddenPrefixes  $\leftarrow \emptyset$ 
parents  $\leftarrow [\emptyset]$ 
while parents  $\neq []$  do
  futureParents  $\leftarrow \emptyset$ 
  for all parent  $\in$  parents do
    forbiddenAtts  $\leftarrow$  FORBIDDENATTS(forbiddenPrefixes, parent)
    for all attribute  $>$  LASTATTRIBUTE(parent) do
      if attribute  $\notin$  forbiddenAtts then
        child  $\leftarrow$  CONSTRUCTCHILD(parent, attribute)
        if  $f_{c_i}(\text{child}) \geq \gamma_{i,i}$  then
          if INTERESTING(child) then
            output (child,  $c_i$ )
            INSERT(child, forbiddenPrefixes)
          else
            futureParents  $\leftarrow$  futureParents  $\cup$  {child}
        else
          INSERT(child, forbiddenPrefixes)
      parents  $\leftarrow$  parents  $\setminus$  {parent}
  parents  $\leftarrow$  futureParents

```

Algorithm 1: EXTRACT(c_i : target class)

4.3 Simultaneously Enforcing Frequency Thresholds in All Classes

Notice that, along the extraction of the interesting CARs targeting c_i , all frequency thresholds $(\gamma_{i,j})_{j \in 1 \dots n}$ are simultaneously enforced. To do so, every CAR body in the lattice is bound to n bitsets related to the $(\mathcal{O}_{c_i})_{i \in 1 \dots n}$. Thus, every bit stands for the match ('1') or the mismatch ('0') of an object and bitwise ANDs enables an incremental and efficient computation of **children's** bitsets.

Alternatively, the interesting CARs targeting c_i could be obtained by computing the $n - 1$ sets of emerging patterns between c_i and every other class c_j (with $\frac{\gamma_{i,i}|\mathcal{O}_j|}{\gamma_{i,j}|\mathcal{O}_i|}$ as a growth rate), one by one, and intersecting them. However, the time complexity of $n^2 - n$ extractions of loosely constrained CARs is far worse than ours (n extractions of tightly constrained CARs). When n is large, it prevents from automatically tuning the parameters with a hill climbing technique.

4.4 Classification

When an interesting CAR is output, we can output its vector of relative frequencies in all classes at no computational cost. Then, for a given unclassified object $o \subseteq \mathcal{A}$, its likeliness to be in the class c_i is quantifiable by $l(o, c_i)$ which is the sum of the relative frequencies in c_i of all interesting CARs applicable to o :

$$l(o, c_i) = \sum_{c \in \mathcal{C}} \left(\sum_{\text{interesting } (X, c) \text{ s.t. } X \subseteq o} \left(\frac{f_{c_i}(X)}{|\mathcal{O}_{c_i}|} \right) \right)$$

Notice that the target class of an interesting CAR does not hide the exceptions it may have in the other classes. The class $c_{\max}$ related to the greatest likeliness value $l(o, c_{\max})$ is where to classify o . $\min_{i \neq \max} \left(\frac{l(o, c_{\max})}{l(o, c_i)} \right)$ quantifies the certainty of the classification of o in the class $c_{\max}$ rather than c_i (the other class with which the confusion is the greatest). This “certainty measure” may be very valuable in cost-sensitive applications.

5 Automatic Parameter Tuning

It is often considered that manually tuning the parameters of an associative classification method, like our CAR-based algorithm, borders the dark arts. Indeed, our algorithm from Sec. 4 requires a n -by- n matrix Γ of input parameters. Fortunately, analyzing the way the interesting CARs apply to the learning set, directly indicates what frequency threshold in Γ should be modified to probably improve the classification. We now describe how to algorithmically tune Γ to obtain a set of interesting CARs that is well adapted to classification purposes. Due to space limitations, the pseudo-code of this algorithm, called **fitcare**³, is only available in an associated technical report [15].

5.1 Hill Climbing

The **fitcare** algorithm tunes Γ following a hill climbing strategy.

Maximizing the Minimal Global Growth Rate. Section 4.4 mentioned the advantages of not restricting the output of a CAR to its target class (its frequencies in every class are valuable as well). With the same argument applied to the global set of CARs, the hill climbing technique, embedded within **fitcare**, maximizes *global growth rates* instead of other measures (e.g., the number of correctly classified objects) where the loss of information is greater.

Given two classes $(c_i, c_j) \in \mathcal{C}^2$ s.t. $i \neq j$, the *global growth rate* $g(c_i, c_j)$ quantifies, when classifying the objects from $\mathcal{O}_{c_i}$, the confusion with the class c_j . The greater it is, the less confusion made. We define it as follows:

$$g(c_i, c_j) = \frac{\sum_{o \in \mathcal{O}_{c_i}} l(o, c_i)}{\sum_{o \in \mathcal{O}_{c_i}} l(o, c_j)}$$

³ **fitcare** is the recursive acronym for **fitcare** is **the** class association rule **extractor**.

From a set of interesting CARs, **fitcare** computes all $n^2 - n$ global growth rates. The maximization of the minimal global growth rate drives the hill climbing, i.e., **fitcare** tunes Γ so that this rate increases. When no improvement can be achieved on the smallest global growth rate, **fitcare** attempts to increase the second smallest (while not decreasing the smallest), etc. **fitcare** terminates when a maximum is reached.

Choosing one $\gamma_{i,j}$ to lower. Instead of a random initialization of the parameters (a common practice in hill climbing techniques), Γ is initialized with high frequency thresholds. The hill climbing procedure only lowers these parameters, one at a time, and by decrements of 1. However, we will see, in Sec. 5.2, that such a modification leads to lowering other frequency thresholds if Γ enters an undesirable state.

The choice of the parameter $\gamma_{i,j}$ to lower depends on the global growth rate $g(c_i, c_j)$ to increase. Indeed, when classifying the objects from $\mathcal{O}_{c_i}$, different causes lead to a confusion with c_j . To discern the primary cause, every class at the denominator of $g(c_i, c_j)$ is evaluated separately:

$$\begin{pmatrix} \sum_{o \in \mathcal{O}_{c_i}} \sum_{\text{interesting}(X, c_1) \text{ s.t. } X \subseteq o} \left(\frac{f_{c_j}(X)}{|\mathcal{O}_{c_j}|} \right) \\ \sum_{o \in \mathcal{O}_{c_i}} \sum_{\text{interesting}(X, c_2) \text{ s.t. } X \subseteq o} \left(\frac{f_{c_j}(X)}{|\mathcal{O}_{c_j}|} \right) \\ \vdots \\ \sum_{o \in \mathcal{O}_{c_i}} \sum_{\text{interesting}(X, c_n) \text{ s.t. } X \subseteq o} \left(\frac{f_{c_j}(X)}{|\mathcal{O}_{c_j}|} \right) \end{pmatrix}$$

The greatest term is taken as the primary cause for $g(c_i, c_j)$ to be small. Usually it is either the i^{th} term (the interesting CARs targeting c_i are too frequent in c_j) or the j^{th} one (the interesting CARs targeting c_j are too frequent in c_i). This term directly indicates what frequency threshold in Γ should be preferably lowered. Thus, if the i^{th} (resp. j^{th}) term is the greatest, $\gamma_{i,j}$ (resp. $\gamma_{j,i}$) is lowered. Once Γ modified and the new interesting CARs extracted, if $g(c_i, c_j)$ increased, the new Γ is committed. If not, Γ is rolled-back to its previous value and the second most promising $\gamma_{i,j}$ is decremented, etc.

5.2 Avoiding Undesirable Parts of the Parameter Space

Some values for Γ are obviously bad. Furthermore, the hill climbing technique cannot properly work if too few or too many CARs are interesting. Hence, **fitcare** avoids these parts of the parameter space.

Sensible Constraints on Γ . The relative frequency of an interesting CAR targeting c_i should obviously be strictly greater in c_i than in any other class:

$$\forall i \in 1 \dots n, \forall j \neq i, \frac{\gamma_{i,j}}{|\mathcal{O}_{c_j}|} < \frac{\gamma_{i,i}}{|\mathcal{O}_{c_i}|}$$

Furthermore, the set of interesting CARs should be *conflictless*, i.e., if it contains (X, c_i) , it must not contain (Y, c_j) if $Y \subseteq X$. Thus, an interesting CAR targeting c_i must be strictly more frequent in c_i than any interesting CAR whose target class is not c_i :

$$\forall i \in 1 \dots n, \forall j \neq i, \gamma_{i,j} < \gamma_{j,j}$$

Whenever a modification of Γ violates one of these two constraints, every $\gamma_{i,j}$ ($i \neq j$) in cause is lowered s.t. Γ reaches another sensible state. Then, the extraction of the interesting CARs is performed.

Minimal Positive Cover Rate Constraint. Given a class $c \in \mathcal{C}$, the *positive cover rate* of c is the proportion of objects in $\mathcal{O}_c$ that are covered by at least one interesting CAR targeting c , i.e., $\frac{|\{o \in \mathcal{O}_c | \exists \text{ interesting } (X, c) \text{ s.t. } X \subseteq o\}|}{|\mathcal{O}_c|}$. Obviously, the smaller the positive cover rate of c , the worse the classification in c .

By default, **fitcare** forces the positive cover rates of every class to be 1 (every object is positively covered). Thus, whenever interesting CARs, with c_i as a target class, are extracted, the positive cover rate of c_i is returned. If it is not 1, $\gamma_{i,i}$ is lowered by 1 and the interesting CARs are extracted again.

Notice that **fitcare** lowers $\gamma_{i,i}$ until $\mathcal{O}_{c_i}$ is entirely covered but not more. Indeed, this could bring a disequilibrium between the average number of interesting CARs applying to the objects in the different classes. If this average in $\mathcal{O}_{c_i}$ is much higher than that of $\mathcal{O}_{c_j}$, $g(c_i, c_j)$ would be artificially high and $g(c_j, c_i)$ artificially low. Hence, the hill climbing strategy would be biased.

On some difficult data sets (e.g., containing misclassified objects), it may be impossible to entirely cover some class c_i while verifying $\forall i \in 1 \dots n, \forall j \neq i, \frac{\gamma_{i,j}}{|\mathcal{O}_{c_j}|} < \frac{\gamma_{i,i}}{|\mathcal{O}_{c_i}|}$. That is why, while initializing Γ , a looser minimal positive cover rate constraint may be decided.

Here is how the frequency thresholds in the i^{th} line of Γ are initialized (every line being independently initialized):

$$\forall j \in 1 \dots n, \gamma_{i,j} = \begin{cases} |\mathcal{O}_{c_j}| & \text{if } i = j \\ |\mathcal{O}_{c_j}| - 1 & \text{otherwise} \end{cases}$$

The interesting CARs targeting c_i are collected with $\text{EXTRACT}(c_i)$. Most of the time, the frequency constraint in c_i is too high for the interesting CARs to entirely cover $\mathcal{O}_{c_i}$. Hence **fitcare** lowers $\gamma_{i,i}$ (and the $(\gamma_{i,j})_{j \in 1 \dots n}$ s.t. $\forall i \in 1 \dots n, \forall j \neq i, \frac{\gamma_{i,j}}{|\mathcal{O}_{c_j}|} < \frac{\gamma_{i,i}}{|\mathcal{O}_{c_i}|}$) until $\mathcal{O}_{c_i}$ is entirely covered. If $\gamma_{i,i}$ reaches 0 but the positive cover rate of c_i never was 1, the minimal positive cover rate constraint is loosened to the greatest rate encountered so far. The frequency thresholds related to this greatest rate constitute the i^{th} line of Γ when the hill climbing procedure starts.

6 Experimental Results

The `fitcare` algorithm has been implemented in C++. We performed an empirical validation of its added-value on various benchmark data sets. The LUCS-KDD software library [16] provided the discretized versions of the UCI data sets [17] and a Java implementation of CPAR. Notice that we name the data sets according to Coenen’s notation, e.g., the data set “breast.D20.N699.C2” gathers 699 objects described by 20 Boolean attributes and organized in 2 classes. To put the focus on imbalanced data sets, the repartition of the objects into the classes is mentioned as well. Bold faced numbers of objects indicate minor classes, i.e., classes having, at most, half the cardinality of the largest class.

The global and the per-class accuracies of `fitcare` are compared to that of CPAR, one of the best CBA-like methods designed so far. The results, reported in Tab. 2, were obtained after 10-fold stratified cross validations.

Table 2. Experimental results of `fitcare` and comparison with CPAR.

Data Sets	Global		Per-class (True Positive rates)	
	fitcare	CPAR	fitcare	CPAR
anneal.D73.N898.C6 8/99/684/0/67/40	92.09	94.99	87.5/46.46/98.09/-/100/90	17/ 90.24/99.44/-/100/96.25
breast.D20.N699.C2 458/241	82.11	92.95	73.36/98.75	98.58/84.68
car.D25.N1728.C4 1210/ 384/69/65	91.03	80.79	98.67/ 73.43/66.66/78.46	92.25/58.74/46.03/23.67
congres.D34.N435.C2 267/168	88.96	95.19	89.13/88.69	97.36/92.31
cylBands.D124.N540.C2 228/312	68.7	68.33	30.7/92.94	61.99/79.99
dermatology.D49.N366.C6 72/112/61/ 52/49/20	77.86	80.8	80.55/82.14/62.29/ 78.84/79.59/85	80.65/88.86/67.71/77.94/ 96.67/46
glass.D48.N214.C7 70/76/ 17/0/13/9/29	72.89	64.1	80/68.42/ 23.52/-/76.92/100/86.2	54.49/65.71/0/-/45/30/ 90
heart.D52.N303.C5 164/ 55/36/35/13	55.44	55.03	81.7/ 21.81/19.44/34.28/23.07	78.68/14.86/ 23.26/23.79/10
hepatitis.D56.N155.C2 32/123	85.16	74.34	50/95.93	45.05/94.37
horsecolic.D85.N368.C2 232/136	81.25	81.57	81.46/80.88	85.69/76.74
iris.D19.N150.C3 50/50/50	95.33	95.33	100/94/92	100/91.57/96.57
nursery.D32.N12960.C5 4320/ 2/328/4266/4044	98.07	78.59	100/-/ 81.7/96.78/99.45	77.64/-/21.24/73.53/98.74
pima.D42.N768.C2 500/268	72.78	75.65	84.2/51.49	78.52/69.03
ticTacToe.D29.N958.C2 626/332	65.76	71.43	63.73/69.57	76.33/63
waveform.D101.N5000.C3 1657/1647/1696	77.94	70.66	59.56/88.4/85.73	72.87/69.13/71.67
wine.D68.N178.C3 59/71/48	95.5	88.03	96.61/94.36/95.83	85.38/87.26/94.67
Arithmetic Means	81.3	79.24	76.57	69.08

6.1 2-Class vs Multiclass Problem

2-class Problem. Five of the seven data sets where CPAR outperforms `fitcare` correspond to well-balanced 2-class problems, where the minimal positive cover constraint has to be loosened for one of the classes (see Sec. 5.2). On the two remaining 2-class data sets, which do not raise this issue (cylBands and hepatitis), `fitcare` has a better accuracy than CPAR.

Multiclass Problem. `fitcare` significantly outperforms `CPAR` on all the nine multiclass data sets but two – anneal and dermatology – on which `fitcare` lies slightly behind `CPAR`. On the nursery data, the improvement in terms of global accuracy even reaches 25% w.r.t. `CPAR`.

6.2 True Positive Rates in Minor Classes

When considering imbalanced data sets, True Positive rates (TP_r) are known to better evaluate classification performances. When focusing on the TP_r in the minor classes, `fitcare` clearly outperforms `CPAR` in 14 minor classes out of the 20 (bold values). Observe also that the 2-class data sets with a partial positive cover of the largest class have a poor global accuracy but the TP_r of the smallest classes often are greater than `CPAR`’s (see breast, horsecolic, ticTacToe).

Compared to `CPAR`, `fitcare` presents better arithmetic means in both the global and the per-class accuracies. However, the difference is much greater with the latter measure. Indeed, as detailed in Sec. 5.1, `fitcare` is driven by the minimization of the confusion between every pair of classes (whatever their sizes). As a consequence, `fitcare` optimizes the True Positive rates. In the opposite, `CPAR` (and all one-vs-all approaches), focusing only on the global accuracy, tends to over-classify in the major classes.

7 Conclusion

Association rules have been extensively studied along the past decade. The CBA proposal has been the first associative classification technique based on a “support-confidence” ranking criterion [2]. Since then, many other CBA-like approaches have been designed. Even if suitable for typical two-class problems, it appears that support and confidence constraints are inadequate for selecting rules in multiclass imbalanced training data sets. Other approaches (see, e.g., [11, 14, 12]) address the problem of imbalanced data sets but show their limits when considering more than 2 classes. We analyzed the limits of all these approaches, suggesting that a common weakness relies on their one-vs-all principle. We proposed a solution to these problems: our associative classification method extracts the so-called interesting class association rules w.r.t. a one-vs-each principle. It computes class association rules that are frequent in the positive class and infrequent in every other class taken separately (instead of their union). Tuning the large number of parameters required by this approach may appear as a bottleneck. Therefore, we designed an automatic tuning method that relies on a hill-climbing strategy. Empirical results have confirmed that our proposal is quite promising for multiclass imbalanced data sets.

Acknowledgments. This work is partly funded by EU contract IST-FET IQ FP6-516169 and by the French contract ANR ANR-07-MDCO-014 Bingo2. We would like to thank an anonymous reviewer for its useful concerns regarding the relevancy of our approach. Unfortunately, because of space restrictions, we could not address them all in this article.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining Association Rules Between Sets of Items in Large Databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, ACM Press (1993) 207–216
2. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: Proceedings of the Fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, AAAI Press (1998) 80–86
3. Bayardo, R., Agrawal, R.: Mining the Most Interesting Rules. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press (1999) 145–154
4. Li, W., Han, J., Pei, J.: CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In: Proceedings of the First IEEE International Conference on Data Mining, IEEE Computer Society (2001) 369–376
5. Boulicaut, J.F., Crémilleux, B.: Simplest Rules Characterizing Classes Generated by Delta-free Sets. In: Proceedings of the Twenty-Second Annual International Conference Knowledge Based Systems and Applied Artificial Intelligence, Springer (2002) 33–46
6. Yin, X., Han, J.: CPAR: Classification Based on Predictive Association Rules. In: Proceedings of the Third SIAM International Conference on Data Mining, SIAM (2003) 369–376
7. Baralis, E., Chiusano, S.: Essential Classification Rule Sets. *ACM Transactions on Database Systems* **29**(4) (2004) 635–674
8. Bouzouita, I., Elloumi, S., Yahia, S.B.: GARC: A New Associative Classification Approach. In: Proceedings of the Eight International Conference on Data Warehousing and Knowledge Discovery, Springer (2006) 554–565
9. Freitas, A.A.: Understanding the Crucial Differences Between Classification and Discovery of Association Rules – A Position Paper. *SIGKDD Explorations* **2**(1) (2000) 65–69
10. Wang, J., Karypis, G.: HARMONY: Efficiently Mining the Best Rules for Classification. In: Proceedings of the Fifth SIAM International Conference on Data Mining, SIAM (2005) 34–43
11. Arunasalam, B., Chawla, S.: CCCS: A Top-down Associative Classifier for Imbalanced Class Distribution. In: Proceedings of the Twelveth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press (2006) 517–522
12. Verhein, F., Chawla, S.: Using Significant Positively Associated and Relatively Class Correlated Rules for Associative Classification of Imbalanced Datasets. In: Proceedings of the Seventh IEEE International Conference on Data Mining, IEEE Computer Society 679–684
13. Dong, G., Li, J.: Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press (1999) 43–52
14. Ramamohanarao, K., Fan, H.: Patterns Based Classifiers. *World Wide Web* **10**(1) (2007) 71–83
15. Cerf, L., Gay, D., Selmaoui, N., Boulicaut, J.F.: Technical Notes on *fitcare*’s Implementation. Technical report, LIRIS (april 2008)
16. Coenen, F.: The LUCS-KDD software library (2004) <http://www.csc.liv.ac.uk/~frans/KDD/Software/>.
17. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI Repository of Machine Learning Databases (1998) <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

From Local to Global Patterns: a Co-Clustering Framework

Ruggero G. Pensa¹, Céline Robardet², Jean-François Boulicaut²

1: Pisa KDD Laboratory, ISTI-CNR, I-56124 Pisa, Italy

2: University of Lyon, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France

Abstract

Co-clustering or simultaneous clustering of rows and columns of a data matrix, enables to reveal interactions among elements of these two sets and appears as a promising conceptual clustering method. Applied on a 0/1 matrix, co-clustering algorithms find submatrices that are dense in 1 values. We propose a new formulation of the co-clustering task that consists in (a) computing locally strong associations between rows and columns, called bi-sets, and then (b) applying a classical clustering algorithm on such collections for delivering a co-clustering. Our framework has two main advantages. First of all, since co-clustering is based on a clustering algorithm, it is then possible to take the most from several decades of results in this area. Secondly, we can also benefit from the many recent methods that perform correct and complete extractions of local patterns that satisfy user-defined constraints. Indeed, it is shown that using these pattern increases the co-clustering quality. To validate this framework, we study in detail the instance CDK-MEANS that is based on the K-MEANS clustering algorithm extended toward overlapping via a user-defined threshold. We consider two different types of bi-sets: the formal concepts and a new pattern type called δ -bi-sets. We provide an experimental validation on many benchmark datasets and a real gene expression dataset. Not only we discuss the interestingness of the computed bi-partitions (i.e., using both an external and an internal quality measures on partitions), but also we consider a qualitative validation on a non trivial problem in functional genomics.

Key words: Data Mining, Co-Clustering, Pattern Discovery, Algorithms, bioinformatics

From Local to Global Patterns: a Co-Clustering Framework

Ruggero G. Pensa¹, Céline Robardet², Jean-François Boulicaut²

1: Pisa KDD Laboratory, ISTI-CNR, I-56124 Pisa, Italy

2: University of Lyon, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France

1. Introduction

Many data mining techniques have been designed to support knowledge discovery from categorical data that can be represented as Boolean matrices: the rows denote objects and the columns denote Boolean attributes that enable to record object properties as attribute-value pairs. For instance, given $\mathbf{r}$ in Table 1, object t_2 satisfies only properties g_2 and g_5 . Many application domains can provide such data, e.g., basket data analysis or WWW usage mining. Our motivating application domain concerns Boolean gene expression data analysis, i.e., knowledge discovery from datasets which encode gene expression properties (e.g., over-expression) in various biological situations (see, e.g., [25]). Given $\mathbf{r}$, we may say that, e.g., genes denoted by g_1, g_3, g_4 are considered over-expressed in situation t_1 .

Clustering is one of the major data mining tasks and it has been studied extensively, including for the special case of categorical or Boolean data. Its main goal is to identify a partition of objects and/or attributes such that an objective function which specifies its quality is optimized. Thanks to local search optimizations, many efficient algorithms can provide good partitions but suffer from the lack of explicit cluster characterization. It has motivated the research on conceptual clustering [12] and co-clustering¹ whose goal is to simultaneously cluster rows and columns of a data matrix to reveal interactions among elements of these two sets (see, e.g., [23] for a survey). Applied on a 0/1 matrix, such methods find submatrices dense in 1 values and each

¹Some authors use the term biclustering instead. We consider however that the term bi-cluster has been often used ambiguously and this will be discussed later.

Table 1: A Boolean context $\mathbf{r}$

	g_1	g_2	g_3	g_4	g_5
t_1	1	0	1	1	0
t_2	0	1	0	0	1
t_3	1	0	1	1	0
t_4	0	0	1	1	0
t_5	1	1	0	0	1
t_6	0	1	0	0	1
t_7	0	0	0	0	1

submatrix is called a bi-cluster. An example of an interesting bi-partition in $\mathbf{r}$ (Table 1) would be $\{\{\{t_1, t_3, t_4\}, \{g_1, g_3, g_4\}\}, \{\{t_2, t_5, t_6, t_7\}, \{g_2, g_5\}\}\}$. The first bi-cluster indicates that the characterization of objects from $\{t_1, t_3, t_4\}$ is that they almost always share properties from $\{g_1, g_3, g_4\}$. Also, properties in $\{g_2, g_5\}$ are characteristic of objects in $\{t_2, t_5, t_6, t_7\}$.

Clearly, the search space for bi-partitions is larger than for partitions. As a result, co-clustering methods need for clever local search optimization techniques to obtain high quality results.

In this paper, we propose to use local patterns to improve bi-partition quality. Local patterns can be considered as data point sets with unusually high density, relatively to some background model [17]. The a priori interestingness of local patterns can be declaratively specified by means of constraints (e.g., minimal frequency) that are such that efficient correct and complete solvers enable to compute them (see, e.g., [6]). For the co-clustering problem, we consider a category of local patterns, called bi-sets that are defined as a set of objects and a set of properties which satisfy some constraints. Such bi-sets contain valuable information that enable to control the search for a high quality bi-partition. For instance, formal concepts can be considered as relevant bi-sets formalized in [36]. A bi-set (T, G) is a formal concept if T is the largest set of objects and G the largest set of properties such

that every object of T satisfies every property from G^2 . These are maximal blocks of 1's in the sense that adding a column or a row results in the appearance of a zero in the block. Clearly, a formal concept reports on a locally strong association. Nevertheless, such a pattern is not a bi-cluster *stricto sensu* because its computation does not take into account the distribution of its elements (objects and properties) among the other bi-clusters as a co-clustering method does. In $\mathbf{r}$ (see Table 1), $(\{t_1, t_3\}, \{g_1, g_3, g_4\})$ is a formal concept. Other formal concepts are given in Table 2. Algorithms are now available to compute efficiently all formal concepts (see, e.g., [21]) or only those involving enough objects and properties: given thresholds γ and γ' , constraints like $|T| \geq \gamma \wedge |G| \geq \gamma'$ are enforced (see, e.g., [32, 4]).

Another issue concerns the shape of the bi-partition in terms of overlapping between bi-clusters. Our thesis is that a user-control on overlapping seems to be useful. On one hand, most of the known co-clustering algorithms compute collections of non overlapping bi-clusters. On another hand, many application domains would benefit from overlapping bi-clusters. This is the case in Boolean gene expression data analysis where bi-clusters point out sets of genes that tend to be co-expressed and sets of biological situations which seem to trigger this co-expression. Under the reasonable hypothesis that a biological function can putatively be assigned to each bi-cluster, discovering overlapping bi-clusters is important since we know that a same gene can be involved in different biological functions. Also, in a WWW usage mining context, bi-clusters of users associated to the resources they upload on some WWW sites can be used to discover interesting groups and, here again, one can get more relevant groups if overlapping is enabled. Given this context, the contribution of this paper is twofold.

- First we propose a new co-clustering framework. It enables to compute bi-partitions by grouping local patterns which capture locally strong associations between objects and properties.

Various local patterns are candidates for such a process. While being obviously non exhaustive, we may point out the popular frequent sets of properties associated to their supporting set of objects [1], error-tolerant frequent itemsets [38], closed sets or formal concepts (see, e.g., [24, 39, 32, 4]), large tiles [13], dense itemsets [30], support en-

²In other terms, a formal concept is built on a closed set of properties associated to its supporting set of objects.

velopes [31], dense and relevant bi-sets [3], or the many other different approaches for local pattern discovery from 0/1 data including subspace clustering approaches [40]. In the specific context of gene expression data analysis, several authors have considered the computation of such relevant local patterns by means of heuristic techniques (see, e.g., [18, 20])³. Deciding which type of local pattern is better as a starting point is out of the scope of this paper: the framework can be applied on any collection of bi-sets which capture a priori relevant associations.

- Secondly, we study one instance of our generic framework, the CDK-MEANS algorithm. It clusters simultaneously the objects and the properties of the data matrix applying the K-MEANS algorithm on the available collection of bi-sets. Finally, the obtained partition of bi-sets is translated into a bi-partition. In our experimental setting, we use formal concepts and the so-called δ -bi-sets as two representative types of local patterns. Our experimental validation based on many benchmark datasets reports on both external and internal quality measures on computed partitions. Comparisons between CDK-MEANS, two co-clustering algorithms (COCLUSTER [10] and BI-CLUST [29]), and two classical clustering algorithms (K-MEANS and EM) have been performed. It confirms the added-value of CDK-MEANS and our generic framework.

Let us however emphasize that we do not claim that CDK-MEANS is better than any other clustering or co-clustering algorithm. Our contribution is to demonstrate that the local to global approach to clustering can work and that it opens interesting directions of research for new co-clustering approaches. Indeed, this framework has been recently exploited for considering constraint-based co-clustering. For instance, given extended must-link or cannot-link constraints that enforce objects and/or properties to be together or not within some bi-clusters, it is possible to push some of the desired constraints on the final bi-partition at the level of the starting local patterns [26]. Notice that a preliminary version of the framework has been sketched in [27]. The rest of the paper is organized as follows. In Section 2, we set up our clustering framework and we introduce its CDK-MEANS instance. Section 3

³These authors consider that the term bi-cluster should refer to a local pattern while most of the authors, including us, consider that the term cluster refers to a global property.

considers two types of local patterns which can be clustered by means of CDK-MEANS. Section 4 discusses our experimental validation methodology and it contains many experimental results, including a discussion on scalability issues and an application to gene expression data. Section 5 presents previous work on co-clustering methods. Section 6 is a short conclusion.

2. Clustering model

We define a new co-clustering model on categorical data which can be put under the form of (possibly huge) Boolean matrices. Intuitively, given the data and a collection of bi-sets which capture some locally strong associations within the data, this framework enables to build a partition of K clusters of bi-sets and thus provides a collection of possibly overlapping bi-clusters of objects and properties. The principles are illustrated within a K-MEANS-like algorithm over bi-sets. Specific instances on some well-defined classes of bi-sets are discussed in Section 3.

2.1. Problem setting

Assume a set of objects $\mathcal{O} = \{t_1, \dots, t_m\}$ and a set of Boolean properties $\mathcal{P} = \{g_1, \dots, g_n\}$. The relation to be mined is $\mathbf{r} \subseteq \mathcal{O} \times \mathcal{P}$, which can be represented by a Boolean 0/1 matrix where $r_{ij} = 1$ if property g_j is satisfied by object t_i . A co-clustering is defined as follows:

Definition 2.1 (Co-clustering). *A co-clustering method provides three elements:*

1. *a partition $C^o = \{C_1^o \dots C_K^o\}$ of K clusters of objects,*
2. *a partition $C^p = \{C_1^p \dots C_K^p\}$ of K clusters of properties*
3. *a bijective function f from C^o to C^p that associates to each cluster of C^o , a cluster of C^p .*

Note that overlapping can be authorized between clusters of each partition.

Our idea is that the two partitions of a co-clustering can be obtained by applying a clustering algorithm on a set of bi-sets. Indeed, a partition of bi-sets gathers objects and properties in a single cluster. The objects can be considered as a cluster of C^o and the properties as a cluster of C^p and the function f associates these two clusters in a one-to-one mapping. Let us first defined bi-sets and introduce some notations.

Definition 2.2 (Bi-set). A bi-set is a couple of sets denoted by $b_j = (T_j, G_j)$ such that $T_j \subseteq \mathcal{O}$ and $G_j \subseteq \mathcal{P}$. b_j can be represented by characteristic vectors $\langle \mathbf{t}_j \rangle = \langle t_{j1}, \dots, t_{jm} \rangle$ and $\langle \mathbf{g}_j \rangle = \langle g_{j1}, \dots, g_{jn} \rangle$ such that $t_{jk} = 1$ if $t_k \in T_j$ (0 otherwise) and $g_{jk} = 1$ if $g_k \in G_j$ (0 otherwise).

In our co-clustering framework, we first compute from $\mathbf{r}$ a set of a priori interesting bi-sets denoted $\mathcal{B}$. We then apply a clustering algorithm such a collection. We will see in Section 2.2 how to cluster bi-sets.

The clustering algorithm provides a partition C made of K clusters of bi-sets $\{C_1, \dots, C_K\}$ such that $C_i \subseteq \mathcal{B}$. Post-processing such a partition into a co-clustering requires to adapt the centroid definition to clusters of bi-sets but also to have a distance between bi-sets and centroids.

Definition 2.3 (Bi-set cluster centroid). The centroid μ_i of a cluster C_i of bi-sets is defined by two real valued vectors $\langle \tau_i \rangle = \langle \tau_{i1}, \dots, \tau_{im} \rangle$ and $\langle \gamma_i \rangle = \langle \gamma_{i1}, \dots, \gamma_{in} \rangle$ where τ and γ are the averages from the characteristic vectors associated to bi-sets that belongs to C_i :

$$\tau_{ik} = \frac{1}{|C_i|} \sum_{b_j \in C_i} t_{jk}, \quad \gamma_{ik} = \frac{1}{|C_i|} \sum_{b_j \in C_i} g_{jk}$$

$|C_i|$ is the cardinal of C_i . The centroid values depend on the number of bi-sets involving each object/property. This makes our approach substantially different from any other co-clustering approach, including approaches dedicated to binary and categorical data.

Definition 2.4 (Distance between a bi-set and a centroid). A distance between a bi-set and a centroid is also needed. Let us first formally define the weighted cardinality of intersection and union between a centroid vector and a set of objects that we use:

$$|\mathbf{t}_j \cap \boldsymbol{\tau}_i| = \sum_{k=1}^m a_k \frac{t_{jk} + \tau_{ik}}{2}, \quad |\mathbf{t}_j \cup \boldsymbol{\tau}_i| = \sum_{k=1}^m \frac{t_{jk} + \tau_{ik}}{2}$$

where $a_k = 1$ if $t_{jk} \cdot \tau_{ik} \neq 0$, 0 otherwise. These quantities are similarly defined for properties.

The distance we use is based on the symmetrical difference of two sets and is defined by:

$$d(b_j, \mu_i) = \frac{1}{2} \left(\frac{|\mathbf{t}_j \cup \boldsymbol{\tau}_i| - |\mathbf{t}_j \cap \boldsymbol{\tau}_i|}{|\mathbf{t}_j \cup \boldsymbol{\tau}_i|} + \frac{|\mathbf{g}_j \cup \boldsymbol{\gamma}_i| - |\mathbf{g}_j \cap \boldsymbol{\gamma}_i|}{|\mathbf{g}_j \cup \boldsymbol{\gamma}_i|} \right)$$

It is the mean of the weighted symmetrical differences of the set components of the bi-set and those of the centroid. Intuitively, the intersection is equal to the mean between the number of common objects and the sum of their centroid weights. The union is the mean between the number of objects and the sum of their centroid weights. The weighted size of the intersection and union for properties can be defined similarly.

We can now formally defined how the partition C is used to produce a co-clustering.

Definition 2.5 (Transformation of C into a co-clustering). *Considering the obtained clustering of bi-sets $C = \{C_1, \dots, C_K\}$ and its associated centroids $\{\mu_1 = (\langle \tau_1 \rangle, \langle \gamma_1 \rangle), \dots, \mu_K = (\langle \tau_K \rangle, \langle \gamma_K \rangle)\}$ the two partitions of the co-clustering as derived as follows:*

$$C_i^o = \{t_j \in \mathcal{O} \mid i = \operatorname{argmax}_{k \in \{1 \dots K\}} \tau_{kj}\}$$

$$C_i^p = \{g_j \in \mathcal{P} \mid i = \operatorname{argmax}_{k \in \{1 \dots K\}} \gamma_{kj}\}$$

We stress the fact that the values of each component of τ_i and γ_i depend exclusively on the number of bi-sets involved in the centroid.

The bijective function f of this co-clustering is defined by:

$$\forall i = 1 \dots K, f(C_i^o) = C_i^p$$

Example 1. *The Boolean vectors corresponding to a collection of bi-sets⁴ contained in $\mathbf{r}$ (see Table 1) are given in Table 2. Formal concepts b_1 and b_8 are respectively the supremum and the infimum subsets in the concept lattice related to the collection. A possible solution for $K = 2$ is given by the two clusters of bi-sets $C_1 = \{b_1, b_3, b_4, b_6, b_8\}$ and $C_2 = \{b_2, b_5, b_7\}$. The Boolean vectors μ_1 and μ_2 corresponding to these two clusters are given in Table 3.*

Then the co-clustering is derived from these vectors. For instance, the object t_1 is assigned to C_1^o , since $\max\{\tau_{11}, \tau_{21}\} = \tau_{11} = 0.80$. Analogously, property g_1 is assigned to C_1^p , since $\max\{\gamma_{11}, \gamma_{21}\} = \gamma_{11} = 0.60$. The final co-clustering is thus

$$\begin{aligned} C^o &= \{\{t_1, t_3, t_4\}, \{t_2, t_5, t_6, t_7\}\} \\ C^p &= \{\{g_1, g_3, g_4\}, \{g_2, g_5\}\} \end{aligned}$$

⁴We consider here that the extracted bi-sets are formal concepts (see Section 3.1)

Table 2: List of vectors corresponding to the 8 formal concepts in Table 1

b_i	$\langle t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7} \rangle$	,	$\langle g_{i1}, g_{i2}, g_{i3}, g_{i4}, g_{i5} \rangle$
b_1	$\langle 0, 0, 0, 0, 0, 0, 0 \rangle$	,	$\langle 1, 1, 1, 1, 1 \rangle$
b_2	$\langle 0, 0, 0, 0, 1, 0, 0 \rangle$	,	$\langle 1, 1, 0, 0, 1 \rangle$
b_3	$\langle 1, 0, 1, 0, 0, 0, 0 \rangle$	,	$\langle 1, 0, 1, 1, 0 \rangle$
b_4	$\langle 1, 0, 1, 0, 1, 0, 0 \rangle$	,	$\langle 1, 0, 0, 0, 0 \rangle$
b_5	$\langle 0, 1, 0, 0, 1, 1, 0 \rangle$	,	$\langle 0, 1, 0, 0, 1 \rangle$
b_6	$\langle 1, 0, 1, 1, 0, 0, 0 \rangle$	,	$\langle 0, 0, 1, 1, 0 \rangle$
b_7	$\langle 0, 1, 0, 0, 1, 1, 1 \rangle$	,	$\langle 0, 0, 0, 0, 1 \rangle$
b_8	$\langle 1, 1, 1, 1, 1, 1, 1 \rangle$	,	$\langle 0, 0, 0, 0, 0 \rangle$

Table 3: List of vectors corresponding to two possible clusters of formal concepts

μ_i	$\langle \tau_{i1}, \tau_{i2}, \tau_{i3}, \tau_{i4}, \tau_{i5}, \tau_{i6}, \tau_{i7} \rangle$	,	$\langle \gamma_{i1}, \gamma_{i2}, \gamma_{i3}, \gamma_{i4}, \gamma_{i5} \rangle$
μ_1	$\langle 0.80, 0.20, 0.80, 0.40, 0.40, 0.20, 0.20 \rangle$	,	$\langle 0.60, 0.20, 0.60, 0.60, 0.20 \rangle$
μ_2	$\langle 0.00, 0.67, 0.00, 0.00, 1.00, 0.67, 0.33 \rangle$	,	$\langle 0.33, 0.67, 0.00, 0.00, 1.00 \rangle$

It is now interesting to discuss the control on bi-cluster overlapping and its relationship with fuzzy clustering. We can enable that a number of objects and/or properties belong to more than one cluster by controlling the size of the overlapping part of each cluster. Our definition of centroid, and the way we compute them, intrinsically provides overlapping bi-clusters. Thus, we compute potentially overlapping bi-clusters, and the overlapping level is controlled by the user-defined coefficients of each object/property in each centroid.

Thanks to our definition of cluster membership determined by the values of τ_i and γ_i , we just need to adapt the cluster assignment step.

Definition 2.6 (Transformation of C into a co-clustering). *Let δ_o and δ_p be two real values in $[0, 1]$ that quantify the membership of each element to a cluster. Using these parameter, we defined the fuzzy overlapping clustering by*

$$\begin{aligned} C_i^o &= \{t_j \in \mathcal{O} \mid \tau_{ij} \geq (1 - \delta_o) \cdot \max_i(\tau_{ij})\} \\ C_i^p &= \{g_j \in \mathcal{P} \mid \gamma_{ij} \geq (1 - \delta_p) \cdot \max_i(\gamma_{ij})\} \end{aligned}$$

Thus, C^o and C^p are no more partitions, but two sets of overlapping sets.

The number of overlapping objects (resp., properties) depends on the distribution of the values of τ_i (resp. γ_i). Notice that if overlapping is enabled, $\delta = 0$ does not imply that each object and each property is assigned to a single cluster (i.e., non overlapping is not guaranteed). The choice of a relevant value for δ is clearly application-dependent. When a co-clustering structure holds in the data, little values of δ are not enough to provide relevant overlapping. On another hand, in noisy contexts, even little values of δ can give rise to significant overlapping zones. Interestingly, as we consider centroid values as membership coefficients, we also embed fuzzy clustering [5]. Indeed, we can easily derive fuzzy bi-clusters μ_i^f starting from the Boolean vectors μ_i , by dividing each component by the sum of the values taken by the component in the K clusters:

$$\mu_i^f = \left\langle \frac{\tau_{i1}}{\Theta_1}, \dots, \frac{\tau_{ij}}{\Theta_j}, \dots, \frac{\tau_{im}}{\Theta_m} \right\rangle, \left\langle \frac{\gamma_{i1}}{\Gamma_1}, \dots, \frac{\gamma_{ij}}{\Gamma_j}, \dots, \frac{\gamma_{in}}{\Gamma_n} \right\rangle$$

where

$$\Theta_j = \sum_{i=1}^K \tau_{ij}, \quad \Gamma_j = \sum_{i=1}^K \gamma_{ij}$$

Table 4: List of vectors for two possible bi-clusters

μ_i^f	$\langle \tau_{i1}^f, \tau_{i2}^f, \tau_{i3}^f, \tau_{i4}^f, \tau_{i5}^f, \tau_{i6}^f, \tau_{i7}^f \rangle$	,	$\langle \gamma_{i1}^f, \gamma_{i2}^f, \gamma_{i3}^f, \gamma_{i4}^f, \gamma_{i5}^f \rangle$
μ_1^f	$\langle 1.00, 0.23, 1.00, 1.00, 0.29, 0.23, 0.37 \rangle$	,	$\langle 0.64, 0.23, 1.00, 1.00, 0.17 \rangle$
μ_2^f	$\langle 0.00, 0.77, 0.00, 0.00, 0.71, 0.77, 0.63 \rangle$	,	$\langle 0.36, 0.77, 0.00, 0.00, 0.83 \rangle$

Example 2. In our toy example from Table 1, if we authorize overlapping on objects with $\delta_o = 0.4$, then object t_7 , is assigned to C_1^o too, since $\tau_{17} \geq (1 - 0.4) \cdot \max\{\tau_{17}, \tau_{27}\} = 0.2$ (see Table 3). As a result, the first bi-cluster becomes

$$(\{t_1, t_3, t_4, t_7\}, \{g_1, g_3, g_4\})$$

The two Boolean vectors μ_1 and μ_2 can be considered as a final solution if we consider the case of fuzzy co-clustering. By applying the simple post-processing described earlier, we get the two fuzzy bi-clusters from Table 4.

2.2. CDK-MEANS algorithm

Let us now introduce the instance CDK-MEANS sketched in Table 5. It computes a bi-partition of a dataset $\mathbf{r}$ given a collection of bi-sets $\mathcal{B}$ extracted from $\mathbf{r}$ beforehand, the desired number of clusters K , the threshold values for δ_o and δ_p , and a maximum number of iterations MI . On our running example, CDK-MEANS provides the example bi-partition given in Section 1. Being a K-MEANS-like algorithm, scalability issues w.r.t. the clustering process are well understood. The complexity is linear in the number of bi-sets in the collection $\mathcal{B}$ and scalability issues w.r.t. the computation of a collection $\mathcal{B}$ for various types of bi-sets (e.g., formal concepts or δ -bi-sets are discussed in Section 3.

The algorithm starts (1) by considering an initial set of K centroids. In our implementation, this initialization is randomly obtained, but it could be also expert-driven or semi-supervised. Moreover, in the case of random initialization, the initial centroids can be built by randomly selecting a set of properties and a set of objects for each centroid, or, by joining a set of randomly selected bi-sets for each centroid (in such a case, overlapping is introduced at the initialization step). Once the initial centroids have been chosen, the algorithm scans the collection of bi-sets and assigns each bi-set

Table 5: CDK-MEANS pseudo-code

CDK-MEANS ($\mathbf{r}$ is a Boolean context, $\mathcal{B}$ is a collection of bi-sets in $\mathbf{r}$, K is the number of clusters, MI is the maximal iteration number, δ_o and δ_p are thresholds values for controlling overlapping) 1. Let $\mu_1 \dots \mu_K$ be the initial cluster centroids. $k := 0$. 2. Repeat (a) For each bi-set $b \in \mathcal{B}$, assign it to cluster C_i s.t. $d(b, \mu_i)$ is minimal. (b) For each cluster C_i, compute τ_i and γ_i. (c) $k := k + 1$. 3. Until centroids are unchanged or $k = MI$. 4. If overlap enabled, $\forall t_j \in \mathcal{O}$ (resp. $g_j \in \mathcal{P}$), assign it to each cluster C_i^o (resp. C_i^p) s.t. $\tau_{ij} \geq (1 - \delta_o) \cdot \max_i(\tau_{ij})$ (resp. $\gamma_{ij} \geq (1 - \delta_p) \cdot \max_i(\gamma_{ij})$). 5. Else, $\forall t_j \in \mathcal{O}$ (resp. $g_i \in \mathcal{P}$), assign it to the first cluster C_i^o (resp. C_i^p) s.t. τ_{ij} (resp. γ_{ij}) is maximal. 6. Return $\{C_1^o \dots C_K^o\}$ and $\{C_1^p \dots C_K^p\}$

to the cluster for which the distance (computed as described in Section 2.1) is minimal. Then, for each cluster, a new centroid is generated by joining the bi-sets assigned to such a cluster (see Section 2.1). Lines 2a and 2b are executed until all the K centroids are unchanged, or the maximal number of iteration MI is reached.

The second part of the algorithm is a simple post-processing of the computed centroids to assign each object and property to one (or more) bi-cluster. When cluster overlapping is enabled (Line 4), the algorithm assigns each object and property to the bi-clusters for which the membership value in the centroid is greater than $(1-\delta)$ times the maximal value. When disjoint bi-clusters are desired (Line 5), the algorithm assigns each property and object to one of the bi-clusters for which the membership value is maximal. In this case, when more than one assignment is possible, different selection criteria might be adopted. In our implementation, we perform an arbitrary choice by assigning each property and object to the first bi-cluster with the maximal membership value.

2.3. Complexity issues

Our instance CDK-MEANS is a rather straightforward adaptation of K-MEANS plus a simple post-processing phase. For the post-processing step (Lines 4-6), the algorithm scans the list of clusters $m + n$ times (m is the number of objects, and n is the number of properties) to get the highest membership coefficient for each object and property. The complexity for this step is then $O(K \cdot (m + n))$. When $K \ll (m + n)$ the algorithm runs in linear time.

The standard complexity for each iteration of a K-MEANS algorithm is $O(K \cdot m \cdot n)$ where K is the number of clusters. Usually $n \ll m$, and $K \ll m$, than the K-MEANS algorithm is said to run in linear time. The important point here is that CDK-MEANS does not manipulate objects, but bi-sets, and that the distances have to be computed on large vectors. Then, the complexity for each iteration is $O(K \cdot N \cdot (m + n))$, where N is the number of bi-sets (i.e., $|\mathcal{B}|$). We can have much more bi-sets than objects or properties in the Boolean data. Since the quality of the extracted bi-partitions relies partly on the relevancy of the extracted bi-sets (i.e., the better they capture locally relevant associations between sets of objects and properties, the better the co-clustering might be relevant), it remains possible to reduce the size of the considered collection of bi-sets provided that relevant associations are

preserved. Examples of sensible reductions can be to remove too small bi-sets or even to apply an interpretation phase to remove irrelevant bi-sets according to some available background knowledge (or some statistical tests). Reducing the bi-set collection size speeds up the centroid construction step of CDK-MEANS. We show in Section 4 that the quality of the extracted bi-partitions can be also improved when the selected bi-sets appear more relevant in terms of captured associations. Minimal size constraints can be used to avoid small patterns (e.g., typically, patterns which might be due to noisy 1 values). Moreover, fault-tolerant patterns (see, e.g., [38, 30, 3]) can be preferred to avoid patterns that are due to noisy 0 values. Notice also that relevancy can be enforced by means of other user-defined constraints that would integrate domain knowledge during the preprocessing phase of the co-clustering, e.g., selecting those bi-sets where a given subset of properties and/or objects occur). To summarize, even though we have a classical trade-off between computational complexity and end result relevancy, it is expected that selecting bi-sets based on their a priori relevancy can improve both computational time and the quality of the computed bi-partitions.

3. Selecting a bi-set type

So far, we assumed that relevant collections of bi-sets have captured beforehand locally strong associations between sets of objects and properties. We now consider two types of bi-sets which can be used for this purpose. The first one comes from the well-studied framework of Formal Concept Analysis [36]. The second one is related to fault-tolerant pattern mining and it is based on δ -free-sets, i.e., one of the few approximate condensed representations of frequent sets [7]. The relationship between these pattern types and other relevant types is discussed further in Section 5.

3.1. Using formal concepts

Formal concepts are used since more than two decades in artificial intelligence [36]. The so-called Formal Concept Analysis is based on a lattice structure over a collection of (local) patterns. Moreover, when trying to tackle the popular association rule mining task [1] in dense and highly-correlated data, researchers have studied (frequent) closed set mining. Since formal concepts are based on such closed sets, we have an interesting cross-fertilization of the two communities: the Formal Concept Analysis community has provided interesting theoretical results on the so-called concept lattices while the data

mining community has been studying in depth scaling issues when processing huge Boolean datasets.

Definition 3.1 (Formal concepts). *If $T \subseteq \mathcal{O}$ and $G \subseteq \mathcal{P}$, let $\phi(T, \mathbf{r}) = \{g \in \mathcal{P} \mid \forall t \in T, (t, g) \in \mathbf{r}\}$ and $\psi(G, \mathbf{r}) = \{t \in \mathcal{O} \mid \forall g \in G, (t, g) \in \mathbf{r}\}$. A bi-set (T, G) is a formal concept in $\mathbf{r}$ when $T = \psi(G, \mathbf{r})$ and $G = \phi(T, \mathbf{r})$. By construction, G and T are closed sets, i.e., $G = \phi \circ \psi(G, \mathbf{r})$ and $T = \psi \circ \phi(T, \mathbf{r})$. Intuitively, (T, G) is a maximal rectangle of true values modulo arbitrary permutations of rows and columns.*

$(\{t_1, t_3\}, \{g_1, g_3, g_4\})$, $(\{t_1, t_3, t_4\}, \{g_3, g_4\})$, and $(\{t_5, t_6\}, \{g_2, g_5\})$ are examples of formal concepts among the 8 ones which hold in $\mathbf{r}$ (see Table 1). Many algorithms have been developed that can extract complete collections of formal concepts (see, e.g., [21] for a survey), or only those involving enough objects and/or properties: constraints like $|T| \geq \gamma$ (see the many papers about γ -frequent closed set mining and, e.g., [14] for a survey), $|G| \geq \gamma$ (see again frequent closed sets on transposed matrices), or $|T| \wedge |G| \geq \gamma$ are enforced. Mining frequent closed sets on one dimension can be done efficiently: Given thresholds γ and γ' , we can compute efficiently every set T of objects (resp. set G of properties) such that $T = \psi \circ \phi(T, \mathbf{r}) \wedge |G| \geq \gamma'$ where $G = \phi(T, \mathbf{r})$ (resp. $G = \phi \circ \psi(G, \mathbf{r}) \wedge |T| \geq \gamma$ where $T = \psi(G, \mathbf{r})$). In both cases, the bi-set (T, G) is a formal concept. Exploiting size constraints on both dimensions simultaneously is harder but this has been studied within the D-MINER algorithm [4].

A fundamental problem when considering collections of formal concepts from real-life datasets, is that the Galois connection (ϕ, ψ) is roughly speaking “too strong”: we have to capture every maximal set of objects and its maximal set of associated properties. As a result, the number of formal concepts even in small matrices can be huge. It can be exponential in the smallest dimension of the Boolean matrix. One alternative is to consider extensions of formal concepts towards fault-tolerance. Among the available proposals, we select the so-called δ -bi-sets for further studies within our co-clustering approach.

3.2. Using δ -bi-sets

The idea of δ -bi-set comes from some previous work on approximate condensed representations for frequent sets [7]. δ -free sets are well-specified sets whose counted frequencies enable to infer the frequency of many sets (sets

included in their δ -closures) without further counting but with a bounded error. When $\delta = 0$, the δ -closure on a 0-free set X (also called a key pattern in [2]) is the classical closure and we can infer the exact frequency of every superset of X which is a subset of its closure, i.e., a closed set. Our idea is now to consider bi-sets built on δ -free sets. When $\delta = 0$, we get exactly formal concepts, otherwise we get bi-sets which can be extracted efficiently (δ -freeness is an anti-monotonic property) which still capture strong associations, i.e., a bounded number of exceptions can occur on columns. Providing details on δ -freeness and δ -closures is beyond the objective of this paper (see [7] for details). We just give here an intuitive definition of these notions. A set $Y \subseteq \mathcal{P}$ is δ -free for a positive integer δ if its absolute frequency in $\mathbf{r}$ differs from the frequency of all its strict subsets by at least $\delta + 1$.

The δ -closure of a set $Y \subseteq \mathcal{P}$ is the superset Z of Y such that every added property ($\in Z \setminus Y$) is almost always true for the objects which satisfy the properties from Y : at most δ false values are enabled.

For example, in Table 1, the 1-free itemsets are $\{g_1\}$, $\{g_2\}$, $\{g_3\}$, $\{g_4\}$, $\{g_5\}$, $\{g_1, g_2\}$, and $\{g_1, g_5\}$. An example of 1-closure for $\{g_1\}$ is $\{g_1, g_3, g_4\}$.

It is possible to consider bi-sets which can be built on δ -free sets and their δ -closures on one hand, on the sets of objects which support the δ -free set on the properties on the other hand.

Definition 3.2 (δ -bi-set). *A bi-set (T, G) in $\mathbf{r}$ is a δ -bi-set iff G can be decomposed into $G = X \cup Y$ such that X is a δ -free set in $\mathbf{r}$, Y is its associated δ -closure and $T = \{t \in \mathcal{O} \mid \forall x \in X, (t, x) \in \mathbf{r}\}$.*

In Table 1, the 1-bi-sets derived from the 1-free-sets $\{g_3\}$ and $\{g_5\}$ are $(\{t_1, t_3, t_4\}, \{g_1, g_3, g_4\})$ and $(\{t_2, t_5, t_6, t_7\}, \{g_2, g_5\})$. When $\delta \ll |T|$, δ -bi-sets are dense bi-sets with a small and bounded number of exceptions per column.

Notice that the 0-closure is the classical closure operator. Looking for a 0-free-set, say X , and its 0-closure, say Y , provides the closed set $X \cup Y$ and thus the formal concept $(\psi(X \cup Y, \mathbf{r}), X \cup Y)$. The added-value w.r.t. formal concepts when $\delta > 0$ has been empirically evaluated in [28].

For the experiments, we have been using a straightforward extension of the implementation described in [7], which can exploit the anti-monotonicity of δ -freeness and minimal frequency constraints. Indeed, we just added the automatic generation of the supporting set for each extracted δ -free-set.

4. Experimental validation

Let us report on different sets of experiments to evaluate our approach w.r.t. other clustering and co-clustering methods. First we compare the quality performances of our algorithm w.r.t. to two existing and recent approaches, using formal concepts and δ -bi-sets on different benchmark datasets. We show that our approach is competitive with these algorithms, and that, in some cases, it outperforms them. We also compare our algorithm with well-known one-dimensional approaches. Second, we give an analysis of the convergence of our algorithm, and discuss some important features about scalability. The next section is dedicated to an application on a gene expression dataset. It addresses the computation of overlapping bi-clusters as well.

4.1. Evaluation method

Different techniques can be used to evaluate the quality of a partition (see [16] for a survey). An external criterion consists in comparing the computed partition with a “correct” one. It means that data instances are already associated to some correct labels and that one quantifies the agreement between computed labels and correct ones. A popular measure is the Jaccard coefficient [19], which measures the agreement between two partitions of m elements. If $\mathbf{C} = \{C_1 \dots C_s\}$ is our clustering structure and $\mathbf{P} = \{P_1 \dots P_t\}$ is a predefined partition, each pair of data points is either assigned to the same cluster in both partitions or to different ones. Let a be the number of pairs belonging to the same cluster of $\mathbf{C}$ and to the same cluster of $\mathbf{P}$. Let b be the number of pairs whose points belong to different clusters of $\mathbf{C}$ and to different clusters of $\mathbf{P}$. The agreement between $\mathbf{C}$ and $\mathbf{P}$ can be estimated using

$$Jaccard(\mathbf{C}, \mathbf{P}) = \frac{a}{m \cdot (m - 1) / 2 - b}$$

which takes values between 0 and 1 and is maximized when $s = t$.

We also want to evaluate the quality of our co-clustering using an internal criterion. An interesting measure for this purpose is the symmetrical Goodman and Kruskal’s τ coefficient [15]. It is evaluated in a co-occurrence table $\mathbf{p}$ and it discriminates well bi-partitions w.r.t. the intensity of the functional link between both partitions. In [29], it is shown that this coefficient is quite interesting given that it does not depend on the number of bi-clusters. Therefore, we decided to use it for our experiments. Let p_{ij} be the frequency

of relations between an object of a cluster C_i^o and a property of a cluster C_j^p , and $p_i = \sum_j p_{ij}$ and $p_j = \sum_i p_{ij}$. We use the τ_S coefficient which evaluates the proportional reduction in error given by the knowledge of C^o on the prediction of C^p and viceversa. It is defined as follows:

$$\tau_S = \frac{\frac{1}{2} \sum_i \sum_j (p_{ij} - p_i p_j)^2 \frac{p_i + p_j}{p_i p_j}}{1 - \frac{1}{2} \sum_i p_i^2 - \frac{1}{2} \sum_j p_j^2}$$

4.2. Experiments on benchmark datasets

We report on experiments using eight well-known datasets. Seven of them (**voting-records**, **iris**, **zoo**, **breast-w**, **credit-a**, **internet-ads**, **mushroom**) are taken from the UCI ML Repository⁵. The last one (**titanic**) comes from the JSE Data Archive of the American Statistical Association⁶. All the experiments have been performed on a PC with 1 Gb RAM and a 3.0 GHz Pentium 4 processor running Linux OS.

We have performed each experiment on three collections of bi-sets. First, we have used collections of formal concepts and then collections of δ -bi-sets with $\delta = 1$ and $\delta = 2$. Moreover, we have performed several experiments on **internet-ads** to illustrate the impact of a selection phase over collections of bi-sets. We have also studied the dynamic behavior of CDK-Means by computing the minimal, maximal and average number of iterations for each dataset. The results concerning the quality w.r.t. the Goodman-Kruskal's coefficient are in Table 6 (for formal concepts) and in Table 13 and Table 14 (for δ -bi-sets). The Jaccard coefficients corresponding to the class variable are in Table 7 (for formal concepts), and in Table 13 and Table 14 (for δ -bi-sets). The results in terms of number of iterations for the different formal concept collections are in Table 9. For δ -bi-sets, results are in Table 11 (for $\delta = 1$) and in Table 12 (for $\delta = 2$). Finally, the scalability analysis is summarized in Table 10.

4.2.1. Experimental results when using formal concepts

Without considering the class variable, we have first processed each dataset with D-MINER [4] and without using additional constraints (except for **voting-records**, **mushroom** and **credit-a** which clearly need size constraints to get

⁵<http://www.ics.uci.edu/~mlearn/MLRepository.html>

⁶http://www.amstat.org/publications/jse/jse_data_archive.html

Table 6: Goodman-Kruskal’s coefficient values for different co-clustering algorithms (mr-2 and mr-5 refer to mushroom with 2 and 5 clusters).

	BI-CLUST	COCLUSTER		CDK-MEANS	
Dataset	Max	Max	Mean	Max	Mean
voting	0.320	0.314	0.308±0.008	0.311	0.311±0.000
titanic	0.332	0.321	0.226±0.076	0.363	0.215±0.118
iris-2	0.544	0.544	0.357±0.195	0.544	0.422±0.159
iris-3	0.544	0.471	0.285±0.141	0.544	0.329±0.085
zoo-2	0.191	0.186	0.157±0.034	0.198	0.165±0.024
zoo-7	-	0.106	0.102±0.006	0.110	0.063±0.014
breast-w	0.507	0.507	0.413±0.196	0.498	0.498±0.000
credit-3	0.104	0.019	0.007±0.005	0.079	0.066±0.008
credit-2	-	0.013	0.003±0.003	0.096	0.086±0.023
mr-2	-	0.198	0.158±0.026	0.176	0.157±0.017
mr-5	0.187	0.142	0.111±0.010	0.118	0.096±0.011
ads	-	0.006	0.003±0.001	0.661	0.158±0.113

smaller but still relevant collections). It has given complete (w.r.t. specified constraints when any) collections of formal concepts for running CDK-MEANS.

We compared CDK-MEANS bi-partitions with those obtained by COCLUSTER [10], and BI-CLUST [29]. As the initialization of these algorithms is randomized, we executed them 100 times on each dataset and computed the average and maximum value of the Goodman-Kruskal’s coefficients. The number of desired clusters for each experiment has been set to the number of class variable values, except for BI-CLUST which automatically determines the number of clusters. The available BI-CLUST prototype has not been able to process *internet-ads* (more than 1500 properties). We summarize these results in Table 6. Notice that, when CDK-MEANS has the worst results, the Goodman-Kruskal’s coefficient is not significantly dissimilar from the other algorithm coefficients. On the other hand, for *internet-ads*, the coefficient obtained with CDK-MEANS is considerably higher than the one obtained with

COCLUSTER. This is due to the high dimension of the dataset which is not well handled by the other algorithms. Also the average behavior is similar to the one of COCLUSTER. The average values but also the standard deviation values of the two algorithms are often similar. Notice that for **voting-records** and **breast-w**, CDK-MEANS has always computed the same bi-partition.

CDK-MEANS generally needs for more execution time than the other algorithms because it processes possibly large collections of bi-sets. In these benchmarks, the extraction of formal concepts by itself is not that expensive (from 1 to 20 seconds). Using minimal size constraints during the formal concept extraction phase enables to reduce the collection size and it will be discussed later. For **titanic**, **iris**, and **zoo**, CDK-MEANS performs in less than one second, while for **breast-w**, **credit-a** and **internet-ads**, the average execution time is less than one minute. For **mushroom**, the average execution time is about seven minutes since more than 50 000 formal concepts have to be processed and the maximum size of centroids is relatively high (more than 8 200 elements).

In a second bunch of experiments, we have also used the Jaccard index to compare the agreement of the object partitions with those determined by the class variables. We provide the comparisons in Table 7. Again, our algorithm is competitive w.r.t. the other co-clustering methods. With the exception of the **breast-w** dataset, our algorithm always performs as or better than **Bi-CLUST** and **COCLUSTER**, and the average behavior is similar to the one of **COCLUSTER**.

Finally, we have compared our results, with those obtained by applying two classical clustering algorithms, the WEKA implementations of **K-MEANS** and **EM** [37] (see Table 8). Except for **breast-w** and **zoo-7**, our algorithm is competitive w.r.t the other ones. For most datasets, CDK-MEANS performs better than standard **K-MEANS** and **EM**. The average behavior is worse in general, but means and standard deviations are not very distant from those obtained by the other two algorithms. These results show that our clustering of formal concepts is a relevant approach for both partitioning and bi-partitioning tasks.

4.2.2. Convergence analysis

As the computational complexity of each iteration of CDK-MEANS depends also on the number of bi-sets, it is interesting to analyze the number of iterations CDK-MEANS needs to obtain stable centroids.

In our previous experiments, we have set to 100 the maximal number of

Table 7: Jaccard coefficient values w.r.t. class variable for different algorithms

	BI-CLUST	COCLUSTER		CDK-MEANS	
Dataset	Max	Max	Mean	Max	Mean
voting	0.647	0.653	0.624±0.019	0.674	0.674±0.000
titanic	0.428	0.560	0.441±0.074	0.598	0.434±0.070
iris-2	0.499	0.499	0.434±0.069	0.505	0.471±0.055
iris-3	0.493	0.524	0.432±0.094	0.556	0.476±0.045
zoo-2	0.514	0.460	0.365±0.062	0.463	0.423±0.037
zoo-7	-	0.61	0.509±0.096	0.772	0.372±0.092
breast-w	0.825	0.829	0.765±0.137	0.767	0.767±0.000
credit-3	0.423	0.518	0.383±0.048	0.506	0.387±0.040
credit-2	-	0.525	0.437±0.058	0.495	0.479±0.033
mr-2	-	0.694	0.494±0.132	0.699	0.480±0.120
mr-5	0.507	0.482	0.338±0.039	0.501	0.350±0.062
ads	-	0.432	0.432±0.000	0.885	0.665±0.131

Table 8: Jaccard coefficient values w.r.t. class variable for different algorithms

Dataset	K-MEANS		EM		CDK-MEANS	
	Max	Mean	Max	Mean	Max	Mean
voting	0.651	0.627 $\pm$ 0.034	0.637	0.637 $\pm$ 0.000	0.674	0.674 $\pm$ 0.000
titanic	0.554	0.466 $\pm$ 0.075	0.443	0.425 $\pm$ 0.002	0.598	0.434 $\pm$ 0.070
iris-2	0.512	0.497 $\pm$ 0.018	0.499	0.499 $\pm$ 0.000	0.505	0.471 $\pm$ 0.055
iris-3	0.539	0.510 $\pm$ 0.043	0.526	0.515 $\pm$ 0.014	0.556	0.476 $\pm$ 0.045
zoo-2	0.462	0.460 $\pm$ 0.013	0.461	0.461 $\pm$ 0.000	0.463	0.423 $\pm$ 0.037
zoo-7	0.922	0.588 $\pm$ 0.144	0.856	0.761 $\pm$ 0.047	0.772	0.372 $\pm$ 0.092
breast-w	0.825	0.788 $\pm$ 0.063	0.833	0.833 $\pm$ 0.000	0.767	0.767 $\pm$ 0.000
credit-3	0.444	0.382 $\pm$ 0.029	0.441	0.371 $\pm$ 0.035	0.506	0.387 $\pm$ 0.040
credit-2	0.519	0.469 $\pm$ 0.036	0.444	0.443 $\pm$ 0.008	0.495	0.479 $\pm$ 0.033
mr-2	0.687	0.537 $\pm$ 0.134	0.694	0.550 $\pm$ 0.141	0.699	0.480 $\pm$ 0.120
mr-5	0.497	0.356 $\pm$ 0.040	0.459	0.352 $\pm$ 0.038	0.501	0.350 $\pm$ 0.062
ads	-	-	-	-	0.885	0.665 $\pm$ 0.131

Table 9: Number of iterations for different datasets (mr-2 and mr-5 refer to mushroom with 2 and 5 clusters).

Dataset	N.Bi-sets	Min	Mean	Max	Failure
voting	199866	3	17.58	24	0
titanic	38	2	3.36	4	0
iris-2	50	3	4.55	6	0
iris-3	50	4	4.26	5	0
zoo-2	309	4	8.97	14	0
zoo-7	309	10	12.69	18	0
breast-w	4903	8	12.35	17	0
credit-3	29447	8	30.67	67	0
credit-2	29447	4	15.45	26	0
mr-2	53942	4	8.46	24	0
mr-5	53942	10	20.46	62	0
ads	7682	6	16.73	54	0

iterations. Is it possible to decide what is a relevant value for this parameter and a given dataset? Since we use a random initialization step, we studied the average behavior and the minimal/maximal number of iteration needed for 100 executions on several datasets. Results are in Table 9.

CDK-MEANS is able to complete the computation of stable centroids by using very few iterations for every used dataset. The average number of iterations is almost always lower than 20 (except for mr-5 and credit-a with $K = 3$ where it is slightly over). The maximum number of iterations is 8 datasets is lower than 25. Interestingly, no execution has terminated before getting stable centroids. These results suggest that we have here room for improvement by, e.g., a more accurate choice of the initial centroids to improve CDK-MEANS convergence and thus to reduce execution time.

4.2.3. Scalability issues

The number of formal concepts holding in even small datasets can be huge, especially in intrinsically noisy data. Since CDK-MEANS has a linear complexity in the number of bi-sets, it can be time-consuming. A simple

Table 10: Clustering results on **ads-internet** with different minimal size constraints

(σ_p, σ_o)	$ \mathcal{B} $	time(s)	$\tau(\text{mean})$	$\tau(\text{max})$	J-class	J-ref
(0,0)	7682	33	0.137 ± 0.109	0.538	0.8019	1
(4,4)	2926	8	0.194 ± 0.137	0.565	0.6763	0.6737
(5,5)	2075	5	0.254 ± 0.148	0.565	0.6862	0.7490
(5,10)	1166	2.5	0.223 ± 0.119	0.511	0.6745	0.7405
(7,10)	873	2	0.204 ± 0.095	0.549	0.6172	0.6658
(10,10)	586	1.5	0.227 ± 0.125	0.543	0.6080	0.7167

solution is to select a subset of the formal concept collection, for instance, selecting the ones which involve enough objects and/or enough properties. Interestingly, such minimal size constraints can be pushed into formal concept mining algorithms like D-MINER [4]. Not only it enables the extraction in hard contexts, but also, intuitively, it removes formal concepts which might be due to noise. We therefore guess that this can increase the quality of the clustering result. Let σ_o be the minimal size of the object set and σ_p be the minimal size of the property set. Properties (resp. objects) that are in relation with less than σ_o objects (resp. σ_p properties) will not be included in any formal concept.

As our bi-partitioning method is based only on a post-processing of these patterns, these objects and/or properties can not be included in the final bi-partition. This is not necessarily a problem if we prefer a better robustness to noise. However, one can be interested in finding a bi-partition that includes all objects and properties. The top and bottom formal concepts $(\mathcal{O}, \emptyset)$ and $(\emptyset, \mathcal{P})$ can be added to solve this problem. This has been done in some experiments (**mushroom**, **credit-a**) and we noticed that the decrease of the Jaccard and Goodman-Kruskal’s coefficients were not significant. We made further experiments to understand the impact of using minimal size constraints on both the needed execution time and the quality of the computed bi-partitions. We have considered **internet-ads** as the most suitable for these experiments (high cardinality for both object and property sets). We extracted formal concepts by setting some combinations of constraints ($0 \leq \sigma_p < 10$ and $0 \leq \sigma_o < 10$) and by adding the top and the bottom

formal concepts. The results are summarized in Table 10. It shows that, increasing the minimal size threshold considerably reduces the number of extracted formal concepts and thus the average execution time. Notice that the extraction time decreases from 4 seconds (for $\sigma_p = \sigma_o = 0$) to less than one second (for $\sigma_p = \sigma_o = 10$).

Moreover, the maximum Goodman-Kruskal’s coefficient does not change significantly. In some cases, it is greater than the coefficient computed when no size constraint is used. Also the average values of the Goodman-Kruskal’s measures are better in general (while standard deviation values are similar). We then computed the Jaccard index of the different partitions w.r.t. the class variable (J-class column) and the partition obtained without setting any constraint (J-ref column). The slight variability of the Jaccard indexes and the high values of the τ measures show that even if there are some differences between the partitions, they are still consistent w.r.t. the class one. Finally, results are always better than those obtained by using COCLUSTER (see Figure 6 and Figure 7) whose average execution time is about 4.2 seconds. In other terms, increasing σ_p and σ_o can eliminate the impact of noise due to sparse sub-matrices. In particular, grouping larger formal concepts can improve the relevancy of computed bi-partitions. Notice that if we do not add $(\mathcal{O}, \emptyset)$ and $(\emptyset, \mathcal{P})$, we obtain better results involving a subset of the original matrix: constraints can be triggered to trade-off between the coverage of the bi-partition and the quality of the result.

4.2.4. Experimental results when using δ -bi-sets

In the previous section, we reported about CDK-MEANS behavior when considering collections of formal concepts. We now emphasize the genericity of the framework by considering experiments on collections of δ -bi-sets (see Section 3.2). In most cases, we can expect smaller collections of possibly larger bi-sets which might however better capture some strong associations within the data (thanks to fault-tolerance). This might at least speed up the first phase of the CDK-MEANS algorithm. The experiments have been performed as described in the previous section. The only difference is the type of bi-sets we have used. We have computed the δ -bi-sets by using our straightforward extension of the implementation from [7]). We have considered $\delta = 1$ (at most one exception per column) and $\delta = 2$ (at most two exceptions per column). For **voting-records**, **mushroom** and **credit-a**, we used the same size constraints as for our previous experiments based on formal concepts.

Table 11: Number of iterations for different datasets using 1-bi-sets (mr-2 and mr-5 refer to mushroom with 2 and 5 clusters).

Dataset	N.Bi-sets	Min	Mean	Max	Gain
voting	97330	17	18.70	21	48.20%
titanic	36	2	3.52	7	0.83%
iris-2	42	4	4.68	7	13.51%
iris-3	42	4	4.80	7	5.35%
zoo-2	258	5	7.42	11	30.93%
zoo-7	258	10	12.82	30	15.61%
breast-w	3776	7	9.06	13	43.52%
credit-3	26740	8	31.66	68	6.24%
credit-2	26740	4	16.90	28	0.66%
mr-2	26304	3	7.98	22	54.03%
mr-5	26304	7	17.63	64	57.98%
ads	6832	6	13.98	38	25.70%

A first analysis concerns the gain in performances. We have three objective parameters to evaluate this gain (or loss): the number of processed bi-sets, the average number of needed iterations for delivering a bi-partition, and the real gain in terms of average number of comparisons between centroids and bi-sets. The average number of comparisons is obtained by multiplying the average number of iterations by the number of bi-sets. The gain is computed as the difference between the average number of comparisons obtained by using formal concepts, and the one obtained by using δ -bi-sets. Results are in Table 11 and Table 11 (negative values of gain indicate loss in performances). As expected, for $\delta = 1$, the size of the δ -bi-set collections is smaller than the formal concept ones.

Let us discuss the dynamic behavior of CDK-MEANS. In these experiments, the average number of iterations is almost the same than for formal concept processing. However, the total number of comparisons is reduced by more than 25% in 6 datasets. When using 2-bi-sets, this performance improvement is clear (see Table 12), with the exception on mushroom. In

Table 12: Number of iterations for different datasets using 2-bi-sets (**mr-2** and **mr-5** refer to mushroom with 2 and 5 clusters).

Dataset	N.Bi-sets	Min	Mean	Max	Gain(FC)	Gain($\delta = 1$)
voting	55359	10	16.73	26	73.65%	49.12%
titanic	36	2	3.27	5	7.80%	7.03%
iris-2	34	3	4.76	6	28.77%	17.64%
iris-3	34	2	3.31	5	47.12%	44.13%
zoo-2	190	6	7.68	11	47.33%	23.74%
zoo-7	190	5	8.16	12	60.45%	53.14%
breast-w	2831	6	7.87	11	63.22%	34.87%
credit-3	23946	14	34.39	69	8.82%	2.75%
credit-2	23946	4	17.78	28	6.41%	5.79%
mr-2	54685	3	8.24	21	1.31%	-114.67%
mr-5	54685	9	19.55	53	3.13%	-130.53%
ads	6304	5	12.22	29	40.07%	19.35%

this case, the number of 2-bi-sets is larger than the number of 1-bi-sets, and even than the number of formal concepts (see Table 9). This could appear a bit strange, but let us remember that we have used minimal size constraints on this particular dataset. Indeed, when extracting δ -bi-sets with $\delta = \delta_1$ under a minimal size constraint, it could happen that the resulting collection is larger than the one obtained with a $\delta = \delta_2$ where $\delta_2 < \delta_1$. Furthermore, formal concepts are δ -bi-sets with $\delta = 0$. The reason is that, even if the size of the whole collection of δ_2 -bi-sets (without minimal size constraints) is smaller than the related collection of δ_1 -bi-sets, there could be more δ_1 -bi-sets satisfying the constraint than δ_2 -bi-sets, since, by allowing some exceptions, they capture larger associations. This explains the loss in performance when using $\delta = 2$ in **mushroom** (the average number of comparisons is doubled w.r.t. the average number of comparisons obtained when $\delta = 1$).

Let us now consider the quality analysis. We analyzed the maximal and average behavior of the Goodman-Kruskal's τ coefficient, and of the Jaccard index (w.r.t. the class variable). The results are in Table 13 (for $\delta = 1$) and Table 14 (for $\delta = 2$) (values in bold mean better results w.r.t. Table 6).

In some cases, we obtained better maximal Goodman-Kruskal's coefficients, while it seems that using δ -bi-sets has no impact on the average behavior. The Jaccard index seems to take advantage of δ -bi-sets vs. formal concepts. The experiments show that δ -bi-sets are a valid alternative to formal concepts for computing bi-partitions. The gain in performances does not imply a loss in quality, but using greater values of δ is not always the best choice. These experiments have shown that CDK-MEANS can work with two different types of bi-sets which capture locally strong associations. Clearly, other types of bi-sets could be used as well, e.g., other fault-tolerant formal concept types or dense rectangles.

4.3. Application to real gene expression data

To demonstrate the added-value of our approach in a “real-world” problem, we applied CDK-MEANS to a microarray dataset [8] concerning the transcriptome of the intraerythrocytic developmental cycle of *Plasmodium Falciparum*, i.e., a causative agent of human malaria. The data provide the expression profile of 3 719 genes in 46 biological samples. Each sample corresponds to a time point of the developmental cycle: it begins with merozoite invasion of the red blood cells, and it is divided into three phases, the ring, trophozoite and schizont stages (resp. the mosquito, liver and blood stages). After 48 hours, the cells replicate and divide. At the 17th and 29th

Table 13: Jaccard and Goodman-Kruskal’s coefficient values for $\delta = 1$ (mr-2 and mr-5 refer to mushroom with 2 and 5 clusters).

Dataset	Goodman-Kruskal		Jaccard	
	Max	Mean	Max	Mean
voting	0.314	0.313 ± 0.000	0.667	0.664 ± 0.002
titanic	0.363	0.217 ± 0.115	0.598	0.428 ± 0.071
iris-2	0.545	0.442 ± 0.134	0.507	0.483 ± 0.046
iris-3	0.544	0.298 ± 0.092	0.556	0.473 ± 0.059
zoo-2	0.193	0.170 ± 0.011	0.464	0.441 ± 0.016
zoo-7	0.120	0.069 ± 0.018	0.799	0.376 ± 0.095
breast-w	0.497	0.497 ± 0.000	0.763	0.763 ± 0.000
credit-3	0.078	0.066 ± 0.008	0.447	0.397 ± 0.043
credit-2	0.095	0.086 ± 0.019	0.497	0.485 ± 0.030
mr-2	0.174	0.151 ± 0.015	0.684	0.538 ± 0.102
mr-5	0.116	0.101 ± 0.017	0.677	0.635 ± 0.019
ads	0.577	0.237 ± 0.154	0.851	0.665 ± 0.138

Table 14: Jaccard and Goodman-Kruskal’s coefficient values for $\delta = 2$ (mr-2 and mr-5 refer to mushroom with 2 and 5 clusters).

Dataset	Goodman-Kruskal		Jaccard	
	Max	Mean	Max	Mean
voting	0.312	0.312 ± 0.000	0.647	0.647 ± 0.000
titanic	0.363	0.171 ± 0.083	0.602	0.498 ± 0.071
iris-2	0.546	0.442 ± 0.141	0.509	0.469 ± 0.057
iris-3	0.546	0.343 ± 0.096	0.563	0.471 ± 0.055
zoo-2	0.191	0.162 ± 0.020	0.471	0.442 ± 0.029
zoo-7	0.111	0.068 ± 0.015	0.737	0.411 ± 0.098
breast-w	0.420	0.420 ± 0.000	0.735	0.735 ± 0.000
credit-3	0.077	0.067 ± 0.009	0.506	0.404 ± 0.045
credit-2	0.094	0.082 ± 0.015	0.661	0.629 ± 0.039
mr-2	0.173	0.158 ± 0.013	0.687	0.570 ± 0.102
mr-5	0.116	0.106 ± 0.005	0.501	0.337 ± 0.031
ads	0.567	0.311 ± 0.137	0.830	0.669 ± 0.142

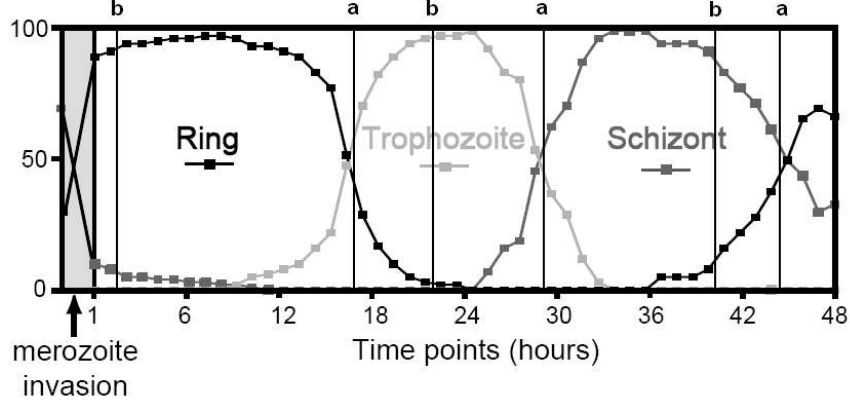


Figure 1: Cluster borderlines for CDK-MEANS (a) and COCLUSTER (b). The curves show the percent representation of ring-, trophozoite-, or schizont-stage parasites in the culture at every timepoint

time points there are two sharp transitions (see Figure 1). The numerical gene expression data given in [8] has been discretized by assessing the property encoding methods as defined in [25]: for each gene g , we assigned the Boolean value 1 to those samples whose expression level was greater than 25% of its max expression level. The first experiment was then to identify the three developmental stages by applying CDK-MEANS with $K = 3$. The maximal score we obtained is $\tau_S = 0.5129$. By applying COCLUSTER, the maximal Goodman-Kruskal's coefficient is sensibly higher ($\tau_S = 0.6123$). However, if we look at the resulting partition on the biological sample set, it is significantly different w.r.t. the one obtained with CDK-MEANS. With CDK-MEANS, the agreement of the obtained partition and the three stages described in [8], is very high (see Figure 1 lines a). The three groups are well distinguished, and the transitions between the clusters correspond to those reported in [8]. With COCLUSTER, cluster borderlines are shifted (see Figure 1 lines b).

We have then analyzed the partition on genes. We checked the cluster assignment of 12 functional groups of genes. Each group contains a number of genes with the same function, and each function concerns one or at most two developmental stages. Again, these groups are well described in [8] and contain from 6 to 135 genes. For each group of genes, Table 15 represents the number of genes that have been assigned to each cluster, divided by the

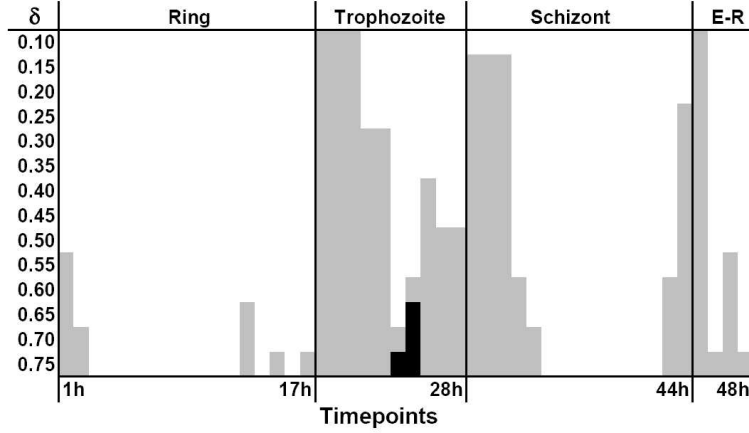


Figure 2: Cluster overlapping zones when δ_p is varying.

total number of genes belonging to the functional group (when some genes are missing in the data, the sum of ratios is less than 1). Here again, the gene repartition is relevant w.r.t. the available biological knowledge [8]. For instance, the first 4 groups of genes, are known to be active in the ring and early trophozoite stages. Analogously, the proteasome, plastid, merozoite invasion and actin myosin motility groups are functions that are characteristic of the schizont stage. This is obvious for the last three groups (including about one hundred genes), where the ratios are equal to 1, while in COCLUS-TER results, these genes are shared by the two classes related to trophozoite and schizont stages.

Finally, we performed soft clustering tasks (enabling overlapping) and we analyzed the relevancy of the intersection between clusters. We used the previous results, and for each value of δ_p (from 0.10 to 0.75, with a step equal to 0.05), we built the related soft partitions. Results are in Figure 2, where the light gray zones are intersections between two clusters, and the black zone is an intersection between all the three clusters. This experiment validates our approach when $\delta_p \leq 0.5$: the intersections contain time points that are involved in the transactions between two adjacent stages.

4.4. Discussion

Let us conclude the whole experimental validation. The experiments on benchmark datasets, show that our approach is competitive with other existing (bi-) clustering techniques. Instead, the application to the gene expression

Table 15: Assignment ratios of functional groups in the three discovered bi-clusters.

Function	Ring	Trophozoite	Schizont
transcription	0.91	0.04	0.00
cytoplasmic translation	0.98	0.02	0.00
glycolytic pathway	0.43	0.43	0.00
ribonucleotide synthesis	0.67	0.22	0.00
deoxynucleotide synthesis	0.00	0.29	0.71
dna replication	0.00	0.33	0.67
tca cycle	0.00	0.36	0.64
proteasome	0.00	0.09	0.51
plastid genome	0.00	0.00	1.00
merozoite invasion	0.00	0.00	1.00
actin myosin motility	0.00	0.00	1.00
early ring transcripts	0.79	0.00	0.21

dataset, has shown some significative differences w.r.t. to COCLUSTER.

First, by considering associations (e.g., formal concepts, δ -bi-sets) between objects and properties as the elements to process, we capture a significantly different structure w.r.t. to COCLUSTER. Indeed, these locally strong interactions play a critical role in our co-clustering technique (the cluster assignation for each object/property depend on the number of bi-sets involving it in each centroid).

From a biological point of view, roughly speaking, if we consider bi-sets as putative transcription modules (or strong association of co-regulated genes and the experimental conditions which give rise to this up-regulation), then a bi-cluster can be view as the result of the sum of a group of close local associations. This can explain while, the three stages of the Plasmodium Falciparum life-cycle, are better identified by our method than by using COCLUSTER.

Another advantage is that CDK-MEANS can easily compute partitions with overlapping clusters. This is a new possibility in bi-partitioning algorithms. Even if some algorithms (e.g., [9]) enable the extraction of overlapping bi-clusters, they are not bi-partitioning approaches. We know that, for some applications, this has an important added value. Let us however notice that, we did not used overlapping during our experimental validation and that, if overlapping is considered, objective quality measures (e.g., Goodman-Kruskal or Jaccard coefficients) can not be used.

Some of the limitations of our approach are due to the K-MEANS-like approach (e.g., sensibility to the the initial centroids, fixed number of clusters). This might be balanced by studying alternative clustering schemes, e.g., hierarchical clustering. We used our instance (CDK-MEANS) only to provide an experimental feedback on one specific instance.

5. Related work

Many co-clustering methods have been developed, possibly dedicated to gene expression data analysis. Y. Kluger et al. [20] propose a spectral co-clustering method. First they perform an adequate normalization of the dataset to accentuate bi-clusters if they exist. Then, they consider that the correlation between two columns is better estimated by the expression level mean of each column with respect to a partition of the rows. The bi-partition is computed by the algebraic eigenvalue decomposition of the normalized matrix. Their algorithm critically depends on the normalization procedure.

I. Dhillon et al. [10] and C. Robardet et al. [29] have considered the two searched partitions as discrete random variables whose association must be maximized. Several measures can be used and whereas COCLUSTER [10] uses the loss in mutual information, BI-CLUST [29] uses Goodman-Kruskal's τ coefficient to evaluate the link strength between the two variables. In both algorithms, a local optimization method is used to optimize the measure by alternatively changing a partition when the other one is unchanged. The main difference is that τ measure is independent of the number of bi-clusters and thus BI-CLUST can automatically decide the number of bi-clusters. None of these methods enable to compute overlapping bi-partitions.

L. Lazzeroni et al. [22] propose to consider each matrix value as a sum of variables. Each variable represents a particular phenomenon in the data and corresponds to a bi-cluster. In each bi-cluster, column or row values are linearly correlated. Then, the method consists in determining the model minimizing the Euclidean distance between the matrix and the modeled values. This method is similar to the eigenvalue decomposition used in [20] without the orthogonal constraint on the computed variables.

It is out of the scope of this paper to consider other local pattern types as a related work which would need a detailed discussion. Indeed, our contribution is not to promote one specific pattern type but to propose a framework which exploits provided local associations. Therefore, instead of considering formal concepts or δ -bi-sets, one might start the process by using different types of bi-sets derived from frequent itemsets [1], error-tolerant frequent itemsets (see, e.g., [38]), large tiles [13], dense itemsets [30], or support envelopes [31], just to name a few. One might even start the process with collections of patterns of different types, e.g., a collection of large formal concepts plus a few fault-tolerant patterns. Discussing the advantages and limitations of these pattern types is well-done in the scientific papers that introduce them.

Furthermore, in the context of gene expression data analysis, several authors have considered the computation of potentially overlapping local patterns that they call bi-clusters (see [23] for a recent survey). J. Ihmels et al. [18] propose a simple algorithm which build in two steps a single association called a bi-cluster starting from a column set. First, they consider that the rows having a high score (greater than a threshold on the normalized matrix) on these columns belong to the bi-cluster. Then, they use the same principle to increase the original column set. In [9], Y. Cheng et al. propose a bi-clustering algorithm for numerical data. They define a bi-cluster as a subset

of rows and subset of columns with a low mean squared residue. When the measure is equal to 0, the bi-cluster contains rows having the same value on the bi-cluster columns. When the measure is greater than 0, one can remove rows or columns to decrease the value. Thus the method consists in finding maximal size bi-cluster such that the measure is inferior to a threshold. Various heuristics can be used for this purposes. Clearly, such collections of local patterns could be used as an input to our local to global clustering scheme.

Notice also that formal concepts have also been used as bi-clusters. In [11], N. Durand et al. propose a measure to select a subset of formal concepts such that these concepts represents with parsimony the whole dataset. In this case, bi-clusters are formal concepts and the number of extracted bi-clusters remains huge: interpretation by end-users turns to be quite hard. In these previous methods, each bi-cluster is built independently and these methods do not provide a global structure of the dataset. There are some analogy between local bi-clustering algorithms and the so-called pattern-based clustering [35, 34] which has been designed for gene expression data analysis. Two genes are clustered together if they exhibit a coherent pattern in a subset of dimension. Even in this case, bi-clusters are not element of a bi-partition, but can be considered as local bi-sets (and thus, as input elements for our algorithm).

6. Conclusion and future work

To face the new challenges in categorical data analysis, e.g., in molecular biology, we need conceptual co-clustering techniques. We have introduced a new co-clustering framework which exploits local patterns in the data when computing a collection of (possibly overlapping) bi-clusters. The instance CDK-MEANS builds simultaneously a partition on objects and a partition on properties by applying a K-MEANS-like algorithm to a collection of bi-sets (e.g., formal concepts or any other collection of set patterns which capture locally strong associations between sets of objects and sets of properties). Our experimental validation has confirmed the added-value of CDK-MEANS w.r.t. other clustering or co-clustering algorithms. We demonstrated that such a “from local patterns to a relevant global pattern” approach can work. It has enabled a re-discovery within a real microarray dataset with more relevancy than available co-clustering algorithms.

Many other instances of such a framework might be studied. For instance, given extracted local patterns, alternative clustering techniques could be con-

sidered. Also, many kinds of local patterns (i.e., relevant bi-sets for capturing local associations) could be considered as well (e.g., the dense itemsets from [30]). An in depth understanding of the fundamental relationship between co-clustering (i.e., computing global similarity structures within a dataset), tiling (see, e.g., [13]) and local pattern detection has yet to be done. Finally, an exciting challenge concerns constraint-based clustering (see, e.g., [33]). Our framework gives rise to opportunities for pushing constraints at two different levels, i.e., during the descriptive local pattern mining phase but also when building bi-partitions from these patterns. Indeed, in [26], we consider the possibility to exploit (extended) must-link and cannot-link constraints plus the so-called interval constraints when looking for a bi-partition. Constraints on the local patterns are derived from the specified constraints on the bi-partition and it becomes possible to achieve a better trade-off between the satisfaction of the user-defined constraint w.r.t. the shape of the bi-clusters on one hand, and the optimization of the clustering objective function on another hand. Very preliminary results have been reported but they already confirm that such a two-level scheme is promising for designing scalable constrained co-clustering techniques.

Acknowledgements

The authors thank Luigi Mantellini, J  r  my Besson, and Christophe Rigotti for technical support and stimulating discussions. This research has been partly funded by EU contract IST-FET IQ FP6-516169 (FET arm of the IST programme).

References

- [1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.
- [2] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66 – 75, December 2000.
- [3] J. Besson, C. Robardet, and J-F. Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *Proceedings ICCS’06*, volume 4068 of *LNCS*, pages 144–157, Aalborg, Denmark, July 2006. Springer.

- [4] J. Besson, C. Robardet, J.-F. Boulicaut, and S. Rome. Constraint-based concept mining and its application to microarray data analysis. *Intelligent Data Analysis*, 9(1):59–82, 2005.
- [5] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.
- [6] F. Bonchi and C. Lucchese. Extending the state-of-the-art of constraint-based pattern discovery. *Data & Knowledge Engineering*, 60:377–399, 2007.
- [7] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003.
- [8] Z. Bozdech, M. Llinás, B. Lee Pulliam, E.D. Wong, J. Zhu, and J.L. DeRisi. The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biology*, 1(1):1–16, October 2003.
- [9] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings ISMB 2000*, pages 93–103, San Diego, USA, 2000. AAAI Press.
- [10] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings ACM SIGKDD 2003*, pages 89–98, Washington, USA, 2003. ACM Press.
- [11] N. Durand and B. Crémilleux. Ecclat: a new approach of clusters discovery in categorical data. In *Proceedings ES 2002*, pages 177–190, Cambridge, UK, 2002. Springer.
- [12] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.
- [13] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Proceedings DS’04*, volume 3245 of *LNAI*, pages 278–289, Padova, Italy, 2004. Springer.
- [14] B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on fimi’03. *SIGKDD Explorations*, 6(1):109–117, 2004.

- [15] L. A. Goodman and W. H. Kruskal. Measures of association for cross classification. *Journal of the American Statistical Association*, 49:732–764, 1954.
- [16] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.
- [17] D. J. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. MIT Press, 2001.
- [18] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31:370–377, august 2002.
- [19] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood cliffs, New Jersey, 1988.
- [20] Y. Kluger, R. Basri, JT Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13:703–716, 2003.
- [21] S. O. Kuznetsov and S. A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14 (2-3):189–216, 2002.
- [22] L. C. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2000.
- [23] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 1(1):24–45, 2004.
- [24] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, January 1999.
- [25] R. G. Pensa and J-F. Boulicaut. Boolean property encoding for local set pattern discovery: An application to gene expression data analysis. In *Local Pattern Detection, Dagstuhl Seminar Revised Selected Papers*, volume 3539 of *LNCS*, pages 115–134. Springer, 2005.

- [26] R. G. Pensa, C. Robardet, and J-F. Boulicaut. *Constrained Clustering: Advances in Algorithms, Theory and Applications*, chapter Constraint-driven Co-Clustering of 0/1 Data, pages 145–170. Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, July.
- [27] R. G. Pensa, C. Robardet, and J.-F. Boulicaut. A bi-clustering framework for categorical data. In *Proceedings PKDD 2005*, volume 3721 of *LNAI*, pages 643–650, Porto, Portugal, October 2005. Springer.
- [28] Ruggero G. Pensa, Céline Robardet, and Jean-François Boulicaut. Supporting bi-cluster interpretation in 0/1 data by means of local patterns. *Intelligent Data Analysis*, 10(5):457–472, 2006.
- [29] C. Robardet and F. Feschet. Efficient local search in conceptual clustering. In *Proceedings DS'01*, volume 2226 of *LNCS*, pages 323–335. Springer, november 2001.
- [30] J. K. Seppänen and H. Mannila. Dense itemsets. In *Proceedings ACM SIGKDD'04*, pages 683–688, Seattle, USA, 2004. ACM Press.
- [31] M. Steinbach, P.-N. Tan, and V. Kumar. Support envelopes: a technique for exploring the structure of association patterns. In *Proceedings ACM SIGKDD'04*, pages 296–305, Seattle, USA, 2004. ACM Press.
- [32] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing iceberg concept lattices with TITANIC. *Data & Knowledge Engineering*, 42:189–222, 2002.
- [33] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings ICML 2001*, pages 577–584, Williamstown, USA, 2001. Morgan Kaufmann.
- [34] H. Wang, J. Pei, and P. S. Yu. Pattern-based similarity search for microarray data. In *Proceedings ACM SIGKDD'05*, pages 814–819, Chicago, USA, 2005. ACM Press.
- [35] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings ACM SIGMOD'02*, pages 394–405, Madison, USA, 2002. ACM Press.

- [36] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. Reidel, 1982.
- [37] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.
- [38] C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings ACM SIGKDD’01*, pages 194–203, San Francisco, USA, 2001. ACM Press.
- [39] M. J. Zaki and C. J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings SIAM DM’02*, Arlington, USA, Avril 2002.
- [40] M. J. Zaki, M. Peters, I. Assent, and T. Seidl. Clicks: An effective algorithm for mining subspace clusters in categorical datasets. *Data & Knowledge Engineering*, 60:51–70, 2007.

Nikolaj Tatti



TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

ADVANCES IN MINING BINARY DATA: ITEMSETS AS SUMMARIES

Nikolaj Tatti

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Information and Natural Sciences for public examination and debate in Auditorium TU1 at Helsinki University of Technology (Espoo, Finland) on the 6th of June, 2008, at 12 noon.

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science

Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietojenkäsittelytieteen laitos

Distribution:

Helsinki University of Technology

Faculty of Information and Natural Sciences

Department of Information and Computer Science

P.O.Box 5400

FI-02015 TKK

FINLAND

URL: <http://ics.tkk.fi>

Tel. +358 9 451 1

Fax +358 9 451 3369

E-mail: series@ics.tkk.fi

© Nikolaj Tatti

ISBN 978-951-22-9375-9 (Print)

ISBN 978-951-22-9376-6 (Online)

ISSN 1797-5050 (Print)

ISSN 1797-5069 (Online)

URL: <http://lib.tkk.fi/Diss/2008/isbn9789512293766/>

Multiprint

Espoo 2008

Tatti N. (2008): **Advances in Mining Binary Data: Itemsets as Summaries**. Doctoral Thesis, Helsinki University of Technology, TKK Dissertations in Information and Computer Science, Report D5, Espoo, Finland.

Keywords: data mining, frequent itemset, boolean queries, safe projections, itemset ranking.

Abstract

Mining frequent itemsets is one of the most popular topics in data mining. Itemsets are local patterns, representing frequently cooccurring sets of variables. This thesis studies the use of itemsets to give information about the whole dataset.

We show how to use itemsets for answering queries, that is, finding out the number of transactions satisfying some given formula. While this is a simple procedure given the original data, the task transforms into a computationally infeasible problem if we seek the solution using the itemsets. By making some assumptions of the structure of the itemsets and applying techniques from the theory of Markov Random Fields we are able to reduce the computational burden of query answering.

We can also use the known itemsets to predict the unknown itemsets. The difference between the prediction and the actual value can be used for ranking itemsets. In fact, this method can be seen as generalisation for ranking itemsets based on their deviation from the independence model, an approach commonly used in the data mining literature.

The next contribution is to use itemsets to define a distance between the datasets. We achieve this by computing the difference between the frequencies of the itemsets. We take into account the fact that the itemset frequencies may be correlated and by removing the correlation we show that our distance transforms into Euclidean distance between the frequencies of parity formulae.

The last contribution concerns calculating the effective dimension of binary data. We apply fractal dimension, a known concept that works well with real-valued data. Applying fractal dimension directly is problematic because of the unique nature of binary data. We propose a solution to this problem by introducing a new concept called normalised correlation dimension. We study our approach theoretically and empirically by comparing it against other methods.

Tiivistelmä

Kattavien joukkojen louhinta on yksi suosituimmista tiedon louhinnan teemoista. Kattavat joukot ovat paikallisia hahmoja: ne edustavat usein esiintyviä muut-tujakombinaatioita. kattavien joukkojen käyttöä koko tietokantaa kuvaaviin tarkoituksiin.

Kattavia joukkoja voidaan käyttää Boolean kyselyihin vastaamiseen, ts. an-netun Boolean kaavan toteuttavien tietuiden lukumäärän arviointiin. Tehtävästä tulee kuitenkin laskennallisesti vaativa, jos käytössä ovat vain kattavat joukot. Väitöskirjassa osoitetaan, että tietyin oletuksin ongelman ratkaisemista voidaan helpottaa käyttäen hyväksi tekniikoita, jotka perustuvat Markov-kenttiin.

Väitöskirjassa tutkitaan myös miten kattavia joukkoja voidaan käyttää tun-temattomien joukkojen frekvenssin ennustamiseen. Varsinaisen datasta lasketun frekvenssin ja ennusteen välistä erotusta voidaan käyttää kattavan joukon merk-itsevyyden mittana. Tämä lähestymistapa on itseasiassa tiedon louhinnassa usein toistuvan tärkeysmitan yleistys, jossa kattavan joukon tärkeys on sen poikkeama riippumattomuusoletuksesta.

Väitöskirjan seuraava tutkimusaihe on kattavien joukkojen käyttö tietokanto-jen välisen etäisyyden määrittelymiseen. Etäisyys määritellään kattavien joukko-jen frekvenssien erotuksena. Kattavien joukkojen frekvenssien välillä saattaa olla korrelaatiota ja eliminoimalla tämä korrelaatio työssä osoitetaan, että etäisyys vastaa tiettyjen pariteettikyselyiden välistä euklidista etäisyyttä.

Väitöskirjan viimeinen teema on binääritietokannan efektiivisen dimension määrittelyminen. Työssä sovelletaan fraktaalidimensiota, joka on suosittu mene-telmä ja soveltuu hyvin jatkuvalle datalle. Tämän lähestymistavan soveltami-nen diskreettiin dataan ei kuitenkaan ole suoraviivaista. Työssä ehdotetaan ratkaisuksi normalisoitua korrelaatioidimensiota. Lähestymistapoja tarkastellaan sekä teoreettisesti että empiirisesti vertailemalla sitä muihin tunnettuihin mene-telmiin.

Contents

Contents	i
List of Figures	iii
List of Notations	v
List of Publications	vii
Preface	viii
1 Introduction	1
2 Binary data	7
2.1 Binary data and Itemsets	7
3 Linear Programming for Predicting Itemset Frequencies	15
3.1 Theory	16
3.2 Algorithms Solving Linear Programs	19
4 Markov Random Field Theory for Optimising Prediction of Itemset Frequencies	21
4.1 Theory	22
5 Kullback-Leibler Divergence for Ranking Itemsets	29
5.1 Definitions	30
5.2 Maximum Entropy	32
6 Predicting Itemset Frequencies	35
6.1 Definition of the Problem	37
6.2 Complexity of Querying Itemsets	40
6.3 Safe sets	40

6.4	Optimising Linear Program via Markov Random Fields	44
6.5	Entropy Based Ranking of Itemsets	45
7	Distances between Binary Data Sets	49
7.1	Constrained Minimum Distance	50
7.2	Alternative Definition	53
8	Fractal Dimension of Binary Data	57
8.1	Correlation Dimension	58
8.2	Normalised Correlation Dimension	61
A	Proofs for the Theorems	65
A.1	Proof of Proposition 2.12	65
A.2	Proof of Proposition 4.9	66
A.3	Proof of Theorem 5.6	66
A.4	Proof of Theorem 8.3	68
A.5	Proof of Theorem 8.4	69
	Bibliography	73
	Index	82

List of Figures

1.1	Dependency graph of theorems presented in the thesis.	6
3.1	A geometrical interpretation of Linear Programming. The feasible set in this case is a triangle. If we assume that the vector c has length 1 and that x is a point from the feasible set, then $c^T x$ is the length of vector p , the orthogonal projection of x into c	17
4.1	An example of a non-triangulated graph and a graph resulting from a triangulation process. In the left graph there is a chordless cycle $a-b-c-d$. This cycle is removed in the right graph by adding a chord $a-c$	23
4.2	A clique graph of the graph G given in Figure 4.1(b). The oval nodes are the cliques of G and the square nodes are the separators, that is, intersections of the immediate cliques.	24
4.3	Spanning trees of the clique graph given in Figure 4.2. The tree in Figure 4.3(c) does not satisfy the running intersection property since the node c is not included in ae . The left and the centre tree are junction trees.	25
6.1	Graphs related to Example 6.8.	42
6.2	Graphs related to Example 6.10.	42
6.3	Distributions of the sizes of safe sets for random queries containing 2–4 attributes. The left histogram is obtained from the <i>Paleo</i> data set and the right histogram is obtained from the <i>Mushroom</i> data set.	43
6.4	Graphs related to Example 6.13.	45
6.5	Ranks for queries from synthetic data set. Each box represents queries with particular number of attributes.	48
6.6	Ranks for queries from the <i>Paleo</i> data set. Each box represents queries with particular number of attributes.	48

7.1	Illustration of the CM distance. The triangle represents the set of all possible distributions. The sets $\mathcal{C}(\mathcal{F}, D_i)$ are lines and the sets $\mathcal{P}(\mathcal{F}, D_i)$ are the segments containing the joint points from the set of all distributions and $\mathcal{C}(\mathcal{F}, D_i)$. The CM distance is proportional to the shortest distance between the spaces $\mathcal{C}(\mathcal{F}, D_1)$ and $\mathcal{C}(\mathcal{F}, D_2)$	51
7.2	Distance matrices for <i>Bible</i> , <i>Addresses</i> , and <i>Abstract</i> . Dark values indicate small distances. In the first column the feature set <i>ind</i> contains the independent means, in the second feature set <i>cov</i> the pairwise correlation is added, and in the third column the feature set <i>freq</i> consists of $10K$ most frequent itemsets, where K is the number of attributes. Darker colours indicate smaller distances.	55
8.1	Examples of $\text{cd}_A(D)$ for different data sets. Plots represent three different data sets, each of them having 50 independent columns. The probability of a variable being 1 is p (indicated in the legend). The left figure is a regular plot of $\mathbb{P}(Z_D < r)$. The right figure is a log-log plot of $\mathbb{P}(Z_D < r)$. The crosses indicate the end points r_1 and r_2 that were determined by using $\alpha_1 = 1/4$ and $\alpha_2 = 3/4$. The slopes of the straight lines in the log-log plot are $\text{cd}_A(D; 1/4, 3/4)$. Note that the lines are gentler for smaller p	59
8.2	Correlation dimension $\text{cd}_A(D; 1/4)$ as a function of $\text{acd}(D)$ for data with independent columns (see Proposition 8.5). The y -axis is $\text{cd}_A(D; 1/4)$ and the x -axis is $\text{acd}(D) = \mu/\sigma$, where $\mu = \mathbb{E}[Z_D]$ and $\sigma^2 = \text{Var}[Z_D]$. The slope of the line is about $C(1/4) = 0.815$	62
8.3	An illustration of computing normalised correlation dimension. The original data D is permuted, thus obtaining $\text{ind}(D)$. The margins of $\text{ind}(D)$ are forced to be equal such that the resulting dataset $\text{ind}(K, s)$ has the same correlation dimension. The dataset $\text{ind}(H, s)$ is computed such that $\text{cd}(\text{ind}(H, s)) = \text{cd}(D)$. H is the normalised correlation dimension.	63
8.4	Normalised correlation dimension for data having K independent dimensions for $K \in \{50, 100, 150, 200\}$. In Figure 8.4(a) the normalised correlation dimension $\text{ncd}_A(D)$ is concentrated around the number of attributes. In Figure 8.4(b) $\text{ncd}_A(D)$ is plotted as a function of μ , the average distance between two random points. The x -axis is $\mu = \mathbb{E}[Z_D]$ and the y -axis is $\text{ncd}_A(D; 1/4)$	64
8.5	Normalised correlation dimension as a function of $K\text{cd}_A(D)^2/\text{cd}_A(\text{ind}(D))^2$. Each point represents one data set. Figure 8.5(a) contains data sets with independent columns and Figure 8.5(b) contains data sets from the <i>20 Newsgroups</i> collection.	64

List of Notations

$\mathbb{Q}$	rational numbers
$\mathbb{R}$	real numbers
x^T	transpose of a vector x
x_i	i th element of a vector x
Ω	sample space, usually $\{0, 1\}^K$
ω	binary vector
K	dimension of sample space, number of attributes
a_i	i th attribute
A	set of all attributes, $A = \{a_1, \dots, a_K\}$
B, C, Q	itemsets
$\mathcal{F}, \mathcal{G}, \mathcal{I}, \mathcal{C}, \mathcal{A}$	families of itemsets
D	binary data set
$ D $	number of elements in D
$\cup$	set union
$\cap$	set intersection
$\vee$	disjunctive operator
$\wedge$	conjunctive operator
$\oplus$	exclusive-or (XOR) operator
S_F	indicator function of a Boolean formula F , returns 1 if and only if F is satisfied
$S_{\mathcal{F}}$	indicator function for a family $\mathcal{F}$ of itemsets
p, q	distributions
p^*, p^{ME}	distribution derived using Maximum Entropy principle
$E_p[X]$	mean of X with respect to p
$\text{Std}_p[X]$	standard variation of X with respect to p
$\text{Var}_p[X]$	variance of X with respect to p
$\text{Cov}_p[X]$	covariance matrix of X with respect to p

$\mathbb{P}(X)$	probability of an event X
$p(B = \omega)$	probability $p(b_1 = \omega_1, \dots, b_L = \omega_L)$
$p(B = 1)$	probability $p(b_1 = 1, \dots, b_L = 1)$
$\mathcal{E}(p)$	entropy of distribution p
$\text{KL}(p; q)$	Kullback-Leibler divergence between p and q
$\mathcal{P}(S, \theta)$	space of distributions satisfying $\text{E}[S] = \theta$
$d_{CM}(D_1, D_2; \mathcal{F})$	constrained minimum (CM) distance between D_1 and D_2 based on itemsets $\mathcal{F}$
$\text{fi}(Q; \mathcal{F}, \theta)$	frequency interval of Q derived from the frequencies θ of itemsets $\mathcal{F}$
$\text{cd}_A(D; \alpha_1, \alpha_2)$	correlation dimension of D with α_1 and α_2 tail cuts
$\text{cd}_R(D; r_1, r_2)$	correlation dimension of D calculated using radii r_1 and r_2
Z_D	distance between two random points from D
$\text{acd}(D)$	approximative correlation dimension of D , $\text{acd}(D) = \text{E}[Z_D] / \text{Std}[Z_D]$
$\text{ind}(D)$	a data set having equal margins as D but independent attributes
$\text{ind}(L, s)$	a data set having L independent attributes with margins equal to s
$\text{ncd}_A(D; \alpha_1, \alpha_2)$	normalised correlation dimension of D with α_1 and α_2 tail cuts

List of Publications

This thesis consists of an introduction and the following papers:

- I Nikolaj Tatti. Distances between data sets based on summary statistics. *Journal of Machine Learning Research*, 8:131–154, Jan 2007.
- II Nikolaj Tatti. Computational complexity of queries based on itemsets. *Information Processing Letters*, pages 183–187, June 2006.
- III Nikolaj Tatti. Safe projections of binary data sets. *Acta Informatica*, 42(8-9):617–638, April 2006.
- IV Nikolaj Tatti. Maximum Entropy Based Significance of Itemsets. Accepted for publication in *Knowledge and Information Systems (KAIS)*.
- V Nikolaj Tatti, Taneli Mielikäinen, Aristides Gionis, and Heikki Mannila. What is the dimension of your binary data. In *Proceedings of Sixth IEEE International Conference on Data Mining (ICDM 2006)*, pages 603–612, 2006.

We will use this numbering throughout the thesis.

Preface

I've been lucky in many ways. One of the greatest privileges I had over past years is that I have been able to do the work that I wanted to do. In my case that work is being an academic researcher. Over the years I have learned to fully appreciate this fact. To me being a researcher isn't about the money, nor it is the fame. In the end it is a simple joy of solving puzzles, seeing order in chaos. Curiosity, if you will.

Having said that, it is obvious that I am most grateful to people and institutions that have helped me during my journey. I did my work at the Laboratory of Computer and Information Science (CIS) at the Helsinki University of Technology.¹ I am also affiliated with the Basic Research Unit of the Helsinki Institute for Information Technology (HIIT BRU). I was funded by ComMIT graduate school and the Academy of Finland. I am also grateful for a personal grant given by the Foundation of Technology (TES).

I would like to thank my advisor Heikki Mannila whose vision, something that I deeply admire, has provided me with a direction and guidance. Jaakko Hollmén has always had time for my problems and has always cheered me with his upbeat spirit. I am also grateful for the help I received from Jouni K. Seppänen whose calm and down-to-earth attitude I greatly respect. In addition, I would like to thank Aris Gionis and Taneli Mielikäinen with whom I had the pleasure to collaborate. Thanks are also due to Dr. Szymon Jaroszewicz from the National Institute of Telecommunications and Prof. Bart Goethals from University of Antwerp who have reviewed the manuscript of the thesis and provided excellent and insightful comments.

¹currently part of the Department of Information and Computer Science (ICS).

In addition, I would like to thank Antti Rasinen, Hannes Heikinheimo, Antti Ukkonen, Kai Puolamäki, Heli Hiisilä, Janne Toivola, and Mikko Korpela. I would also like to thank Gemma Garriga who has helped me greatly by reading my thesis and providing comments that significantly improved the readability of the work. My deepest thanks are due to my close friend Anne Patrikainen who originally inspired me to begin working at CIS.

В заключение я хочу поблагодарить мою семью, маму, папу, Валеру и Аниту. Вы для меня важнее всех во всем свете.

Otaniemi, April 2008

Nikolaj Tatti / Николаï Тамму

Chapter 1

Introduction

There's a war out there, old friend.
A world war. And it's not about
who's got the most bullets. It's
about who controls the information.
What we see and hear, how we
work, what we think... it's all about
the information!

Cosmo, Sneakers

On data mining. It is appropriate to begin this work by discussing goals and origin of data mining. One of the possible definitions of the field is the following.

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

[HMS01]

The need for data mining comes from the gap between the traditional statistics and computer science: The focus in the traditional statistics is testing hypotheses, the computation involved is usually ignored. On the other hand, in computer science we are focused on fast computing but statistical analysis of the underlying data does not play any major role. However, there is a growing need for doing both simultaneously. Modern, constantly improving technology has enabled us to store huge data sets, both in academia and in industry [HAK⁺02, KRS02]. These databases are large enough so that manual exploration is infeasible. Hence, we need to either summarise data or explore data automatically for useful statistically relevant information.

Soda	SodaLight	Diapers	Beer
0	1	1	1
1	0	1	1
0	1	0	0
1	0	0	1
1	1	0	0

Table 1.1: A toy example of market basket data with 5 transactions and 4 items.

Data mining, however, is more than simply statistics for large data sets; namely, the actual mining process is involved. In traditional hypothesis test we already know what we are testing whereas in a typical data mining scenario we do not know what are our hypotheses. The novelty of data mining is to find possibly interesting information from data — statistics only provides means for testing whether our acquired information is statistically significant.

The aforementioned definition of the field allows us to divide the data mining techniques into two categories: global methods and local methods. The goal of global methods is to summarise the data as a whole. Prime examples of such techniques are decision trees [Qui93], graphical modelling [Jor99], clustering tasks [DHS00], principal component analysis [Hay98], and graph analysis [BRRT05]. On the other hand, local methods concentrate finding patterns in data. Thus, the goal is not to cover all data but only interesting parts of it. Classic examples of such methods are association rules [AIS93, AMS⁺96] and episodes [MTV97].

The division of the field into local and global methods is not crisp. For instance, clustering can be viewed as an attempt to model the whole data. However, a single cluster can also be considered as a pattern explaining only a portion of data. Similarly, a collection of local patterns can be viewed as a summary for the complete data set. A large portion of this thesis concentrates on using itemsets, local patterns for binary data, as a summary of the original binary data.

On mining of binary data. Perhaps the most classical example of binary data is market basket data (see e.g. [BSVW99]). A toy example of a such data set is given in Table 1.1. Such data consists of transactions represented by binary vectors. An element of a transaction describes whether the customer bought the corresponding product. For instance, in Table 1.1 transactions 2, 4, and 5 contain Soda.

Association rules are one of the most classical methods for analysing binary data [AIS93, AMS⁺96]. These are statements of type ‘If a customer buys product X, then with a high probability he also buys product Y’. For instance, we may

Itemset	# of customers
Soda	3
SodaLight	3
Diapers	2
Beer	3
Soda, SodaLight	1
Diapers, Beer	2

Table 1.2: A toy example of a collection of itemsets as a surrogate for the data set given in Table 1.1.

conclude from our toy data that customers buying diapers also buy beer.¹ A more fundamental patterns for the analysis are itemsets. Itemsets are sets of products. For instance, in our toy data these are 'Soda', 'SodaLight', 'Soda, SodaLight', and so on. To each itemset we assign a number which we call support of frequency. This is the number of transactions in which every product included in the itemset occurs. For instance, in our toy data we have 3 customers buying Soda and 3 customers buying SodaLight but only one customer buying both. Thus the support for 'Soda' is 3, but the support for 'Soda, SodaLight' is only 1.

In addition to market basket analysis, binary data occur in a wide range of scenarios, such as bag-of-words representations of text documents [BFS03], fossil occurrence at paleontological sites [For05], geographical co-occurrence [HFEM07] of mammals, traffic accident reports [GWBV03], click-stream data [KBF⁺00], co-authorship and citations in scientific papers [BMS02, GKM03].

Topic of the thesis. A large portion of the thesis focuses on using itemsets as a surrogate for the original data set. In other words, imagine that instead of a binary data set given in Table 1.1 you are provided with a collection of itemsets presented in Table 1.2. The idea behind this scenario is that the itemsets capture the essential information from the original data set. Note that in our toy example we are not able to fully construct the original data set from the itemsets, that is, there are other data sets that produce equivalent itemsets. This is an important observation since this enables using this scenario in privacy-preserving data mining.

Once we have replaced the original data with a family of itemsets, many simple tasks become difficult. For instance, it is easy to deduce from Table 1.1 that we have one customer buying Beer and SodaLight. On other hand, deducing the

¹This particular example is actually an infamous urban legend in the data mining community, <http://web.onetel.net.uk/~hibou/Beer%20and%20Nappies.html>

number of customers buying Beer and SodaLight using Table 1.2 is surprisingly difficult. In fact, we will show that in general solving such a query problem takes an exponential amount of time. However, by imposing some restrictions we are able to ease the computational burden.

We can use the family of itemsets to estimate the frequencies of unknown itemsets. For instance, we can estimate that the number of customers buying Beer and SodaLight is $3 \times 3/5 = 9/5$. By computing the difference between the estimate and the actual support we receive a number telling how surprising is the itemset. This number can be used for ranking itemsets, that is, the more we are surprised by the support of the itemset, the more important the itemset is.

We can also use itemsets for computing the difference between two entire data sets. Assume that we have market basket data from two different months. We can compute some selected family of itemsets from the data sets and compute the difference by comparing the supports of the itemsets. Defining the distance between data sets allows us treat data sets, highly complex objects, as units and enables us to use traditional data mining tools.

Our last topic of the thesis deals with defining an intrinsic dimension for binary data. Traditionally, the dimension is understood to be the number of columns in the data set, usually a very high number. On the other hand, binary data is usually sparse and contains structure and hence is less complex than the number of columns would suggest. One possible way of defining intrinsic dimension is to use fractal dimension which is often used with real-valued data sets. We study how the fractal dimension can be applied to binary data.

Contributions of the thesis. This thesis is based on the publications listed in List of Publications. The content of the papers is discussed in greater detail at the beginning of Chapters 6, 7, and 8.

In Publication II we show that certain query problems, namely predicting itemsets from a set of known itemsets, are infeasible. Problems of the same type are well-studied [Cal04, Cal03], but we focus more on downward closed families of itemsets. We use a construction similar to [Coo90] to show that even the restriction of being downward closed does not prevent the problem from being infeasible.

The query theme continues in Publication III where we provide a novel optimisation scheme for solving a classic query problem [Hai65, Cal03, BSH04] by applying theorems from Markov Random Field theory [CDLS99].

In Publication IV we introduce a method for ranking itemsets by comparing the observed frequency against a Maximum Entropy estimate [PMS03]. Our work can be seen as an extension of the approach in [BMS97] in which the observed value is compared against the independence assumption.

In Publication [I](#) we study the idea of computing the distance between data sets via itemset frequencies. Similar approaches has been suggested for example in [\[HSM03\]](#). However, we show that our distance possesses many theoretical properties, some of them being unique among alternative distances.

In Publication [V](#) we apply correlation dimension, a well-known concept [\[Bar88, Ott97\]](#), to binary data sets. Our investigation leads to a novel idea, called normalised correlation dimension, that takes into account the unique nature of binary data.

Contributions of the author. The author of the thesis is the sole author of Publications [I](#), [II](#), [III](#), and [IV](#).

In Publication [V](#) the author collaborated with the co-authors of the paper. The theoretical analysis of the paper is a joint work of the author and H. Mannila. The author is responsible for introducing the concept of the normalised correlation dimension. The author implemented all the experiments which were designed jointly with H. Mannila, T. Mielikäinen and A. Gionis. The paper was written jointly with the co-authors.

The structure of the thesis. The purpose of the subsequent chapters is to provide the needed background mathematics used in the articles so that a reader with a reasonable knowledge in statistics, calculus and algebra will be able to follow the articles. The chapters also summarise the articles, emphasising heavily the theoretical side. We also review the research related to our ideas.

The first four chapters focus solely on background mathematics. They provide a sound base for the three remaining chapters which describe the key theorems in the articles. The style of the thesis is a traditional definition-theorem-example approach. The dependencies of theorems presented in introduction is given in [Figure 1.1](#).

In [Chapter 2](#) we introduce the notation and basic concepts related to binary data and itemsets. In [Chapter 3](#) we introduce basic theory of Linear Programming. In [Chapter 4](#) we study Markov Random Fields and in [Chapter 5](#) we introduce Kullback-Leibler divergence and Maximum Entropy principle. In [Chapter 6](#) we discuss the problem of predicting itemset frequencies from a known set of itemsets. We also introduce a rank measure for itemsets that uses information available from the sub-itemsets. In [Chapter 7](#) we introduce the idea of using itemsets for computing a distance between two binary data sets. Finally, in [Chapter 8](#) we use concepts from fractal theory for defining an effective dimension of a binary data set. Proofs for some theorems are provided in [Appendix](#).

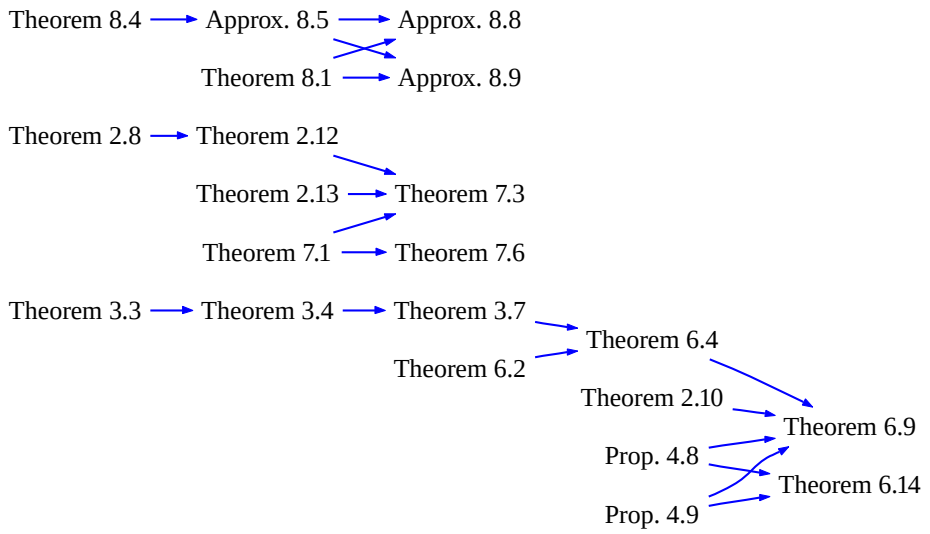


Figure 1.1: Dependency graph of theorems presented in the thesis.

Chapter 2

Binary data

The world isn't run by weapons
anymore, or energy, or money. It's
run by little ones and zeroes, little
bits of data. It's all just electrons.

Cosmo, Sneakers

Extracting patterns from binary data is an active subfield of data mining. The most popular patterns are itemsets, sets of columns, that have unusually high concentration of 1s. Originally, itemsets were used as intermediate result for finding association rules [AMS⁺96, AIS93]. Nowadays, they are considered to be interesting patterns on their own and they have other applications in addition to association rules [MT96]. One major factor for the popularity of itemsets is their anti-monotonicity property which allows using level-wise mining algorithms, for example, APRIORI [AMS⁺96].

The main theme of the thesis is to use itemsets as a surrogate for the original binary data. We can think that the mined itemsets act like a summary of the original data.

2.1 Binary data and Itemsets

We begin by defining basic concepts related to mining binary data. A *binary data set* is a finite multiset of a binary vectors of length K . In other words, it is a collection of elements from a space $\Omega = \{0, 1\}^K$. A single element of a data set is called a *transaction*. The space Ω is called *sample space* and K is the *dimension* of Ω . The number of binary vectors in a data set D is denoted by $|D|$. It is customary to visualise binary data as a binary matrix of size $|D| \times K$

		attributes		
		a_1	a_2	a_3
transactions	{	1	0	0
		0	1	0
		0	0	1
		1	1	0
		0	1	1
		0	1	1

Table 2.1: An example of a binary data set represented as a binary matrix. The data contains $K = 3$ attributes, namely a_1 , a_2 , and a_3 , and 6 transactions.

(see Table 2.1 for such an example). In this matrix the rows are the transactions. Note that for our purposes the order of the transactions is irrelevant.

Let $\omega \in D$ be a randomly selected binary vector from D . We define an *attribute* a_i to be the boolean random variable representing the i th component of ω . If the data is represented as a matrix (see Table 2.1), then the attributes represent the columns of the matrix. We set $A = \{a_1, \dots, a_K\}$ to be the collection of all attributes.

In the context of this work it is more convenient to talk about distributions rather than data sets. We can represent a data set D by an *empirical distribution* p_D defined on a sample space Ω by setting

$$p_D(a_1 = \omega_1, \dots, a_K = \omega_K) = \frac{\text{number of elements in } D \text{ equal to } \omega}{|D|},$$

where $\omega \in \Omega$ is a binary vector of length K , ω_i is the i th element of ω , and a_i is the corresponding attribute, a boolean random variable representing the i th dimension of the data set.

Example 2.1. Consider the data set given in Table 2.1. The data set has 3 attributes $A = \{a_1, a_2, a_3\}$ and 6 transactions. The corresponding empirical distribution is

$$\begin{aligned} p_D(a_1 = 0, a_2 = 0, a_3 = 0) &= 0, & p_D(a_1 = 0, a_2 = 0, a_3 = 1) &= 1/6, \\ p_D(a_1 = 0, a_2 = 1, a_3 = 0) &= 1/6, & p_D(a_1 = 0, a_2 = 1, a_3 = 1) &= 1/3, \\ p_D(a_1 = 1, a_2 = 0, a_3 = 0) &= 1/6, & p_D(a_1 = 1, a_2 = 0, a_3 = 1) &= 0, \\ p_D(a_1 = 1, a_2 = 1, a_3 = 0) &= 1/6, & p_D(a_1 = 1, a_2 = 1, a_3 = 1) &= 0. \end{aligned}$$

□

We use some convenient abbreviations. Given a distribution p defined on Ω , a set of attributes $B = \{b_1, \dots, b_L\} \subseteq A$,¹ and a binary vector ω of length L , we use $p(B = \omega)$ for $p(b_1 = \omega_1, \dots, b_L = \omega_L)$. By writing $p(B = 1)$ we mean $p(B = \omega)$, where ω is a vector containing only 1s.

Example 2.2. Consider the distribution p_D given in Example 2.1. Then, for instance, we have

$$\begin{aligned} p_D(a_1 a_2 = 1) &= 1/6, & p_D(a_1 = 1) &= 1/3, \\ p_D(a_2 a_3 = 1) &= 1/3, & p_D(a_3 = 1) &= 1/2. \end{aligned}$$

□

One of the most useful properties of binary data is that we can apply Boolean logic. Assume that we are given a Boolean formula F defined on (a subset of) the attributes A . Let $S_F : \Omega \rightarrow \{0, 1\}$ be an *indicator function*,

indicator function

$$S_F(\omega) = \begin{cases} 1 & \omega \text{ satisfies } F, \\ 0 & \text{otherwise.} \end{cases}$$

Given a distribution p , the *frequency* θ_F of F is then the probability of S_F being 1, that is, it is the mean $\theta_F = \mathbb{E}_p[S_F]$.

Example 2.3. Consider the data set given in Table 2.1. Consider the formulae $F_1 = a_1$, $F_2 = a_2$, $F_3 = a_3$, $F_4 = a_1 \wedge a_2$, and $F_5 = a_2 \wedge a_3$. For example, the indicator function of F_5 is

$$S_{F_5}(\omega) = \begin{cases} 1 & \omega_2 = \omega_3 = 1, \\ 0 & \text{otherwise.} \end{cases}$$

There are 2 transactions that satisfy F_5 , namely transactions 5 and 6, hence the frequency for F_5 is $\theta_{F_5} = \frac{1}{3}$. Similarly, the frequencies for the rest of the formulae are $\theta_{F_1} = \frac{1}{3}$, $\theta_{F_2} = \frac{4}{6}$, $\theta_{F_3} = \frac{1}{2}$, and $\theta_{F_4} = \frac{1}{6}$. □

Given a formula F , a frequency θ , we say that a distribution p *satisfies the frequency* θ if $\mathbb{E}_p[S_F] = \theta$. We can easily extend these definitions for multiple Boolean formulae. If we have a family $\mathcal{F} = \{F_1, \dots, F_N\}$ of Boolean formulae, then the frequencies $\theta_{\mathcal{F}} = (\theta_{F_1}, \dots, \theta_{F_N})$ is a vector containing N elements. Similarly, the indicator function is $S_{\mathcal{F}} : \Omega \rightarrow \{0, 1\}^N$, defined as $S_{\mathcal{F}}(\omega) = (S_{F_1}(\omega), \dots, S_{F_N}(\omega))$.

¹We will rather use $\{b_1, \dots, b_L\}$ instead of cumbersome $\{a_{i_1}, \dots, a_{i_L}\}$.

Example 2.4. We continue Example 2.3. Consider a family of formulae

$$\mathcal{F} = \{a_1, a_2, a_3, a_1 \wedge a_2, a_2 \wedge a_3\}.$$

Example 2.3 tells us that the corresponding frequencies are $\theta_{\mathcal{F}} = \frac{1}{6}(2, 4, 3, 1, 2)$. We can easily see that the empirical distribution p_D given in Example 2.1 satisfies the frequencies since $E_{p_D}[S_{\mathcal{F}}] = \theta_{\mathcal{F}}$. The distribution p_D is not the only distribution satisfying $\theta_{\mathcal{F}}$. In fact, there are infinite number of such distributions. For example, a distribution q defined as

$$\begin{aligned} q(a_1 = 0, a_2 = 0, a_3 = 0) &= 1/6, & q(a_1 = 0, a_2 = 0, a_3 = 1) &= 1/6, \\ q(a_1 = 0, a_2 = 1, a_3 = 0) &= 1/6, & q(a_1 = 0, a_2 = 1, a_3 = 1) &= 1/6, \\ q(a_1 = 1, a_2 = 0, a_3 = 0) &= 1/6, & q(a_1 = 1, a_2 = 0, a_3 = 1) &= 0, \\ q(a_1 = 1, a_2 = 1, a_3 = 0) &= 0, & q(a_1 = 1, a_2 = 1, a_3 = 1) &= 1/6. \end{aligned}$$

also satisfies the frequencies. □

Our special interest lies in conjunctive formulae, that is, formulae having the form $b_1 \wedge \dots \wedge b_L$. Such formulae are called *itemsets* and they are usually represented by a subset of attributes $B = \{b_1, \dots, b_L\}$. Often the condensed notation is used $B = b_1 \dots b_L$. Note that if B is an itemset, then the frequency θ_B can be expressed in a form $p(B = 1)$.

Itemsets possess many useful properties, the most important one is the anti-monotonic property:

Proposition 2.5. Assume that we are given two itemsets U and V such that $U \subseteq V$, then the frequencies obey $\theta_U \geq \theta_V$.

One of the largest research areas in binary data mining is retrieving σ -frequent itemsets. In other words, given a data set (or a distribution), the frequency θ_B of an itemset B is σ -frequent if $\theta_B \geq \sigma$. A family $\mathcal{F}$ of itemsets is said to be *downward closed* or *antimonotonic* if each subset of each member of $\mathcal{F}$ is also included in $\mathcal{F}$. Proposition 2.5 says that a family containing all σ -frequent itemsets is downward closed.

Example 2.6. The $\frac{1}{3}$ -frequent itemsets in Example 2.3 are $\emptyset$, a_1 , a_2 , a_3 , and a_2a_3 . Note that this family is downward closed. □

Our main interest is to study the properties of downward closed families of itemsets. The fundamental property of such families is that we are able to express the frequency of a Boolean formula as a linear combination of frequencies of itemsets. This is illustrated in the following example.

Example 2.7. Let $B = a_1 \vee a_2$ be a disjunctive formula of two attributes. Let $\mathcal{F}$ be a family of itemsets $\mathcal{F} = \{a_1, a_2, a_1 a_2\}$. We can express the indicator function S_B as a linear combination of indicator functions of itemsets:

$$S_B = S_{a_1} + S_{a_2} - S_{a_1 a_2}.$$

Given a distribution p we can express the frequency of H as

$$\theta_B = \mathbb{E}_p[S_B] = \mathbb{E}_p[S_{a_1}] + \mathbb{E}_p[S_{a_2}] - \mathbb{E}_p[S_{a_1 a_2}] = \theta_{a_1} + \theta_{a_2} - \theta_{a_1 a_2}.$$

□

The following theorem states the property of downward families of itemsets used in the previous example.

Theorem 2.8 (Proposition 1 in [MT96]). *Let $\mathcal{F}$ be a downward closed family of itemsets along with the frequencies $\theta_{\mathcal{F}}$. Let p be a distribution satisfying $\theta_{\mathcal{F}}$. Let B be a Boolean formula and let S_B be its indicator function. If we assume that B depends only on variables that are contained in some member of $\mathcal{F}$, then there is a set of constants $\{u_C\}$ not depending of $\theta_{\mathcal{F}}$ such that*

$$\theta_B = \mathbb{E}_p[S_B] = \sum_{C \subseteq B} u_C \theta_C,$$

where C ranges over all subsets of B .

More generally, if $\mathcal{B} = \{B_1, \dots, B_M\}$ is a collection of Boolean formulae such that a formula B_i depends only on variables containing in some member of $\mathcal{F}$, then there is a matrix U of size $|\mathcal{B}| \times |\mathcal{F}|$ not depending of $\theta_{\mathcal{F}}$ such that

$$\theta_{\mathcal{B}} = \mathbb{E}_p[S_{\mathcal{B}}] = U \theta_{\mathcal{F}}.$$

The immediate corollary of Theorem 2.8 states that certain marginal distributions obtained from a distribution satisfying the frequencies are unique. We illustrate this with the following toy example.

Example 2.9. Consider two attributes a and b and let their frequencies be $\theta_a = 0.5$ and $\theta_b = 0.6$. Assume two distributions p and q being

$$\begin{aligned} p(a=1, b=1) &= 0.5, & p(a=0, b=1) &= 0.1, \\ p(a=1, b=0) &= 0.0, & p(a=0, b=0) &= 0.4 \end{aligned}$$

and

$$\begin{aligned} q(a=1, b=1) &= 0.3, & q(a=0, b=1) &= 0.3, \\ q(a=1, b=0) &= 0.2, & q(a=0, b=0) &= 0.2. \end{aligned}$$

Although p and q are different distributions, they both satisfy the itemsets a and b . This implies that

$$\begin{aligned} p(a=1) &= q(a=1), & p(a=0) &= q(a=0), \\ p(b=1) &= q(b=1), & p(b=0) &= q(b=0). \end{aligned}$$

In other words, p and q are equal when they are marginalised to a (or to b). $\square$

We generalise the preceding example in the following corollary of Theorem 2.8.

Corollary 2.10. *Let $\mathcal{F}$ be a downward closed family of itemsets and let $\theta_{\mathcal{F}}$ be the corresponding frequencies. Let $B = b_1 \cdots b_L \in \mathcal{F}$ be an itemset from $\mathcal{F}$. If p and q satisfy the frequencies $\theta_{\mathcal{F}}$, then $p(B = \omega) = q(B = \omega)$ for any ω . In other words, the distribution obtained by ignoring the attributes outside B from a distribution p is unique.*

This corollary combined with the theory of Markov Random Fields (Chapter 4) will play a crucial role in Chapter 6.

In addition to itemsets, there are also other families of Boolean functions that satisfy Theorem 2.8. A *parity formula* $B = b_1 \oplus \cdots \oplus b_L$, where $\oplus$ is the XOR-operator, returns 1 if and only if an odd number of the variables b_i are equal to 1. We can express parity functions as a linear combination of conjunctive functions and visa versa.

Example 2.11. Let us continue Example 2.3. Consider the following parity functions $H_1 = a_1$, $H_2 = a_2$, $H_3 = a_3$, $H_4 = a_1 \oplus a_2$, and $H_5 = a_2 \oplus a_3$ and let their frequencies be $\theta_{\mathcal{H}} = \frac{1}{6}(2, 4, 3, 4, 3)$. We know that

$$S_{a \oplus b} = S_a + S_b - 2S_{ab}.$$

This implies that

$$\theta_{a \oplus b} = E[S_{a \oplus b}] = E[S_a] + E[S_b] - 2E[S_{ab}] = \theta_a + \theta_b - 2\theta_{ab}.$$

We can restate this connection by using vector notation. Let U be

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & -2 & 0 \\ 0 & 1 & 1 & 0 & -2 \end{bmatrix}.$$

Recall that the frequencies in Example 2.3 were $\theta_{\mathcal{F}} = \frac{1}{6}(2, 4, 3, 1, 2)$. By multiplying $\theta_{\mathcal{F}}$ with U we obtain a vector

$$U\theta_{\mathcal{F}} = \frac{1}{6}(2, 4, 3, 4, 3)$$

that corresponds to the parity frequencies $\theta_{\mathcal{H}}$. It is important to note that U is invertible, that is, we can transform parity frequencies into itemset frequencies by a linear transformation. $\square$

The following proposition generalises the preceeding example.

Proposition 2.12. *Let $\mathcal{F}$ be a downward closed family of itemsets. Define*

$$\mathcal{H} = \{b_1 \oplus \dots \oplus b_L; B = \{b_1, \dots, b_L\} \in \mathcal{F}\}$$

to be the set of corresponding parity formulae. Then there is an invertible matrix U such that $\theta_{\mathcal{H}} = U\theta_{\mathcal{F}}$.

The proof of the theorem is provided in [Appendix A.1](#).

The theorem tells us that once we know the frequencies for itemsets, then we can deduce by linear transformation to parity formulae without making any additional queries from the data set. We can also deduce itemsets from parity formulae. In other words, parity formulae and itemsets contain essentially the same information. This idea turns out to be important in [Chapter 7](#) in which we study the distances between data sets.

The reason why we are interested in parity formulae is that the frequencies are very regular when computed from the uniform distribution. This regularity enables us to ease the computational burden when solving the distances discussed in [Chapter 7](#).

Proposition 2.13 (Lemma 8 in [Publication I](#)). *Let B_1 and B_2 be two parity formulae such that $B_1 \neq B_2$. Let p be the uniform distribution defined on Ω . Then, $E_p[S_{B_1}] = \frac{1}{2}$ and $E_p[S_{B_1}S_{B_2}] = \frac{1}{4}$.*

Chapter 3

Linear Programming for Predicting Itemset Frequencies

Our need for understanding Linear Programming (LP) stems from Chapter 6. In that chapter we study the problem of deducing the frequency of an itemset from a known set of itemsets. We provide a sneak peak of this scenario in the following example.

Example 3.1. Assume that we are given two attributes, namely a and b . Assume also that we have the frequency for a equal to 0.5 and, similarly, 0.6 for b . We wish to find a distribution p having the highest possible frequency $p(ab = 1)$ for the itemset ab while at the same time having $p(a = 1) = 0.5$ and $p(b = 1) = 0.6$. In this particular case the distribution is equal to

$$\begin{aligned} p(a = 1, b = 1) &= 0.5, & p(a = 0, b = 1) &= 0.1, \\ p(a = 1, b = 0) &= 0.0, & p(a = 0, b = 0) &= 0.4. \end{aligned}$$

Hence, the maximum frequency for ab is 0.5. □

We study the problems similar to the one given in Example 3.1 in Chapter 6. It turns out that these problems can be solved with LP.

Linear programming is perhaps the most classical constrained optimisation problem. The modern theory of LP was developed during the decades 1930–1950, however, the roots go as far as the 18th century. In those decades, the need for solving optimisation problems sprang from the industrial and military management, especially in the United States of America. World War II and the Great Depression had influences on these developments, as well as rapid development of computers. Perhaps the most important event, a breakthrough, is the invention of SIMPLEX, an algorithm for solving linear program, by George

Dantzig in 1947. Another famous event was the conference at the University of Chicago in 1949, arranged by Tjalling Koopmans under the sponsorship of the Cowles Commission for Research in Economics. At the conference the power of linear programming was demonstrated through different military and industrial applications. The articles of the conference are available in [Koo51]. A reader interested in the history of the development of Linear Programming Theory is advised to chapters 1–2 in [Dan63] and the introduction in [Koo51].

In this chapter we represent rudimentary theory of Linear Programming. In Section 3.1 we LP and analyse its solutions. In Section 3.2 we review some of widely known solving algorithms.

3.1 Theory

In this section we define the linear program and using geometrical intuition explain how the solution of the problem is found.

In the standard form of linear program, we are given a vector $c \in \mathbb{R}^N$, a $M \times N$ matrix A , and a vector $b \in \mathbb{R}^M$. Linear program involves in finding real vector $x \in \mathbb{R}^N$ such that the following optimisation problem is solved:

$$\begin{aligned} \min c^T x \\ Ax = b \\ x \geq 0 \end{aligned} \tag{3.1}$$

In other words, we are asked to minimise $c^T x$ under certain constraining conditions. A set containing the vectors x satisfying the constraints is called - *feasible set*. The problem given in Eq. 3.1 is known as LP in *standard form*. There are alternative ways of stating the same problem but they can be polynomially reduced into the standard form [PS98, Section 2.1].

Example 3.2. Let us consider the following linear program:

$$\begin{aligned} \min c_1 x_1 + c_2 x_2 + c_3 x_3 \\ x_1 + x_2 &= 1 \\ x_1 + 2x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

Here we have denoted x_i as the i th component of a vector $x \in \mathbb{R}^3$ and c_i as the i th component of a vector $c \in \mathbb{R}^3$. The feasible set of this program is a segment lying in $\mathbb{R}^3$ and having $(1, 0, 0)$ and $(0, 1, \frac{1}{2})$ as end points. The program is in standard form. By combining the conditions we can rewrite the problem in a simpler form

$$\min \left(c_1 - c_2 - \frac{1}{2}c_3 \right) x_1 + 2$$

$$1 \geq x_1 \geq 0.$$

We see that the solution is attained, depending on the sign of $c_1 - c_2 - \frac{1}{2}c_3$, either at $(1, 0, 0)$ or $(0, 1, \frac{1}{2})$, the end points of the feasible set. If $c_1 - c_2 - \frac{1}{2}c_3 = 0$, then c is orthogonal with the feasible set and any point has the minimal value. $\square$

From now on, we will assume that the feasible set is not empty and that there exists a finite solution.

We have seen from Example 3.2 that the optimal solution was always a corner point (a vertex) of the feasible set. In general, the feasible set is a polytope lying in $\mathbb{R}^N$. Scaling the vector c does not change the outcome of LP, hence we can assume that the length of c is 1. Let x be a vector in the feasible set. Then $c^T x$ is the length of the orthogonal projection of x into c (See Figure 3.1).

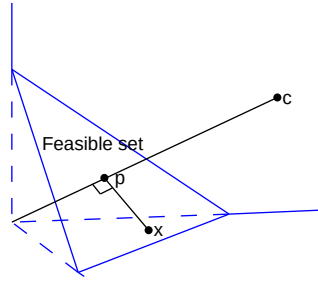


Figure 3.1: A geometrical interpretation of Linear Programming. The feasible set in this case is a triangle. If we assume that the vector c has length 1 and that x is a point from the feasible set, then $c^T x$ is the length of vector p , the orthogonal projection of x into c .

This geometrical interpretation reveals us the following theorem:

Theorem 3.3 (Theorems 2.4 and 2.6 in [PS98]). *There exists a vertex x of the feasible set of given LP such that x results the optimal value of LP.*

There exists a clever algebraic way of expressing the vertices of the feasible set. Consider the $M \times N$ matrix A in Eq. 3.1. We may safely assume that $N \geq M$ and that the rank of A is M . Otherwise, we can reduce the number of constraints, so that the condition holds.

Assume now that we are given a set $U = \{u_i\} \subseteq \{1, \dots, N\}$ of M integers. Let A_U be a submatrix of A containing only the columns corresponding to U . Assume that A_U is invertible. Let $x_{u_i} = A_U^{-1}b_i$, and 0 for the rest entries. Such x is called *basic solution*. The vector x satisfies $Ax = b$ in Eq. 3.1, but it is not guaranteed that x has only positive elements. However, if this is the case, then x is called *basic feasible solution* (BFS).

The following theorem shows that the vertices of the feasible set and BFS's are equivalent concepts.

Theorem 3.4 (Theorems 2.4 and 2.6 in [PS98]). *A basic feasible solution x is a vertex of the feasible set. In the other direction, if x is a vertex, then there is U such that $x_{u_i} = A_U^{-1}b_i$, and 0 for the rest entries. Consequently, there exists a basic feasible solution producing the optimal value for LP.*

Example 3.5. Consider the standard form of LP and assume that we have $N = 3$, $M = 1$, $A = [1, 2, 3]$, and $b = 1$. The feasible set is a triangle having the vertices $(1, 0, 0)$, $(0, \frac{1}{2}, 0)$, and $(0, 0, \frac{1}{3})$. Let $U = \{1\}$, $V = \{2\}$, and $W = \{3\}$. We have $A_U = 1$, $A_V = 2$, $A_W = 3$. Hence the basic solutions are $x = (1, 0, 0)$, $y = (0, \frac{1}{2}, 0)$, and $z = (0, 0, \frac{1}{3})$. These solutions are all feasible and they correspond to the vertices of the feasible area. $\square$

Example 3.6. Let us consider Example 3.2. The condition variables in this case are

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Let $U = \{1, 2\}$, $V = \{2, 3\}$, and $W = \{1, 3\}$. The submatrices are

$$A_U = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_V = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad A_W = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}.$$

The corresponding basic solutions are $x = (1, 0, 0)$, $y = (0, 1, \frac{1}{2})$, and $z = (1, 0, 0)$. We see that these solutions are feasible and they correspond to the vertices of the feasible set. $\square$

We have interest in investigating the complexity problems involved with linear programming. Hence, Theorem 3.4 has the following important corollary:

Theorem 3.7 (Lemma 2.1 in [PS98]). *There is a solution x for LP having only M non-zero elements. Also, if A and b contains rational elements expressed with L bits, then we can express a non-zero element of x in $f(M, L)$ bits, where f is a function of polynomial growth.*

In complexity theory we are often required to provide a polynomial-size certificate. Theorem 3.7 allows us to use the optimal solution x as a certificate (if we needed to) because we can express it in polynomial space.

3.2 Algorithms Solving Linear Programs

While the theory of Linear Programming is straightforward and simple, solving LP in practice is a complex task.

The most known algorithm for solving LP is SIMPLEX [Dan51]. Roughly *Simplex* put, the algorithm solves the problem by finding the best basic feasible solution using a hill-climbing approach. The algorithm is easy to implement and it works fairly well in practice. However, there is a major drawback: The number of basic feasible solutions (vertices of the feasible set) can be exponential. We can construct a problem such that finding the optimal solution using SIMPLEX requires an exponential number of steps [PS98, Section 8.6].

The question whether LP can be solved in polynomial time remained open until ELLIPSOID Algorithm was introduced in [Kha79]. The algorithm does solve *Ellipsoid Algorithm* the problem in polynomial time but it is highly complex and cumbersome. Hence, while this algorithm has important theoretical value, it is not used in practice.

A modern polynomial-time algorithm used for solving LP is called PRIMAL-DUAL PATH-FOLLOWING Algorithm [BSS93, Section 9.5]. The idea is that we *Primal-Dual Path-Following Algorithm* remove the condition $x \geq 0$ from Eq. 3.1 and instead of minimising $c^T x$ we minimise $c^T x - \mu \sum_j \log x_j$, where μ is some small constant. We can show that by letting μ approach 0, the solution of this modified problem approaches the optimal solution of the original LP problem.

Chapter 4

Markov Random Field Theory for Optimising Prediction of Itemset Frequencies

In Chapter 2 we considered distributions defined over a sample Ω , the set of all binary vectors of length K . Although, these distributions have finite number of elements, they are too large to work with. However, we are able to reduce the number of free parameters by using the techniques from Markov Random Field (MRF) theory.

Our interest in MRFs is somewhat unorthodox. We are mainly interested in decomposable distributions. Such distributions can be expressed efficiently — the MRF theory states that we need only the cliques of a certain graph. In Chapter 6 we use these decomposable distributions, and especially Proposition 4.9, to drastically ease the computational burden of one of our main tasks considered in this work. In order to justify our need for MRF theory even further we consider the following example.

Example 4.1. Let us consider Example 3.1. We have two attributes a and b with the corresponding frequencies 0.5 and 0.6, respectively. The maximum value for the frequency of ab is 0.5. Now consider adding a third attribute c with a frequency 0.3. We wish to find a distribution p maximising the frequency for ab and having $p(a = 1) = 0.5$, $p(b = 1) = 0.6$, $p(c = 1) = 0.3$. It turns out that the maximum frequency of ab is again 0.5. In fact, the frequency of c *does not play any role* in maximising the frequency for ab . To see this, note that we can expand any distribution satisfying itemsets a and b into a full joint distribution satisfying itemsets a , b , and c . $\square$

The reason that we were able to prune out the attribute c in Example 4.1

is that we did not have any constraints on itemsets ac and bc . We will see in Chapter 6 that MRF Theory provides a neat framework for identifying the attributes that can be pruned.

Frameworks for graph-based modelling in order to represent efficiently multivariate distributions were developed in the late 1980's [Pea88, LS90]. The causality between the attributes is expressed by a directed (acyclic) graph. Such concepts had led into research area called graphical modelling. Markov Random Fields (MRF) can be considered as undirected version of Bayesian networks.

4.1 Theory

In this section we introduce the basic concepts of Markov Random Fields. We explain how to obtain junction tree from the dependency graph and using the junction tree obtain a decomposition for certain distributions. For more detailed description on Markov Random Fields see e.g., [CDLS99].

A major part of Markov Random Field theory, and the part in which we are interested, is the way of expressing distributions effectively. To illustrate the situation, let us provide a simple example:

Example 4.2. Consider K binary variables a_i , $i = 1, \dots, K$. Without any assumptions, to express a joint distribution p of these variables, we need to store 2^K elements. On other hand, if we assume that the variables are independent, then we can express the distribution p as

$$p(a_1, \dots, a_K) = p(a_1) \cdots p(a_K). \quad (4.1)$$

To express such a distribution, we need to store only K elements. □

Generally speaking, consider that we have K binary random variables a_i . In this case, a joint distribution contains 2^K elements. However, if we make some assumptions (very similar to the independence assumption in Example 4.2), MRF theory allows us to express the distributions more succinctly.

Our interest is to study decomposable distributions. We have already seen one group of such distributions in Example 4.2 but the independence model is very strict. We will consider more general case by applying MRF concept.

Let us consider an undirected graph $G = (V, E)$ containing K nodes $V = \{v_i\}$; a node v_i represents the random variable a_i . We say that nodes v_i and v_j are connected if there is an edge (v_i, v_j) . The edges of the graph represent the dependencies between the variables a_i . Roughly put, an edge (v_i, v_j) tells us that we have some dependency between a_i and a_j and hence these variables should not be split in different components. Note that if the graph has no edges, then

connected nodes

this property is equal to the independence assumption demonstrated in Example 4.2. Our goal is to decompose p into components similarly to decomposition demonstrated in Eq. 4.1.

Our next goal is to make sure that the dependency graph is regular enough. In order to this, we need to introduce some concepts from graph theory. A *cycle* is a set of nodes $\{v_1, \dots, v_N\} \subseteq V$ such that v_j and v_{j+1} are connected, and v_1 and v_N are connected. Any possible additional edges between the nodes $\{v_1, \dots, v_N\}$ are called *chord* edges. A cycle without chord edges is called *chordless*. A graph is called *triangulated* if there are no chordless cycles.

cycle
chord
chordless
triangulated

Example 4.3. Consider the graph given in Figure 4.1(a). There is a chordless cycle $a-b-c-d$. This cycle can be removed, for example, by adding a chord edge $a-c$. We see that the resulting graph (Figure 4.1(b)) contains no chordless cycles and it is therefore triangulated. Note that this is not the only possible triangulation, adding a chord $b-d$ results also a triangulated graph.

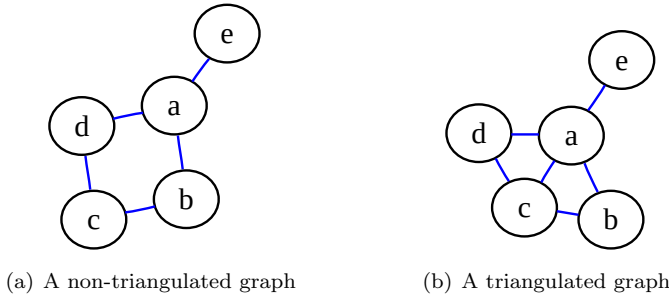


Figure 4.1: An example of a non-triangulated graph and a graph resulting from a triangulation process. In the left graph there is a chordless cycle $a-b-c-d$. This cycle is removed in the right graph by adding a chord $a-c$.

□

We will see that a triangulated graph possesses useful properties but let us consider how we can triangulate the graph G . The idea is to find the chordless cycles and add the missing edges until no chordless cycle can be found. A simple algorithm, called **ELIMINATION Algorithm**, iteratively picks a node v_i , connects its immediate neighbours, and delete the node [CDLS99, Section 4.4.1]. The graph G with the edges added during the elimination process is guaranteed to be triangulated.

*Elimination
Algorithm*

Let us assume that G is triangulated. Consider an undirected graph H having a node C_i for each clique of G . Two nodes C_i and C_j are connected in H if they

clique graph share a node in G . The graph H is called the *clique graph* of G . Let us fix $C_i \in V(H)$ and $C_j \in V(H)$ and assume that they are connected by an edge $S \in E(H)$. We associate a set of mutual nodes (lying in $V(G)$) $C_i \cap C_j$ to the *separator* edge S . This set is called a *separator*.

Example 4.4. Let us continue Example 4.3. The cliques of the graph in Figure 4.1(b) are abc , acd , and ae . The separators are $abc \cap acd = ac$, $abc \cap ae = a$, and $acd \cap ae = a$. The resulting clique graph H (given in Figure 4.2) is a triangle. In the figure, the oval nodes are the cliques and the square nodes are the separators.

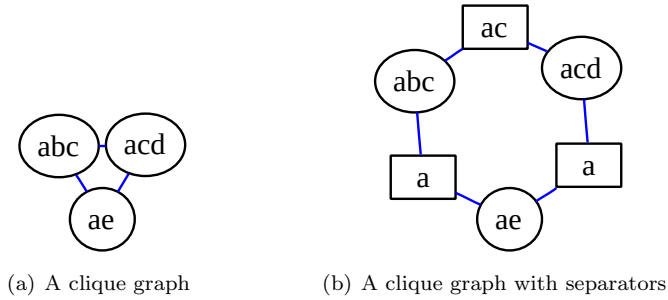


Figure 4.2: A clique graph of the graph G given in Figure 4.1(b). The oval nodes are the cliques of G and the square nodes are the separators, that is, intersections of the immediate cliques.

□

Assume for simplicity that the clique graph H is connected. Consider a spanning tree T of H . Select two cliques C_i and C_j sharing a mutual node v . We say that C_i and C_j have *running intersection property* if the intermediate cliques connecting C_i and C_j in T contain also v . If this property holds for any pair of cliques C_i and C_j sharing a mutual node, then we say that the tree T has the running intersection property. Such a tree is called *junction tree*. There can be several junction trees and not all spanning trees are junction trees.

Example 4.5. We continue Example 4.4. There are three possible spanning trees (illustrated in Figure 4.3) of the clique graph given in Figure 4.2. However, only two of these are junction trees. The tree in Figure 4.3(c) does not satisfy the running intersection property because the node c is not included in ae , a clique lying on a path between the cliques abc and acd .

□

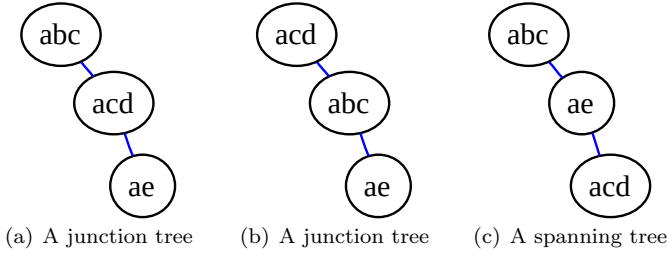


Figure 4.3: Spanning trees of the clique graph given in Figure 4.2. The tree in Figure 4.3(c) does not satisfy the running intersection property since the node c is not included in ae . The left and the centre tree are junction trees.

The following states that junction trees always exist, if G is triangulated.

Theorem 4.6 (Theorems 4.4 and 4.6 in [CDLS99]). *If the graph G is triangulated and connected, then there exists a junction tree T , that is, a spanning tree of the clique graph H satisfying the running intersection property.*

Let T be a junction tree. Let $C = \{C_1, \dots, C_N\}$ be the set of cliques (the nodes of the clique graph H) and let $\{S_1, \dots, S_{N-1}\}$ be the separators of the junction tree. We say that a distribution p is decomposable with respect to T if p has a form

*decomposable
distribution*

$$p(a_1, \dots, a_K) = \prod_{i=1}^N p(C_i) / \prod_{j=1}^{N-1} p(S_j).$$

There are several reasons why we are interested in decomposable distributions. The first reason is that storing such distributions requires less space as the following example demonstrates.

Example 4.7. We continue Examples 4.3–4.5. Assume that we have 5 binary variables and that the dependency graph is given in Figure 4.1. The number of elements in a general joint distribution is $2^5 = 32$. However, if p is decomposable with respect to tree given in Figure 4.3(a), then p has the form

$$\frac{p(a, b, c) p(a, c, d) p(a, e)}{p(a, c) p(a)}.$$

Since we can obtain the separator components $p(ac)$ and $p(a)$ from the clique components, we need to store only the clique components. The total number of elements in the components is $2^3 + 2^3 + 2^2 = 20$. $\square$

The second reason is seen in a way the decomposition takes into account the dependency edges of the graph G . If we have a fully connected set of nodes W in G , then there is a clique that contains W . The following theorem follows.

Proposition 4.8. *Let G be a graph and let T be its junction tree. Assume that W is a set of fully connected nodes in G . Then there is a component C in T such that $W \subseteq C$. Consequently, if the distribution is decomposed using T , there is a component in the decomposition containing W .*

Final (and the most important) reason is in the way we can compose a joint distribution from the components

Proposition 4.9. *Let T be a junction tree. Let $C = \{C_1, \dots, C_N\}$ be the corresponding set of cliques and let $\{S_1, \dots, S_{N-1}\}$ be the separators of T . Assume that for each cliques C_i we have a distribution p_i defined on C_i . Assume also that for two connected (in T) cliques C_i and C_j the components p_i and p_j are equivalent when marginalised to the separator. Then, there is a decomposable distribution p with respect to T such that marginalising p to C_i produces p_i . Moreover, p is equal to*

$$p(a_1, \dots, a_K) = \prod_{i=1}^N p_i(C_i) / \prod_{j=1}^{N-1} p_j(S_j).$$

The proof of the theorem is provided in Appendix A.2.

Example 4.10. We continue Example 4.7. Consider the following three distributions

$$\begin{aligned} p_1(abc = (0, 0, 0)) &= 0/8, & p_1(abc = (0, 0, 1)) &= 3/8, \\ p_1(abc = (0, 1, 0)) &= 1/8, & p_1(abc = (0, 1, 1)) &= 1/8, \\ p_1(abc = (1, 0, 0)) &= 0/8, & p_1(abc = (1, 0, 1)) &= 2/8, \\ p_1(abc = (1, 1, 0)) &= 0/8, & p_1(abc = (1, 1, 1)) &= 1/8, \\ \\ p_2(acd = (0, 0, 0)) &= 1/8, & p_2(acd = (0, 0, 1)) &= 0/8, \\ p_2(acd = (0, 1, 0)) &= 2/8, & p_2(acd = (0, 1, 1)) &= 2/8, \\ p_2(acd = (1, 0, 0)) &= 0/8, & p_2(acd = (1, 0, 1)) &= 0/8, \\ p_2(acd = (1, 1, 0)) &= 2/8, & p_2(acd = (1, 1, 1)) &= 1/8, \end{aligned}$$

and

$$\begin{aligned} p_3(ae = (0, 0)) &= 3/8, & p_3(ae = (0, 1)) &= 2/8, \\ p_3(ae = (1, 0)) &= 2/8, & p_3(ae = (1, 1)) &= 1/8. \end{aligned}$$

The distributions p_3 and p_2 match at the separator a

$$\begin{aligned} p_3(a = 0) &= p_2(a = 0) = 5/8, \\ p_3(a = 1) &= p_2(a = 1) = 3/8. \end{aligned}$$

Also, the distributions p_1 and p_2 match at the separator ac

$$\begin{aligned} p_1(ac = (0, 0)) &= p_2(ac = (0, 0)) = 1/8, \\ p_1(ac = (0, 1)) &= p_2(ac = (0, 1)) = 4/8, \\ p_1(ac = (1, 0)) &= p_2(ac = (1, 0)) = 0/8, \\ p_1(ac = (1, 1)) &= p_2(ac = (1, 1)) = 3/8. \end{aligned}$$

Theorem 4.9 now states that there is a distribution p having the marginals

$$p(a, b, c) = p_1(a, b, c), \quad p(a, c, d) = p_2(a, c, d), \quad p(a, e) = p_3(a, e).$$

□

In other words, as long as the component distributions match at the separators, we can join them to create a joint distribution. The total number of variables in the components is usually drastically smaller than the number of variables in a full joint distribution. These savings help us to reduce the computational burden of the problems introduced in Chapter 6.

Chapter 5

Kullback-Leibler Divergence for Ranking Itemsets

In this chapter we briefly review the theory related to information entropy and Kullback-Leibler divergence. The idea of information entropy in the context of communication theory was introduced introduced by Shannon in [Sha48], although the concept of thermodynamical entropy already existed in physics. Kullback and Leibler introduced Kullback-Leibler divergence in [KL51].

A concept that we are particularly interested in is the principle of maximum entropy which is discussed in Section 5.2. The idea was adopted from physics by Jaynes in [Jay57].

Concepts introduced in this chapter are used in Section 6.5. In that section we consider a ranking measure of itemsets by comparing the prediction made by Maximum Entropy against the actual value obtained from the data set. The difference is measured using Kullback-Leibler divergence. To motivate this chapter we provide the following sneak-peak example.

Example 5.1. We continue Example 3.1. Assume that we have two attributes a and b with the corresponding the frequencies 0.5 and 0.6, respectively. What value for the frequency of ab we expect to have? One approach is to consider the independence model, that is, the distribution p defined as

$$\begin{aligned} p(a = 1, b = 1) &= 0.5 \times 0.6, & p(a = 0, b = 1) &= (1 - 0.5) \times 0.6, \\ p(a = 1, b = 0) &= 0.5 \times (1 - 0.6), & p(a = 0, b = 0) &= (1 - 0.5) \times (1 - 0.6). \end{aligned}$$

This is, in fact, the Maximum Entropy distribution satisfying the itemsets a and b . Now consider seeing the actual frequency of ab and assume that it is equal to

0.5. The empirical distribution is equal in this case

$$\begin{aligned} q(a=1, b=1) &= 0.5, & q(a=0, b=1) &= 0.1, \\ q(a=1, b=0) &= 0.0, & q(a=0, b=0) &= 0.4. \end{aligned}$$

Our measure for the significance of ab is the difference between q and p , which is measured using Kullback-Leibler divergence. In this particular case, we have $\text{KL}(q; p) = 0.42$. $\square$

5.1 Definitions

In this section we will define Kullback-Leibler divergence, an asymmetric distance between two distributions, and the related quantity called entropy.

The distributions in this chapter are defined on the sample space Ω , a set of binary vectors of length K . However, we should point out that the concepts of this chapter work directly with any other finite space. The finiteness of Ω enables us to define a distribution as a function $p : \Omega \rightarrow [0, 1]$ mapping a point $\omega \in \Omega$ to a number between 0 and 1 such that $\sum_{\omega \in \Omega} p(\omega) = 1$. This naïve approach is adequate for our purposes but it should be kept in mind that the concepts introduced in this section can be expanded to arbitrary distributions.

entropy We define the *entropy* of a distribution p to be

$$\mathcal{E}(p) = - \sum_{\omega \in \Omega} p(\omega) \log p(\omega).$$

Here we use the natural logarithm and we use the convention $0 \cdot \log 0 = 0$.

Example 5.2. Consider the distribution q given in Example 5.1. The entropy is equal to

$$\mathcal{E}(q) = -0.5 \log(0.5) - 0.1 \log(0.1) - 0 \log(0) - 0.4 \log(0.4) = 0.94.$$

$\square$

Theorem 5.3 (Lemma 2.2.1 and Theorem 2.6.4 in [CT91]). *The entropy $\mathcal{E}(p)$ is a finite positive number. Among all the distributions defined on Ω , the uniform distribution has the largest entropy.*

Assume that we are given two distributions p and q . We say that q is *absolutely continuous* with respect to p if $p(\omega) = 0$ implies that $q(\omega) = 0$. Given p and q such that q is absolutely continuous with respect to p we define the *Kullback-Leibler divergence* to be

$$\text{KL}(q; p) = \sum_{\omega \in \Omega} q(\omega) \log \frac{q(\omega)}{p(\omega)}.$$

If q is not absolutely continuous with respect to p , then the divergence $\text{KL}(q; p)$ is defined to be infinite.

Example 5.4. The divergence between q and p given in Example 5.1 is

$$\text{KL}(q; p) = 0.5 \log \left(\frac{0.5}{0.3} \right) + 0.1 \log \left(\frac{0.1}{0.3} \right) + 0 \log \left(\frac{0}{0.2} \right) + 0.4 \log \left(\frac{0.4}{0.2} \right) = 0.42.$$

□

Theorem 5.5 (Theorem 3.1 in [Kul68]). *The divergence $\text{KL}(q; p)$ is a finite positive number if q is absolutely continuous with respect to p , and infinite otherwise. We have that $\text{KL}(q; p) = 0$ if and only if p equals to q .*

Assume that p is the uniform distribution, then

$$\text{KL}(q; p) = \sum_{\omega \in \Omega} q(\omega) \log \frac{p(\omega)}{|\Omega|^{-1}} = -\mathcal{E}(q) + \log |\Omega|.$$

Hence, we can see the entropy $\mathcal{E}(q)$ as a measure of closeness (in a Kullback-Leibler sense) of q to the uniform distribution: The higher the entropy, the closer is q to the uniform distribution.

The following theorem describes a possible interpretation of the values produced by the divergence.

Theorem 5.6 (Section 5.6 in [Kul68]). *Let $p(\omega; \theta)$ be a family of distributions parametrised by a vector $\theta \in \mathbb{R}^K$. Given θ_0 , let D be a collection of n independent points sampled from a distribution $p(\omega; \theta_0)$. Let θ_n be an estimate of θ_0 from D . Then, under some broad regularity conditions¹, we have that*

$$2n \sum_{\omega \in \Omega} p(\omega; \theta_n) \log \frac{p(\omega; \theta_n)}{p(\omega; \theta_0)}$$

converges weakly into a χ^2 distribution with K degrees of freedom, as n goes into infinity.

The theorem is stated but not proven in [Kul68]. We provide a proof for the case of a finite sample space Appendix A.3. The finite case is sufficient for our purposes.

According to the theorem we can use the quantity $2n\text{KL}(\theta_n; \theta_0)$ as a statistical test by comparing the P-value $\mathbb{P}(\chi^2(K) < 2n\text{KL}(\theta_n; \theta_0))$ to the selected risk threshold.

5.2 Maximum Entropy

Maximum Entropy Our next topic, *Maximum Entropy*, is closely related to the Kullback-Leibler divergence. Assume that we are given a function $S : \Omega \rightarrow \mathbb{R}^K$ mapping a sample $\omega \in \Omega$ into a vector of length K . Assume also that we are given a vector $\theta \in \mathbb{R}^K$.
satisfies We say that a distribution p satisfies θ if

$$\theta = \mathbb{E}_p[S] = \sum_{\omega \in \Omega} p(\omega) S(\omega),$$

that is, the mean of S taken with respect to p is equals to θ . We denote the set of all distributions satisfying θ by $\mathcal{P}(S, \theta)$, that is,

$$\mathcal{P}(S, \theta) = \{p; \mathbb{E}_p[S] = \theta\}.$$

Assume that the set $\mathcal{P}(S, \theta)$ is not empty. We denote the distribution from $\mathcal{P}(S, \theta)$ having the maximal entropy by p^* . Note that p^* depends on S and θ but we have ignored these variables from the notation for the sake of clarity. The
exponential form distribution p^* can be expressed with an *exponential form*.

Theorem 5.7 (Theorem 3.1 in [Csi75]). *Let $S : \Omega \rightarrow \mathbb{R}^K$ be a function and let θ be a vector of length K . Assume that $\mathcal{P}(S, \theta)$ is not empty and let p^* be the distribution maximising the entropy. There exists a vector $r \in \mathbb{R}^K$, a real number r_0 , and a set $Z \subseteq \Omega$ such that*

$$p^*(\omega) = \begin{cases} \exp(r_0 + r^T S(\omega)) & , \text{ if } \omega \notin Z \\ 0 & , \text{ if } \omega \in Z \end{cases}.$$

Moreover, for every $q \in \mathcal{P}(S, \theta)$ and $\omega \in Z$ we have $q(\omega) = 0$.

Example 5.8. Assume our sample space is the set of all binary vectors of length K , that is, $\Omega = \{0, 1\}^K$. Let $S(\omega) = \omega$. The mean $\theta = \mathbb{E}[S(\omega)]$ is now a vector containing the margins of the individual attributes, that is, $\theta_i = p(a_i = 1)$. Theorem 5.7 now states that p^* has the form

$$p^*(\omega) = \exp(r_0 + r^T S(\omega)) = \exp(r_0) \prod_{i=1}^K \exp(r_i \omega_i).$$

Define $p_i(a_i = 1) = \theta_i$ and $p_i(a_i = 0) = 1 - \theta_i$. It turns out that p_i is proportional to $\exp(r_i \omega_i)$. Thus we must have

$$p^*(\omega) = p_1(\omega_1) p_2(\omega_2) \cdots p_K(\omega_K).$$

This is the distribution related to the independence model. For example, the distribution p in Example 5.1 obeys the independence model. $\square$

In practice, the distribution p^* is solved using the ITERATIVE SCALING Algorithm. The algorithm was introduced in its modern form in [DR72]. However, the basic idea was introduced originally in [DS40], where the authors considered the problem of solving cell probabilities in the contingency tables. The algorithm applies Theorem 5.7 in the following way: Instead of exploring the space of distributions satisfying θ , the algorithm searches the distribution of an exponential form that satisfies θ . Theorem 5.7 guarantees that once such distribution is found, it will have the maximal entropy. The algorithm works in an iterative fashion: Assume that p is the current distribution, and let $\theta_{old} = E_p[S]$. The algorithm picks a new distribution q such that $\theta_{new} = E_q[S]$ is closer to θ than θ_{old} was [DR72, Theorem 1]. The step is repeated with q replacing p . Under certain conditions the algorithm can be speeded up considerably by decomposing the distributions [JP95].

*Iterative Scaling
Algorithm*

¹The exact conditions are stated in Appendix A.3.

Chapter 6

Predicting Itemset Frequencies

A large portion of the literature related to mining binary data deals with finding frequent itemsets or condensing them into smaller space. Itemsets can be viewed as a summary of the relevant information of the data set. We will focus on a scenario where the original data is replaced by a family of itemsets. Such an scenario is interesting theoretically and computationally, but it can also occur in practice in privacy-preserving data mining where the researcher has no access to the original data, instead he is given a family of itemsets. Namely, we consider a specific problem of finding the frequency of an unknown itemset from a set of known itemsets. This classic problem can be reduced to a linear program. Unfortunately, the solution is intractable since the program contains an exponential amount of variables. However, we can greatly reduce the number of variables by applying ideas from Markov Field Theory.

We begin by defining the query problem and reducing it to a linear program in Section 6.1. We point out in Section 6.2 that the query problem is intractable, unless $\mathbf{P} = \mathbf{NP}$. In Sections 6.3 and 6.4 we discuss how can we reduce the number of variables in the linear program. In Section 6.5 we define a framework for ranking itemsets based on the information available from sub-itemsets.

Contribution of the papers. This chapter is based on Publications II, III, and IV.

In Publication II we show that the query problems we are considering are intractable: Finding the possible frequencies of an itemset from a family of known itemsets is $\mathbf{NP}$ -hard. In the paper, the problems CONSISTENT and MAXQUERY can be seen as special case of FREQSAT, an $\mathbf{NP}$ -complete problem introduced in [Cal04, Cal03]. In FREQSAT, the constraining family of itemsets need not to be downward closed and we are also allowed to have inequality constraints. The proof for the $\mathbf{NP}$ -hardness of FREQSAT given in [Cal03] is actually a valid proof

for CONSISTENT although this is not explicitly mentioned. In a more general scenario we are allowed to have conditional first-order logic sentences as constraints and queries [Luk01]. In PSAT, a famous **NP**-complete problem, we are given a CNF-formula, a frequency for each clause, and we are asked to find a distribution satisfying these frequencies [GKP88]. The construction in the paper resembles the technique used in [Coo90], in which it is used to prove that the inference of Belief Networks is **NP**-hard.

The query problem can be reduced to linear program [Hai65]. The reduction, however, contains an exponential number of variables with respect to the number of attributes. To remedy this problem the attributes outside the query are ignored [BSH04, PMS03]. In Publication III we show that this may change the outcome. We develop a novel idea of safe sets, a set of attributes that are guaranteed to produce the right outcome for the query. We present a polynomial algorithm for finding the minimal safe sets. We also present a heuristic for finding restricted safe sets, that is, sets with a limited number of attributes. In our experiments, using restricted safe sets improves 10% of the queries in which the restricted safe set is larger than the actual query.

In Publication IV we study the idea of ranking itemsets based on their deviation from the prediction. Namely, we are given a set of known itemsets and a query itemset. We use the Maximum Entropy principle to predict the contingency table and compare it using Kullback-Leibler divergence against the actual contingency table obtained from the data set. Our prediction method is equivalent to the approach used in [PMS03].

Many measures has been suggested for ranking itemsets [AY98, Omi03, AIS93, GCB07, DP01], association rules [PS91, BMUT97, AIS93, JS02], and other related patterns [BMS97, HHM⁺07]. In many of these works the comparison is based on the independence model [BMS97, AY98, DP01, GCB07, PS91, BMUT97]. In some approaches the frequency is compared to a more flexible model. For instance, in [JS04, JS05] the frequency of an itemset is compared to an estimate obtained from the Bayes network. In [Meo00] the authors introduce the concept of dependence value, a difference between the actual frequency of an itemset and its Maximum Entropy estimate.

A special case of our framework results in a measure that ranks itemsets based on their deviation from the independence model. On the other hand, our proposal allows us to use richer models, such as, discrete Gaussian model. Our technique resembles greatly the active interestingness in [JS02] in which Kullback-Leibler divergence along with the Maximum Entropy principle is used for ranking association rules.

In a related work [HHM⁺07] the authors seek tree patterns with low entropy. In this approach interesting patterns would be those trees that have strong correlations compared to our approach in which interesting patterns would be the

contingency tables that cannot be modelled well by trees.

6.1 Definition of the Problem

Given a data set, solving a frequency of an itemset is a straightforward task, a single data scan. A more complicated scenario is when we are given, instead of the data set, a family of itemsets $\mathcal{F}$ along with their frequencies and we are asked to deduce the frequency of some unknown itemset, say, Q from $\mathcal{F}$. In this section we will discuss this particular problem and show how the task can be solved using Linear Programming.

We should point out immediately that generally deducing the frequency of the itemset Q from the family $\mathcal{F}$ of itemsets does not yield a unique solution. One can easily form data sets having different frequencies for Q but same frequencies for $\mathcal{F}$. Hence our interest is to find *all* possible frequencies of Q that can be produced by data sets having some specific frequencies for $\mathcal{F}$.

To continue our analysis let us rephrase the problem using the distributions. We will point out later that this transformation has no particular effect on the outcome. Given a family of itemsets $\mathcal{F}$ along their frequencies θ and a (query) itemset Q we define a *frequency interval* $\text{fi}(Q; \mathcal{F}, \theta)$ to be a set

$$\text{fi}(Q; \mathcal{F}, \theta) = \{p(Q = 1) ; p \text{ is a distribution satisfying } \theta\}$$

of possible frequencies of Q produced by distributions satisfying θ . Our goal in this section is to provide a method for solving this interval.

Example 6.1. Assume that we have two attributes a_1 and a_2 and that our family of itemsets consists of one-order itemsets $\mathcal{F} = \{a_1, a_2\}$. The corresponding frequencies are set to be $\theta = (\theta_{a_1}, \theta_{a_2})$, where $\theta_{a_1} = 0.6$ and $\theta_{a_2} = 0.7$. Let us calculate the frequency interval $\text{fi}(Q; \mathcal{F}, \theta)$ for $Q = a_1 a_2$.

We know that we must have

$$p(Q = 1) \leq \min \{p(a_1 = 1), p(a_2 = 1)\} = 0.6$$

and

$$p(Q = 1) \geq p(a_1 = 1) + p(a_2 = 1) - 1 = 0.3.$$

Let us define distributions p_1 and p_2 as

$$\begin{aligned} p_1(a_1 = 0, a_2 = 0) &= 0, & p_1(a_1 = 1, a_2 = 0) &= 0.3 \\ p_1(a_1 = 0, a_2 = 1) &= 0.4, & p_1(a_1 = 1, a_2 = 1) &= 0.3 \end{aligned}$$

and

$$\begin{aligned} p_2(a_1 = 0, a_2 = 0) &= 0.3, & p_2(a_1 = 1, a_2 = 0) &= 0 \\ p_2(a_1 = 0, a_2 = 1) &= 0.1, & p_2(a_1 = 1, a_2 = 1) &= 0.6 \end{aligned}$$

We see that p_1 and p_2 are genuine distributions. They also satisfy the frequencies θ :

$$\begin{aligned} p_1(a_1 = 1) &= 0.3 + 0.3 = 0.6 = \theta_{a_1} \\ p_1(a_2 = 1) &= 0.4 + 0.3 = 0.7 = \theta_{a_2} \\ p_2(a_1 = 1) &= 0 + 0.6 = 0.6 = \theta_{a_1} \\ p_2(a_2 = 1) &= 0.1 + 0.6 = 0.7 = \theta_{a_2} \end{aligned}$$

We have $p_1(Q = 1) = 0.3$ and $p_2(Q = 1) = 0.6$. Hence we know that $\max(\text{fi}(Q; \mathcal{F}, \theta)) = 0.6$ and $\min(\text{fi}(Q; \mathcal{F}, \theta)) = 0.3$. The discussion below shows us that we have $\text{fi}(Q; \mathcal{F}, \theta) = [0.3, 0.6]$. $\square$

Let us next analyse the frequency interval. Assume that two distributions p_0 and p_1 satisfy the given frequencies θ and that $p_0(Q = 1) = \eta_0$ and $p_1(Q = 1) = \eta_1$. Let a be a real number, $0 \leq a \leq 1$. Then a distribution $p_a = (1 - a)p_0 + ap_1$ satisfies the frequencies θ and $p_a(Q = 1) = (1 - a)\eta_0 + a\eta_1$. We conclude that $\text{fi}(Q; \mathcal{F}, \theta)$ is truly an interval and hence to solve this set we need to solve the extrema points.

To solve the right side of the interval $\text{fi}(Q; \mathcal{F}, \theta)$ we consider the following optimisation problem:

$$\begin{aligned} \max p(Q = 1) \\ p(F_i = 1) = \theta_i, \text{ for } i \in \{1, \dots, N\}. \end{aligned}$$

This problem resembles greatly linear program and we can transform this problem into a linear form. Note that the distribution p is defined on a sample space $\Omega = \{0, 1\}^K$ of binary vectors of length K . Let $\omega \in \Omega$ be a binary vector and let p_ω be the corresponding probability of p producing ω . Let Q be an itemset and let S_Q be the corresponding indicator function. We can formulate the optimisation problem in the following form

$$\begin{aligned} \max \sum_{\omega \in \Omega} S_Q(\omega) p_\omega \\ \sum_{\omega \in \Omega} S_{F_i}(\omega) p_\omega = \theta_i, \text{ for } i \in \{1, \dots, N\} \\ \sum_{\omega \in \Omega} p_\omega = 1 \\ p_\omega \geq 0, \text{ for } \omega \in \Omega. \end{aligned}$$

Clearly, this is a linear program of a standard form. The left side of the interval $\text{fi}(Q; \mathcal{F}, \theta)$ can be solved similarly. The following theorem summarises the previous discussion:

Theorem 6.2 (Theorem 1 in [BSH04]). *Given a family $\mathcal{F}$ of itemsets along their frequencies θ , a frequency interval $\text{fi}(Q; S, \theta)$ for a query itemset is an interval whose boundaries can be solved using Linear Programming.*

Example 6.3. Let us reformulate the setup given in Example 6.1 as a linear program. In order to do that let p_{yz} represent the probability $p(a_1 = y, a_2 = z)$. The following linear program solves the right side of the frequency interval:

$$\begin{aligned} \max & p_{11} \\ p_{10} + p_{11} &= 0.6 \\ p_{01} + p_{11} &= 0.7 \\ \sum_{y,z \in \{0,1\}} p_{yz} &= 1 \\ p_{yz} &\geq 0, \text{ for } y, z \in \{0, 1\}. \end{aligned}$$

The min-version of the program results in the left side of $\text{fi}(Q; \mathcal{F}, \theta)$. $\square$

Let us now return to our original setup and consider what are the possible frequencies for query produced by *data sets*. The main difference here is that data sets must have finite number of elements. Thus, the empirical distribution formed from a finite data set has rational probabilities. This means that the possible frequencies for a query itemset Q should be rational. Given a distribution having only rational probabilities, we can easily form a data set having the distribution as empirical distribution. Theorem 3.7 guarantees that given a rational frequency $\eta \in \text{fi}(Q; S, \theta)$ there is a rational distribution producing η as a frequency for Q . Theorem 3.7 also guarantees that the boundaries of $\text{fi}(Q; S, \theta)$ are rational. We summarise this in the following theorem:

Theorem 6.4 (Lemma 1 in [BSH04]). *Given a family $\mathcal{F}$ of itemsets along their frequencies θ , possible frequencies for a query itemset Q produced by data sets satisfying the frequencies θ are $\text{fi}(Q; S, \theta) \cap \mathbb{Q}$. Also, boundaries of the interval $\text{fi}(Q; S, \theta)$ are rational and hence there are data sets producing these extrema frequencies.*

Example 6.5. We see that the boundaries in Example 6.1 are rational. For instance, a data set satisfying the frequencies θ and producing a frequency 0.6 for $Q = a_1 a_2$ is

$$D = \left\{ \begin{array}{l} (0, 0), (0, 0), (0, 0), (0, 1), (1, 1), \\ (1, 1), (1, 1), (1, 1), (1, 1), (1, 1) \end{array} \right\}.$$

$\square$

6.2 Complexity of Querying Itemsets

The major drawback in the query problem is that the number of variables in the linear program is $|\Omega| = 2^K$, where K is the number of attributes. In this section we will demonstrate that solving the query problem is **NP**-complete.

Recall that the downward closed family of itemsets is the one in which the subsets of a member itemset are also members. Consider the following problems:

<i>Consistent</i>	• CONSISTENT: Given a set of downward closed family of itemsets $\mathcal{F}$ and a set of rational frequencies θ, decide if there is a data set that produces θ for $\mathcal{F}$.
<i>MaxQuery</i>	• MAXQUERY: Given a set of downward closed family of itemsets $\mathcal{F}$, a set of rational consistent frequencies θ, and a query Q, find the maximal frequency of Q that a data set satisfying θ may achieve.
<i>EntrQuery</i>	• ENTRQUERY: Given a set of downward closed family of itemsets $\mathcal{F}$, a set of rational consistent frequencies θ, and a query Q, calculate the frequency $p^*(Q = 1)$, where p^* has the highest entropy among distributions satisfying θ.

Solving MAXQUERY is equal to solving the right side of $\text{fi}(Q; \mathcal{F}, \theta)$. ENTRQUERY is relevant because empirical tests indicate that this method leads to a good approximation of the frequency of Q [PMS03].

The following theorem states the complexity results of these problems.

Theorem 6.6 (Theorems 4, 6, and 7 in Publication II). *CONSISTENT and the decision version of MAXQUERY are **NP**-complete. The decision version of ENTRQUERY is **PP**-complete.*

6.3 Safe sets

As we have pointed out in Section 6.2, the evaluation time of the query time is exponential with respect to the number of attributes. Hence, we can speed up the algorithm if we can reduce the attributes: Assume that we are given a query itemset Q and a family of itemset $\mathcal{F}$ along with the frequencies θ . Define $\mathcal{F}_Q$ to contain only the itemsets from $\mathcal{F}$ that are subsets of Q . Let θ_Q be the corresponding frequencies. Instead of computing $\text{fi}(Q; \mathcal{F}, \theta)$, we *project* out the variables outside Q and compute $\text{fi}(Q; \mathcal{F}_Q, \theta_Q)$. In doing this, we reduce the number of attributes from K to $|Q|$. The downside is that the frequency interval may change.

Example 6.7. Assume that we have three attributes a , b , and c . Let $\mathcal{F}$ be $\{a, b, c, ab, ac\}$, and $\theta = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Let $Q = bc$ be the query itemset. Then

$\mathcal{F}_Q = \{b, c\}$. In this case $\text{fi}(Q; \mathcal{F}_Q, \theta_Q) = [0, \frac{1}{2}]$. However, it follows from θ that $a = b$ and $a = c$, hence we must have $b = c$. This implies that $\text{fi}(Q; \mathcal{F}, \theta) = \frac{1}{2}$. $\square$

Our goal is to study which attributes we can remove and which attributes we must keep. We begin by giving some definitions. Let $A = \{a_1, \dots, a_K\}$ be the set of all attributes and B a subset of A . Let q be a distribution defined on B . We say that p , a distribution defined on A , is an *extension* of q , if the marginalisation of p on B is equal to q , that is, $p(B = \omega) = q(B = \omega)$ for any binary vector ω . Given a family of itemsets $\mathcal{F}$ and frequencies θ , we say that B is θ -safe if a distribution q (defined on B) satisfying θ_B can be extended into p satisfying θ . If B is θ -safe for all θ , we say that B is *safe*.

It is easy to see that if B is a safe set and a query itemset Q is a subset of B , then we can remove the attributes outside B without changing the outcome.

The rest of this section is devoted to the analysis of the safe sets. Namely, we will apply Markov Random Field Theory in order to characterise safe sets. Let G be a graph of K nodes, each node corresponding to an attribute. For each itemset X in the family $\mathcal{F}$, we connect the nodes in G corresponding to X , that is X is a clique in G . We call G a *dependency graph* of $\mathcal{F}$. If, in addition, we connect all the nodes from B , we obtain a dependency graph of $\mathcal{F} \cup \{B\}$. The following theorem provides a neat way of characterising safe sets using dependency graphs.

Example 6.8. Assume that we have 6 attributes. Let

$$\mathcal{F} = \{a, b, c, d, e, f, ab, bc, ac, ad, bd, be, cf\}$$

be a family of itemsets. Let $B = abc$. The dependency graph of $\mathcal{F} \cup B$ is given in Figure 6.1(a). We are interested in finding out whether B is a safe set. To prove this we need to show that the distribution q defined on B can be extended into distribution p defined on all attributes.

Consider the junction tree of the dependency graph (given in Figure 6.1(b)). Let p_1 be a distribution defined on abd , p_2 a distribution defined on be , and p_3 a distribution defined on cf . The separator between abd and abc is ab . Note that ab is a member of $\mathcal{F}$. Hence, Theorem 2.10 implies that p_1 and q are equal at the separator ab . We have also p_2 and q being equal at the separator b and p_3 and q being equal at the separator c . Now we can apply Proposition 4.9 to combine q , p_1 , p_2 , and p_3 into a joint distribution. $\square$

The following theorem shows that the constuction done the previous example holds also in general case.

Theorem 6.9 (Theorems 1–2 in Publication III). *Let $\mathcal{F}$ be a downward closed family of itemsets. Let $B \notin \mathcal{F}$ be a subset of attributes. Let G be a dependency graph of $\mathcal{F} \cup \{B\}$. Then B is safe if and only if there is a junction tree T of G such that B is a node of T and all the separators of B are in $\mathcal{F}$.*

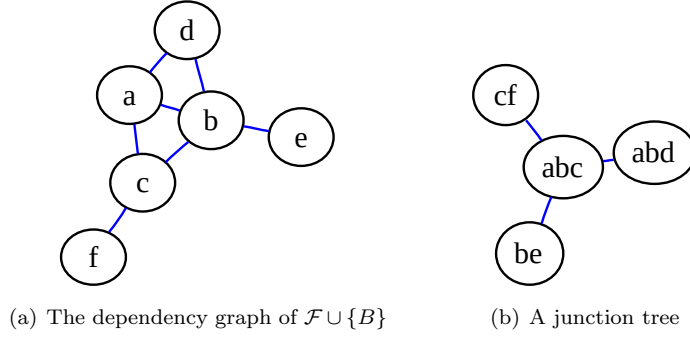


Figure 6.1: Graphs related to Example 6.8.

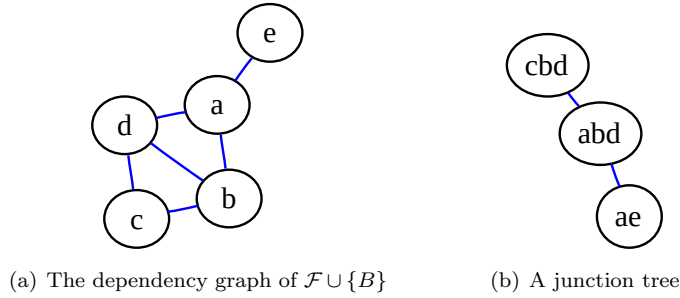


Figure 6.2: Graphs related to Example 6.10.

Example 6.10. Assume that we have 5 attributes. Let

$$\mathcal{F} = \{a, b, c, d, e, ab, bc, cd, ad, ae\}$$

be a family of itemsets. Let $B = abd$. The dependency graph of $\mathcal{F} \cup B$ is given in Figure 6.2(a) and its junction tree is given in Figure 6.2(b). The separators of B are a and bd . The itemset bd is not included in $\mathcal{F}$, hence B is not a safe set. However, if we augment $\mathcal{F}$ with bd , then B becomes a safe set. $\square$

Given a family of itemsets $\mathcal{F}$ and a query itemset Q , our goal is to find a safe set B that contains Q . The algorithm for finding such a set is described in Algorithm 1 in Publication III. The algorithm starts by setting $B = Q$ and augments B with attributes until B is safe. The addition order of the attributes is selected such that when B becomes safe, it is guaranteed that B will be also

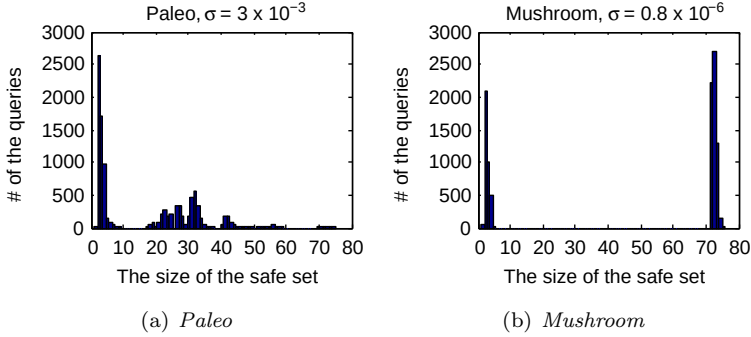


Figure 6.3: Distributions of the sizes of safe sets for random queries containing 2–4 attributes. The left histogram is obtained from the *Paleo* data set and the right histogram is obtained from the *Mushroom* data set.

the minimal safe set. The exact details of the algorithm are outside the scope of this introduction.

Theorem 6.11 (Theorem 6 in Publication III). *Let $\mathcal{F}$ be a downward closed family of itemsets and let Q be a query itemset. The minimal safe set B containing Q is unique. There exists a polynomial algorithm for finding B .*

Example 6.12. In this example we consider the sizes of safe sets for random queries. We used two data sets: *Paleo*¹, a data set containing information of species fossils found in specific paleontological sites in Europe [For05], and *Mushroom*, a data set available from the FIMI repository².

From these data sets we extracted a family of itemsets using a modified version of APRIORI (see Publication III for more details). Using these families as surrogates for the data sets we calculated the minimal safe sets for 10000 random queries having 2–4 attributes. The results given in Figure 6.3 show that even though the queries were relatively simple they may produce large safe sets, that is, we need a large amount of additional attributes to guarantee that the prediction boundaries are correct.

□

¹NOW public release 030717 available from [For05], preprocessed as in [FGJM06].

²<http://fimi.cs.helsinki.fi>

6.4 Optimising Linear Program via Markov Random Fields

In the previous section we have used MRF Theory to remove the attributes from the query problem. In this section we demonstrate that we can use similar ideas to reduce the complexity of the query problem even further.

Before presenting the main theorem of this section we will demonstrate the technique with an example.

Example 6.13. Assume that we have K attributes $A = \{a_1, \dots, a_K\}$. The family $\mathcal{F}$ contains $2K - 1$ itemsets,

$$\mathcal{F} = \{a_i; i = 1, \dots, K\} \cup \{a_i a_{i+1}; i = 1, \dots, K - 1\}.$$

Let query be $Q = a_1 a_K$. We see that the minimal safe set is A itself. Hence, we have 2^K variables in the linear program. However, we can reformulate the program in the following way: Let G be a dependency graph of $\mathcal{F} \cup \{Q\}$. The graph G is a cycle (see Figure 6.4(a)). Triangulate the graph by connecting a_1 with the rest of the attributes (Figure 6.4(b)). A junction tree T has $K - 2$ nodes (cliques) $C_i = a_1 a_{i+1} a_{i+2}$, where $i = 1, \dots, K - 2$ (Figure 6.4(c)). Consecutive nodes C_i and C_{i+1} are connected. Let S_j be the separators of T . Note that $S_j = a_1 a_{j+2}$.

Let θ be the frequencies for $\mathcal{F}$. Let p_i be a distribution defined on a clique C_i . Let $\mathcal{F}_i = \mathcal{F}_{C_i}$ be the subset of $\mathcal{F}$ containing only itemsets that are subsets of C_i . Consider the following linear program:

$$\begin{aligned} \max & p_{K-2}(Q = 1) \\ & p_i(X = 1) = \theta_X, \quad X \in \mathcal{F}_i, \quad i = 1, \dots, K - 2, \\ & p_j = p_{j+1} \text{ at } S_j, \quad j = 1, \dots, K - 3. \end{aligned} \tag{6.1}$$

The first set of conditions says that p_i must satisfy the related frequencies. The second set of conditions forces p_i to be consistent with respect to each other. Let p be a distribution (defined on A) maximising the frequency of Q . Clearly p can be decomposed into p_i such that the conditions in Eq. 6.1 hold. Assume now that p_i solves Eq. 6.1. We can apply Theorem 4.9 and compose p_i into p such that p satisfies θ .

This implies that we can solve $\text{fi}(Q; \mathcal{F}, \theta)$ by solving the linear program in Eq. 6.1. The number of variables in the program is $(K - 2) \times 8$ which is drastically smaller than 2^K . $\square$

The following theorem summarises the previous discussion.

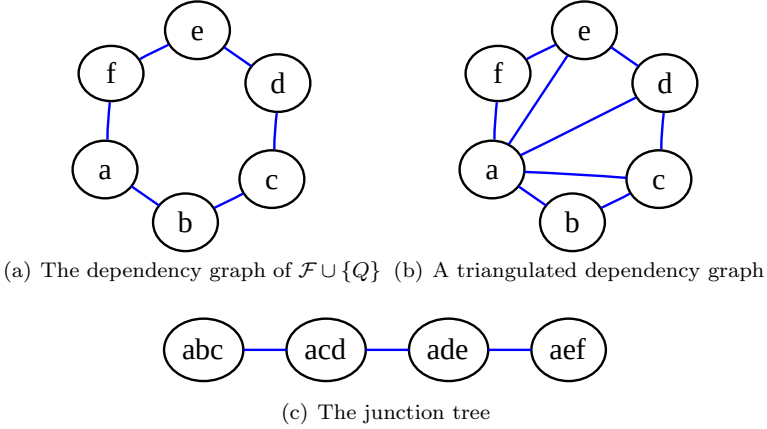


Figure 6.4: Graphs related to Example 6.13.

Theorem 6.14 (Theorem 9 in Publication III). *Let $\mathcal{F}$ be the family of itemsets along with the corresponding frequencies θ . Let Q be the query itemset. Let G be a dependency graph for $\mathcal{F} \cup Q$ and let T be its junction tree. Denote the nodes of T by C_i and let $\mathcal{F}_i = \mathcal{F}_{C_i}$. Assume that $Q \subseteq C_1$. Let p_i be a distribution defined on C_i . Then the frequency interval $\text{fi}(Q; \mathcal{F}, \theta)$ can be solved with a linear program (and its min-version)*

$$\begin{aligned} & \max p_1(Q = 1) \\ & p_i \text{ satisfies } \mathcal{F}_i, \quad \text{for each } C_i \\ & p_i = p_j \text{ at the separator,} \quad \text{for each neighbours (in } T) C_i, C_j. \end{aligned}$$

The number of variables in the program is $\sum_i 2^{|C_i|}$.

The method works well when the queries are relatively small but it fails if the query contains all the attributes in which case we asking the probability of having a transaction with no 0s. For this particular case we can use an alternative approach deccribed in [DF00]. In this approach we are able to solve the queries without using linear programming. The limitation for this approach is that the query must contain all the attributes.

6.5 Entropy Based Ranking of Itemsets

So far we have been interested in finding the frequency interval of an itemset given some known itemsets. A similar approach can be used in ranking itemsets:

We predict the frequency of an itemsets from some set of known itemsets and compare the actual value with the prediction. The more the actual value deviate from the prediction, the more the itemset is an interesting one. We will use Maximum Entropy as our estimation method and Kullback-Leibler divergence for comparison between the actual value and estimate.

Assume that we have a family of itemsets $\mathcal{F}$ and an itemset $Q \notin \mathcal{F}$. Assume for simplicity that there are no attributes in the data set outside Q . If there are, then these attributes are projected out. Let θ be the frequencies for $\mathcal{F}$. Let p^* be distribution having the highest entropy and satisfying the frequencies θ . Let q be the empirical distribution obtained from the data set D ,

$$q(\omega) = \frac{\text{transactions equal to } \omega \text{ in } D}{|D|}.$$

We define the rank of an itemset to be

$$r(Q; \mathcal{F}, D) = \text{KL}(q; p^*) = \sum_{\omega \in \Omega} q(\omega) \log \frac{q(\omega)}{p^*(\omega)}.$$

Example 6.15. Assume that we have $Q = ab$ and $\mathcal{F} = \{a, b\}$. Let data set D be

$$D = \left\{ \begin{array}{l} (0, 0), (0, 0), (0, 0), (0, 1), (1, 1), \\ (1, 1), (1, 1), (1, 1), (1, 1), (1, 1) \end{array} \right\}.$$

The frequency for a is 0.6 and the frequency for b is 0.7. We know that in this case the maximum entropy distribution p^* is equal to the independence model. Hence, we have

$$\begin{aligned} p^*(a = 0, b = 0) &= 0.4 \times 0.3, & p^*(a = 0, b = 1) &= 0.4 \times 0.7, \\ p^*(a = 1, b = 0) &= 0.6 \times 0.3, & p^*(a = 1, b = 1) &= 0.6 \times 0.7. \end{aligned}$$

On the other hand, the empirical distribution is equal to

$$\begin{aligned} q(a = 0, b = 0) &= 0.3, & q(a = 0, b = 1) &= 0.1, \\ q(a = 1, b = 0) &= 0.0, & q(a = 1, b = 1) &= 0.6. \end{aligned}$$

The rank of Q is equal to $r(Q; \mathcal{F}, D) = 0.3859$. □

Let us briefly discuss the evaluation of our rank measure. Distributions q and p^* have $2^{|Q|}$ entries. In addition, Theorem 6.6 points out that solving p^* is a **PP**-complete problem. Hence, we cannot solve this rank for large itemsets. Nevertheless, the rank is doable for itemsets of smaller size.

The following theorem, which follows from Theorem 5.6, explains the asymptotical behaviour of the measure.

Theorem 6.16 (Theorem 5 in Publication IV). *Let $\mathcal{F}$ be a family of itemsets and $Q \notin \mathcal{F}$. Let D be a data set with N points sampled from p^* . If Q is non-derivable, then the quantity $2Nr(Q; \mathcal{F}, D)$ approaches to a χ^2 distribution with $2^{|Q|} - 1 - |F|$ degrees of freedom as N approaches the infinity.*

Theorem suggests that the ranks should be normalised — instead of using the raw values we should compare the P -values.

Example 6.17. We continue Example 6.15. The number of degrees is

$$2^{|Q|} - 1 - |F| = 2^2 - 1 - 2 = 1.$$

The P -value in our case is

$$\mathbb{P}(\chi^2 \leq 2 \times 10 \times 0.3859) = 0.9945.$$

Such a high P -value tells us that the actual empirical frequency of the itemset ab is statistically significant different than the prediction based on the independence model. $\square$

Example 6.18. Consider a synthetic data set D with 100 independent columns and 1000 transactions. From this data set we select a certain set of queries (see Publication IV for more details) and calculate 3 different rank measures:

1. Measure $r(Q; \mathcal{I})$, where $\mathcal{I}$ is the set of itemsets of size 1. In this case, the Maximum Entropy distribution p^* is equal to the independence model.
2. Measure $r(Q; \mathcal{C})$, where $\mathcal{C}$ is the set of itemsets of size 1, 2. In this case, p^* is equal to the discrete Gaussian model.
3. Measure $r(Q; \mathcal{A})$, where $\mathcal{A}$ is the set of all proper sub-itemsets of Q .

The normalised ranks are given in Figure 6.5. We see that the ranks for $r(Q; \mathcal{I})$ are relatively small. This is a natural result since the 0-hypothesis of Theorem 6.16 holds for this particular data set. We also note that the ranks for richer models tends to be higher than for the independence model. In other words, the measure overfits the data and the prediction is misguided by the noise in the frequencies of the itemsets with the higher number of attributes. $\square$

Example 6.19. We repeat Example 6.18 using *Paleo*³, a dataset containing information of species fossils found in specific paleontological sites in Europe [For05]. The normalised ranks are given in Figure 6.6.

³NOW public release 030717 available from [For05], preprocessed as in [FGJM06].

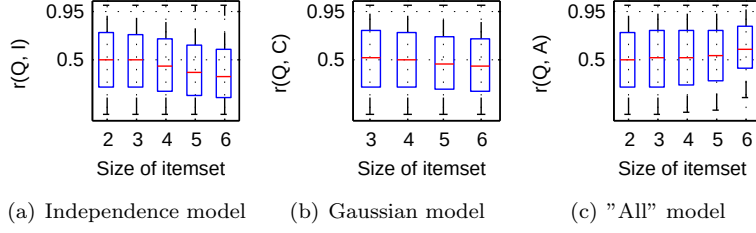


Figure 6.5: Ranks for queries from synthetic data set. Each box represents queries with particular number of attributes.

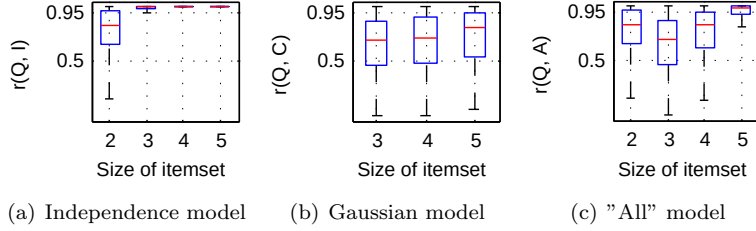


Figure 6.6: Ranks for queries from the *Paleo* data set. Each box represents queries with particular number of attributes.

Here we see that the ranks for $r(Q; \mathcal{I})$ are high. The attributes in the data set is known to be highly correlated so the independence model produces poor estimates. The Gaussian model produces more accurate prediction and, hence, smaller ranks. We also note that the "All" model overfits and produces higher ranks than the Gaussian model.

□

Chapter 7

Distances between Binary Data Sets

The notion of similarity plays a crucial part in data mining. Once a distance between two objects is established a large number of data mining algorithms can be applied and the highly complex objects can be studied as units.

In this chapter we discuss the distances between binary data sets. Instead of defining the distance directly we apply the theme of Chapter 6 and use itemsets as a surrogate for the actual data. We begin by defining the distance in Section 7.1 via geometrical notions. We point out in Section 7.2 that our distance is the only one among Mahalanobis distances that satisfies certain broad assumptions.

The contribution of the paper. This chapter is based on Publication I. In the paper we define a computable distance between two binary distances with solid statistical properties. We approach the problem by first calculating the frequencies of some given itemsets and compare the frequencies. Since we expect the frequencies to correlate we provide a proper normalisation. We provide 3 different definitions for the distance: Firstly, we use geometrical notions to define the distance. Secondly, we show that the distance is the unique Mahalanobis distance satisfying specific axioms. Thirdly, we show that the distance is the unique Mahalanobis distance that generalises the L_2 distance between two empirical distributions. We show that the distance can be solved in cubic time with the respect to the number of itemsets and in linear time if the itemsets form a downward closed family. Our experiments with real-world data show that our distance produces interesting results that agree with our expectations. An alternative approach to define a data set distance is to use some natural distance between single data points and apply some known set distance. Some data set distances defined in this way can be evaluated in cubic time with respect to the

number of transactions [EM97]. However, this is too slow for us since we may have a vast amount of data points. We can also approach the problem by considering distances for distributions (see [Bas89] for a nice review). From these distances the CM distance resembles the statistical tests involved with Minimum Discrimination Theorem [Kul68, Csi75]: From each data set a Maximum Entropy distribution is calculated and they are compared using Kullback- Leibler divergence. However, the major drawback is that solving the distributions is an NP-hard problem [Coo90].

7.1 Constrained Minimum Distance

In our approach we do not compute the distance directly between data sets. Rather, we compare the frequencies of some given family of itemsets. Such an approach provides us a flexible family of distances, since we can choose which itemsets we are interested in. A problem with the itemset frequencies is that they correlate: If the frequency of an attribute a is small, we expect that the frequency of an itemset ab is also small. Thus, we should seek a distance that decorrelates the frequencies.

Assume that we are given two binary data sets D_1 and D_2 , both having K attributes. Also assume that we are given a family of itemsets $\mathcal{F}$. We let $\mathcal{P}(\mathcal{F}, \theta)$ to be the set of distributions satisfying the frequencies θ , that is, $p \in \mathcal{P}(\mathcal{F}, \theta)$ if and only if $p(F_i = 1) = \theta_i$, for all $F_i \in \mathcal{F}$. The set $\mathcal{P}(\mathcal{F}, \theta)$ can be seen as a polytope in $\mathbb{R}^{2^K}$. We use the notation $\mathcal{P}(\mathcal{F}, D)$ if θ is calculated from a data set D .

One approach for defining the distance between D_1 and D_2 is to use the sets $\mathcal{P}(\mathcal{F}, D_1)$ and $\mathcal{P}(\mathcal{F}, D_2)$. However, as we have seen in Section 6.2, these sets are difficult to compute. Thus, another approach is needed. Recall that $S_{\mathcal{F}}$ is an indicator function for $\mathcal{F}$, that is, the i th component of $S_{\mathcal{F}}(\omega)$ is equal to 1 if ω satisfies $F_i \in \mathcal{F}$, and 0 otherwise. We define the set $\mathcal{C}(\mathcal{F}, \theta)$ to be

$$\mathcal{C}(\mathcal{F}, \theta) = \left\{ x \in \mathbb{R}^{2^K}; \sum_{\omega} x_{\omega} = 1, \sum_{\omega} S_{\mathcal{F}}(\omega) x_{\omega} = \theta \right\},$$

that is, $\mathcal{C}(\mathcal{F}, \theta)$ is similar to $\mathcal{P}(\mathcal{F}, \theta)$ except that we are allowed to have negative elements. See Figure 7.1 for illustration.

We see immediately that $\mathcal{C}(\mathcal{F}, \theta)$ is an affine space (a linear subspace shifted by vector) and that $\mathcal{P}(\mathcal{F}, \theta)$ is a subset of $\mathcal{C}(\mathcal{F}, \theta)$. In addition, the spaces $\mathcal{C}(\mathcal{F}, D_1)$ and $\mathcal{C}(\mathcal{F}, D_2)$ are parallel. This fact enables us to define the *constrained minimum (CM) distance* as

$$d_{CM}(D_1, D_2; \mathcal{F}) = \sqrt{2^K} \times \begin{array}{l} \text{the shortest distance between} \\ \mathcal{C}(\mathcal{F}, D_1) \text{ and } \mathcal{C}(\mathcal{F}, D_2). \end{array}$$

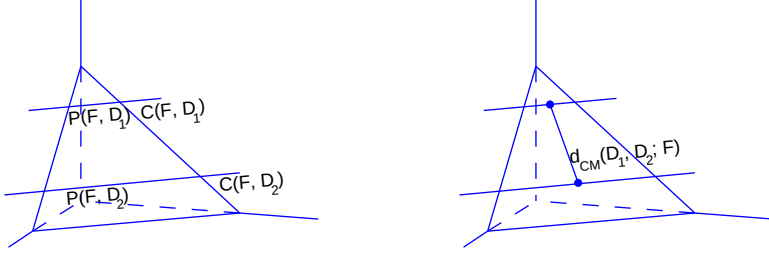


Figure 7.1: Illustration of the CM distance. The triangle represents the set of all possible distributions. The sets $\mathcal{C}(\mathcal{F}, D_i)$ are lines and the sets $\mathcal{P}(\mathcal{F}, D_i)$ are the segments containing the joint points from the set of all distributions and $\mathcal{C}(\mathcal{F}, D_i)$. The CM distance is proportional to the shortest distance between the spaces $\mathcal{C}(\mathcal{F}, D_1)$ and $\mathcal{C}(\mathcal{F}, D_2)$.

See Figure 7.1 for illustration.

It turns out that we can compute $d_{CM}(D_1, D_2; \mathcal{F})$ in polynomial time.

Theorem 7.1 (Theorem 1 in Publication I). *Assume two data sets D_1 and D_2 , and let $\mathcal{F}$ be a family of itemsets. Let θ and η be the frequencies calculated from D_1 and D_2 , respectively. Let q be the uniform distribution and define a covariance matrix C as*

$$C_{ij} = q(F_i = 1, F_j = 1) - q(F_i = 1)q(F_j = 1), \quad F_i, F_j \in \mathcal{F}.$$

We have

$$d_{CM}(D_1, D_2; \mathcal{F})^2 = (\theta - \eta)^T C^{-1} (\theta - \eta).$$

Theorem 7.1 suggests that the CM distance is an L_2 distance of the decorrelated itemset frequencies. We demonstrate Theorem 7.1 with the following example.

Example 7.2. Assume that we have D_1 and D_2 defined as

$$D_1 = \left\{ \begin{array}{l} (0, 0), (0, 0), (0, 0), (0, 1), (0, 1), \\ (0, 1), (1, 0), (1, 1), (1, 1), (1, 1) \end{array} \right\}$$

and

$$D_2 = \left\{ \begin{array}{l} (0, 0), (0, 0), (0, 1), (0, 1), (0, 1), \\ (1, 0), (1, 0), (1, 0), (1, 1), (1, 1) \end{array} \right\}.$$

Let $\mathcal{F} = \{a_1, a_2, a_1 a_2\}$. The frequencies for D_1 are equal to $\theta = (0.4, 0.6, 0.3)$ and the frequencies for D_2 are equal to $\eta = (0.5, 0.5, 0.2)$. The covariance matrix

C is equal to

$$C = \begin{bmatrix} \frac{1}{4} & 0 & \frac{1}{8} \\ 0 & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{3}{16} \end{bmatrix}.$$

Theorem 7.1 implies that the distance is

$$d_{CM}(D_1, D_2; \mathcal{F})^2 = (\theta - \eta)^T C^{-1} (\theta - \eta) = 0.24.$$

Let p_1 be the empirical distribution for D_1 ,

$$\begin{aligned} p_1(a_1 = 0, a_2 = 0) &= 0.3, & p_1(a_1 = 0, a_2 = 1) &= 0.3, \\ p_1(a_1 = 1, a_2 = 0) &= 0.1, & p_1(a_1 = 1, a_2 = 1) &= 0.3. \end{aligned}$$

and let p_2 be the empirical distribution for D_2 ,

$$\begin{aligned} p_2(a_1 = 0, a_2 = 0) &= 0.2, & p_2(a_1 = 0, a_2 = 1) &= 0.3, \\ p_2(a_1 = 1, a_2 = 0) &= 0.3, & p_2(a_1 = 1, a_2 = 1) &= 0.2. \end{aligned}$$

Note that $\mathcal{C}(\mathcal{F}, D_1) = \{p_1\}$ and $\mathcal{C}(\mathcal{F}, D_2) = \{p_2\}$. Hence, by the definition of the CM distance, we have

$$\begin{aligned} d_{CM}(D_1, D_2; \mathcal{F})^2 &= 2^K \|p_1 - p_2\|_2^2 \\ &= 4 [(0.3 - 0.2)^2 + (0.3 - 0.3)^2 + (0.1 - 0.3)^2 + (0.3 - 0.2)^2] \\ &= 0.24. \end{aligned}$$

□

We can also express CM distance in a neat form using parity functions. The following theorem is a corollary of Theorems 2.12, 2.13, and 7.1.

Theorem 7.3 (Section 3.1 in Publication I¹). *Assume two data sets D_1 and D_2 , and let $\mathcal{F}$ be a downward closed family of itemsets. Let $\mathcal{H}$ be the corresponding set of parity functions. Let α and β be the frequencies for $\mathcal{H}$ calculated from D_1 and D_2 , respectively. Then*

$$d_{CM}(D_1, D_2; \mathcal{F}) = 2 \|\alpha - \beta\|_2.$$

Example 7.4. We continue Example 7.2. Recall that the parity function results 1 if and only if an odd number of attributes are active. The parity frequencies for

¹In Eq. 4 in Publication I there should be 2 instead of $\sqrt{2}$.

D_1 are $\alpha = (0.4, 0.6, 0.4)$ and the parity frequencies for D_2 are $\beta = (0.5, 0.5, 0.6)$. Theorem 7.3 implies that

$$\begin{aligned} d_{CM}(D_1, D_2; \mathcal{F})^2 &= 4 \|\alpha - \beta\|_2^2 \\ &= 4 \left[(0.4 - 0.5)^2 + (0.6 - 0.5)^2 + (0.4 - 0.6)^2 \right] = 0.24. \end{aligned}$$

□

Example 7.5. We consider the following 3 data set families: *Bible*, a collection of 73 books from the Bible², *Addresses*, a collection of 55 inaugural addresses given by the presidents of the U.S.³, and *Abstract*, was composed of abstracts describing NSF awards from 1990–1999⁴ (see Publication I for more details).

We calculated the distance matrices for each data set collection using the following 3 itemset families: *ind*, the collection of itemsets containing only one attribute, *cov*, the family of itemsets containing 1–2 attributes, and *freq* a collection of 10K most frequent itemsets, where K is the dimension of the dataset.

From the results given in Figure 7.2 we see temporal behaviour in the data sets *Abstract* and *Addresses*. In *Bible* we note two clusters which are the Old and New Testaments.

□

7.2 Alternative Definition

In this section we will give an alternative definition for the CM distance. We have pointed out in Example 7.2 that if we know the frequencies of all itemsets, then the CM distance is basically an L_2 distance between the empirical distributions. It turns out that this property is almost sufficient to characterise the CM distance.

We say that a distance $d(x, y)$ is a *Mahanalobis distance* if it can be expressed as

Mahanalobis distance

$$d(x, y) = (x - y)^T C (x - y),$$

where C is a symmetric invertible matrix not depending on x or y . Theorem 7.1 shows that the CM distance is a Mahanalobis distance.

Assume that we have a Mahanalobis distance between data sets having the form

$$d(D_1, D_2; \mathcal{F}) = (\theta - \eta)^T C (\theta - \eta),$$

where θ and η are the frequencies of $\mathcal{F}$ calculated from D_1 and D_2 , respectively. The matrix C does not depend of the data sets but may depend of $\mathcal{F}$.

²The books were taken from <http://www.gutenberg.org/etext/8300> in 20. July 2005

³The addresses were taken from <http://www.bartleby.com/124/> in 17. August 2005

⁴The data set was taken from <http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.data.html> in 13. January, 2006

We impose the following assumptions on the distance.

(A1) Adding extra attributes, but not changing $\mathcal{F}$, does not change the distance.

(A2) Let $\mathcal{F} \subseteq \mathcal{G}$ be two families of itemsets. Then,

$$d(D_1, D_2; \mathcal{F}) \leq d(D_1, D_2; \mathcal{G}).$$

(A3) Let $\mathcal{A}$ be the family of all itemsets. Let p_i be the empirical distribution of D_i . Then,

$$d(D_1, D_2; \mathcal{A}) = \|p_1 - p_2\|_2.$$

Assumption A1 can be justified by noting that we haven't changed anything essential. We have added some extra attributes but they are ignored in $\mathcal{F}$. Assumption A2 states that additional information can only increase the difference between the data sets. Assumption A3 is motivated by Theorem 2.10 which states that we can deduce the empirical distribution if know the frequencies of all itemsets. Hence, we are able to use some distance between the distributions. In this case, we use the L_2 distance.

The following theorem states that the distance satisfying the aforementioned assumptions is essentially the CM distance.

Theorem 7.6 (Theorem 9 in Publication I). *Let $d(D_1, D_2; \mathcal{F})$ be a Mahanalobis distance satisfying Assumptions A1–A3. Let $\mathcal{F}$ be a downward closed family of itemsets. Then,*

$$d(D_1, D_2; \mathcal{F}) = \alpha d_{CM}(D_1, D_2; \mathcal{F}),$$

where α is a constant not depending on D_1 , D_2 , or $\mathcal{F}$.

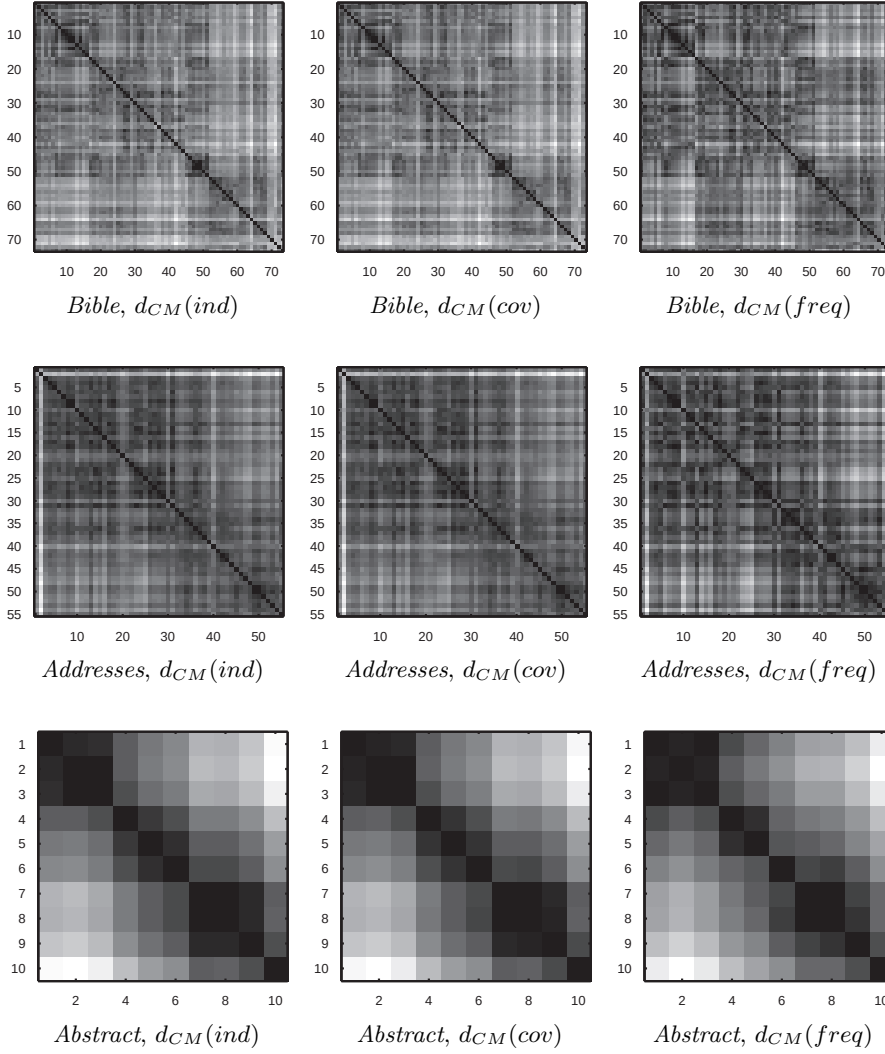


Figure 7.2: Distance matrices for *Bible*, *Addresses*, and *Abstract*. Dark values indicate small distances. In the first column the feature set *ind* contains the independent means, in the second feature set *cov* the pairwise correlation is added, and in the third column the feature set *freq* consists of 10K most frequent itemsets, where K is the number of attributes. Darker colours indicate smaller distances.

Chapter 8

Fractal Dimension of Binary Data

When asked about the dimensionality of a data set one's first answer would be that the dimension of a data set is equal to the number of columns. This, however, is too simplistic approach: Imagine a curve in a plane. The number of columns needed to represent the curve is 2. However, a more natural dimension for the curve is 1. Our goal is to define and measure this intrinsic dimension. The curve example points out that this dimension should take into account the structure of data.

We use fractal dimension, a popular and successful notion, to determine the intrinsic dimension of a binary data set. In Section 8.1 we will introduce the correlation dimension and provide some analysis. The problem with the fractal dimensions is that they are designed for continuous data and have some undesired properties. We remedy these problems in Section 8.2 by defining the normalised correlation dimension.

The contribution of the paper. This chapter is based on Publication V. In the paper we study the idea of using fractal dimension with binary data. We study the behaviour of the correlation dimension, one of the many fractal dimensions. However, the dimension has some undesired properties that are directly related to binary data: For instance, the dimension depends on the sparsity of data and the dimension is not a linear function of the number of attributes. We overcome these problems by defining the normalised correlation dimension. The idea is to compare the correlation dimension of the original data set against the correlation dimension against the correlation dimension of the data set having equal margins but independent attributes. We provide approximations for both dimensions and show empirically that these estimates yield good results. We also compare the dimension against PCA.

There has been a significant amount of work in defining the concept of dimensionality in datasets. Even though most of the methods can be adapted to the case of binary data, they are not specifically tailored for it. For instance, many methods assume real-valued numbers and they compute vectors/components that have negative or continuous values. Such methods include, PCA, SVD, and non-negative matrix factorisation (NMF) [Jol02, LS01]. Other methods such as multinomial PCA (mPCA) [BP03], and latent Dirichlet allocation (LDA) [BNJ03] assume specific probabilistic models of generating the data and the task is to discover latent components in the data rather than reasoning about the intrinsic dimensionality of the data. Methods for exact and approximate decompositions of binary matrices in Boolean semiring have also been proposed [GGM04, MMG⁺06, MPR95], but similarly to mPCA and LDA, they focus on finding components instead of the intrinsic dimensionality. In addition, many different notions of complexity of binary datasets have been proposed and used in various contexts, for instance VC-dimension [AB97], discrepancy [Cha00], Kolmogorov complexity [LV97] and entropy-based concepts [CT91, POP04]. Finally, methods such as multidimensional scaling (MDS) [Kru64] and Isomap [TdSL00] focus on embedding the data (not necessarily binary) in low-dimensional spaces with small distortion, mainly for visualisation purposes. A key difference with many above approaches is that fractal dimension does not provide a mapping to a lower-dimensional space, and hence traditional applications, such as feature reduction, are not (directly) possible. However, fractal dimension has been used in many applications related to database and data mining, such as, making nearest neighbour computations more efficient [PKF00], speeding up feature selection methods [TJTWf00], outlier detection [PKGF03], and performing clustering tasks based on the local dimensionality of the data points [GHPT05].

8.1 Correlation Dimension

There are infinite number of ways in defining the fractal dimension; see, e.g., [Bar88, Ott97] for a survey. The standard definitions involve usually partitioning the data into infinitesimal pieces and study how the data is distributed with respect to the partition. This cannot be done with finite data but the definitions can be modified to fit our purposes.

Given a data set D with K columns, let $0 \leq r_1 < r_2 \leq K$. Let Z_D be the distance between two randomly picked points from D . The *correlation dimension* $\text{cd}_R(D; r_1, r_2)$ for a binary data set D and radii r_1 and r_2 is the fraction

$$\text{cd}_R(D; r_1, r_2) = \frac{\log \mathbb{P}(Z_D < r_2) - \log \mathbb{P}(Z_D < r_1)}{\log r_2 - \log r_1},$$

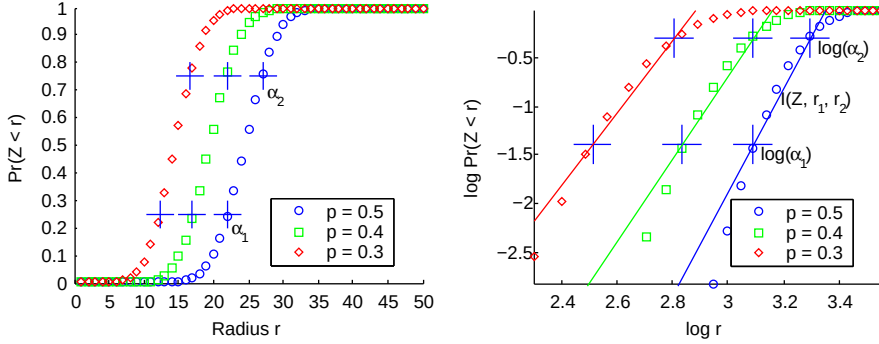


Figure 8.1: Examples of $\text{cd}_A(D)$ for different data sets. Plots represent three different data sets, each of them having 50 independent columns. The probability of a variable being 1 is p (indicated in the legend). The left figure is a regular plot of $\mathbb{P}(Z_D < r)$. The right figure is a log-log plot of $\mathbb{P}(Z_D < r)$. The crosses indicate the end points r_1 and r_2 that were determined by using $\alpha_1 = 1/4$ and $\alpha_2 = 3/4$. The slopes of the straight lines in the log-log plot are $\text{cd}_A(D; 1/4, 3/4)$. Note that the lines are gentler for smaller p .

that is, the correlation dimension is the slope of a line fitted into a log-log plot of the cumulative distribution function of Z_D ¹. For more details about the correlation dimension see e.g., [CY92].

The correlation dimension cd_R assumes that the radii r_1 and r_2 are given. The drawback with this approach is that the radii cannot be constant but should depend on the data set: For instance, $r_1 = 25$ and $r_2 = 75$ may be reasonable for a data set with 100 attributes but are absurd for a data set with only 20 columns. To remedy this problem we infer the radii from the distribution of Z_D : Assume that we are given α_1 and α_2 such that $0 \leq \alpha_1 < \alpha_2 \leq 1$. We define $\text{cd}_A(D; \alpha_1, \alpha_2)$ to be $\text{cd}_R(D; r_1, r_2)$, where the radii r_i are set such that $\alpha_i = \mathbb{P}(Z_D < r_i)$. For example,

$$\text{cd}_A(D; 1/4, 3/4) = \frac{\log 3/4 - \log 1/4}{\log r_2 - \log r_1},$$

where r_1 is the lower quartile point and r_2 is the upper quartile point. See Figure 8.1 for an illustration.

A direct analysis of the correlation dimension is difficult. To overcome these problems we define a much simpler quantity that can be used to approximate the

¹The definition given in Publication V is somewhat more complex. However, the above definition is adequate for our purposes.

correlation dimension.

Given a binary data set D , let Z_D be the distance between two randomly picked points from D . Given a real number $0 < \alpha < 1/2$ we define the -
approximative correlation dimension to be

$$\text{acd}(D) = \frac{E[Z_D]}{\text{Std}[Z_D]},$$

where $E[Z_D]$ is the average distance and $\text{Std}[Z_D]$ is the standard variation of Z_D . We will see in Theorem 8.4 that $\text{acd}(D)$ is asymptotically proportional to the correlation dimension $\text{cd}_A(D; \alpha, 1 - \alpha)$.

The following theorem describes $\text{acd}(D)$ when D has independent attributes.

Theorem 8.1 (Proposition 2 in Publication V). *Assume that the data set D has K independent variables, and that the probability of the variable i being 1 is p_i for each i , and let $q_i = 2p_i(1 - p_i)$. We have*

$$\text{acd}(D) = \frac{\sum_i q_i}{\sqrt{\sum_i q_i(1 - q_i)}},$$

In particular, if all probabilities p_i are equal to p , then for $q = 2p(1 - p)$ we have

$$\text{acd}(D) = \sqrt{\frac{Kq}{1 - q}}.$$

Corollary 8.2 (Corollary 3 in Publication V). *Assume the data set D has independent columns. The correlation dimension $\text{acd}(D)$ is maximised if the variables have frequency $\frac{1}{2}$.*

Given a data set D with K columns, we denote by $\text{ind}(D)$ a random binary data set having K independent variables such that the probability of i th variable being 1 is equal to the probability of i th column of a random transaction sampled from D being 1. Alternatively, $\text{ind}(D)$ can be considered as a data set obtained
permuted data set by permuting each column of D independently. We call $\text{ind}(D)$ a *permuted data set*. By permuting we keep the margins of the individual attributes but destroy any inter-column dependence.

Theorem 8.3 (Discussion after Conjecture 6 in Publication V). *Assume the marginal probability of all original variables are less than 0.5, and that all pairs of original variables are positively correlated. Then*

$$\text{acd}(D) \leq \text{acd}(\text{ind}(D)),$$

i.e., the approximative correlation dimension of the original data is not larger than the approximative correlation dimension of the data with each column permuted randomly.

The proof of the theorem is given in Appendix A.4.

The following theorem points out that $\text{acd}(D)$ is asymptotically proportional to the correlation dimension.

Theorem 8.4. *Assume that we have a sequence of independent binary variables X_i . Let D_K be the data set containing K first variables. Assume that $\text{Std}[Z_{D_K}]$ goes to infinity as K approaches infinity. We have*

$$\lim_{K \rightarrow \infty} \frac{\text{cd}_A(D_K; \alpha, 1 - \alpha)}{C(\alpha)\text{acd}(D_K)} = 1,$$

where $C(\alpha)$ is a constant depending only on α .

The proof of the theorem is given in Appendix A.5.

Theorem 8.4 justifies the following approximation.

Approximation 8.5. *Assume that the data set D has K independent variables. Assuming that K is large enough, we have*

$$\text{cd}_A(D; \alpha, 1 - \alpha) \approx C(\alpha)\text{acd}(D),$$

where $C(\alpha)$ is a constant depending only on α .

The assumption of independence in the statement of Theorem 8.4 is needed for estimating Z_D with a normal distribution. There are alternative versions of central limit theorems that allow non-independency, such as, central limit theorem for m -dependent variables [Ber73]. Hence, we can partly justify Approximation 8.5 for non-independent case.

Example 8.6. We study how accurate is Approximation 8.5 with synthetic data sets. We generated 100 data sets with independent columns and random margins (see Publication V for more details). The results given in Figure 8.2 show that $\text{acd}(D)$ yields a good approximation of the correlation dimension. □

8.2 Normalised Correlation Dimension

The scale of the correlation dimension is not very intuitive: the dimension of a dataset with K independent variables is not K , although this would be the most natural value. In fact, Theorem 8.1 implies that the correlation dimension is proportional to $\sqrt{K}$ for large K . The correlation dimension gives much smaller values and hence we need some kind of normalisation. Informally, we define the normalised correlation dimension of a dataset D to be the number of variables

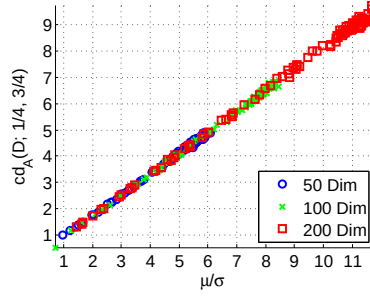


Figure 8.2: Correlation dimension $cd_A(D; 1/4)$ as a function of $acd(D)$ for data with independent columns (see Proposition 8.5). The y -axis is $cd_A(D; 1/4)$ and the x -axis is $acd(D) = \mu/\sigma$, where $\mu = E[Z_D]$ and $\sigma^2 = \text{Var}[Z_D]$. The slope of the line is about $C(1/4) = 0.815$.

that a dataset with independent variables must have in order to have the same correlation dimension as D has.

More formally, let $\text{ind}(H, p)$ be a dataset with H independent variables, each of which is equal to 1 with probability p . From Proposition 8.1 we have an approximation for $cd_A(\text{ind}(H, p); \alpha, 1 - \alpha)$: setting $q = 2p(1 - p)$ we have

$$cd_A(\text{ind}(H, p); \alpha, 1 - \alpha) \approx C(\alpha) \sqrt{\frac{Hq}{1 - q}}. \quad (8.1)$$

If the dataset would have the same marginal frequency, say s , for each variable, the normalised correlation dimension of a dataset D could be defined to be the number H such that

$$cd_A(D; \alpha, 1 - \alpha) = cd_A(\text{ind}(H, s); \alpha, 1 - \alpha).$$

The problem with this way of normalising the dimension is that it takes as the point of comparison a dataset where all the variables have the same marginal frequency. This is very far from being true in real data. We overcome this problem by first finding a value s such that

$$cd_A(\text{ind}(K, s); \alpha, 1 - \alpha) = cd_A(\text{ind}(D); \alpha, 1 - \alpha),$$

that is, a summary of the marginal frequencies of the columns of D : s is the frequency that variables of an independent dataset should have in order that it has the same correlation dimension as D has when the columns of D have been randomised. We define the *normalised correlation dimension*, denoted by

*normalised
correlation
dimension*

$\text{ncd}_A(D; \alpha, 1 - \alpha)$, to be an integer H such that

$$\text{cd}_A(\text{ind}(H, s); \alpha, 1 - \alpha) = \text{cd}_A(D; \alpha, 1 - \alpha).$$

The process is illustrated in Figure 8.3.

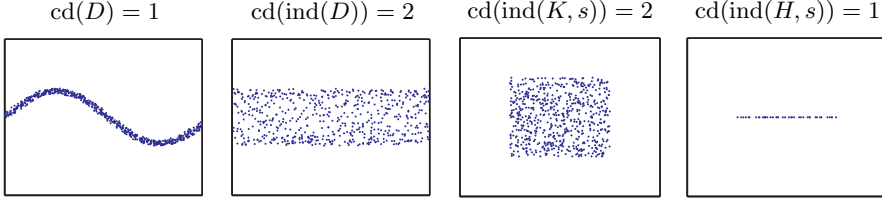


Figure 8.3: An illustration of computing normalised correlation dimension. The original data D is permuted, thus obtaining $\text{ind}(D)$. The margins of $\text{ind}(D)$ are forced to be equal such that the resulting dataset $\text{ind}(K, s)$ has the same correlation dimension. The dataset $\text{ind}(H, s)$ is computed such that $\text{cd}(\text{ind}(H, s)) = \text{cd}(D)$. H is the normalised correlation dimension.

Example 8.7. We examine the normalised correlation dimension using the toy data sets. We generated 100 data sets with independent columns and random margins and calculated $\text{ncd}_A(D; 1/4)$ for each data set. Figure 8.4(a) shows that the $\text{ncd}_A(D)$ is concentrated around the number of attributes, as expected, since the attributes are independent. On the other hand, Figure 8.4(b) shows that the sparsity of the data set does not change the normalised correlation dimension. $\square$

The estimate in Eq. 8.1 implies the following approximation.

Approximation 8.8. Given a data set D with K columns, the normalised dimension $\text{ncd}_A(D; \alpha, 1 - \alpha)$ can be approximated by

$$\text{ncd}_A(D; \alpha, 1 - \alpha) \approx \left(\frac{\text{cd}_A(D; \alpha, 1 - \alpha)}{\text{cd}_A(\text{ind}(D); \alpha, 1 - \alpha)} \right)^2 K.$$

We can estimate even further by approximating the correlation dimensions $\text{cd}_A(D)$ and $\text{cd}_A(\text{ind}(D))$. This gives us

Approximation 8.9. Given a data set D with K columns, the normalised dimension $\text{ncd}_A(D; \alpha, 1 - \alpha)$ can be approximated by

$$\text{ncd}_A(D; \alpha, 1 - \alpha) \approx \frac{\text{Var}[Z_{\text{ind}(D)}]}{\text{Var}[Z_D]} K = \frac{\sum_i C(Z_D)_{ii}}{\sum_{i,j} C(Z_D)_{ij}} K,$$

where $C(Z)$ is the covariance matrix $C(Z)_{ij} = E[Z_i Z_j] - E[Z_i] E[Z_j]$.

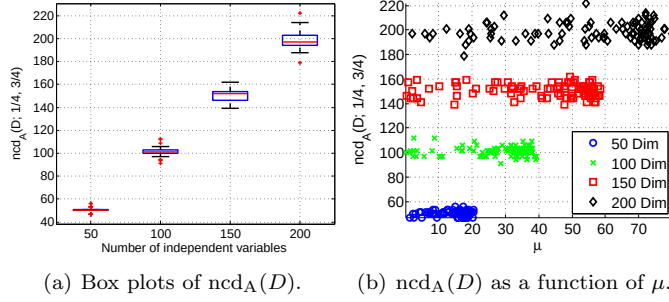


Figure 8.4: Normalised correlation dimension for data having K independent dimensions for $K \in \{50, 100, 150, 200\}$. In Figure 8.4(a) the normalised correlation dimension $ncd_A(D)$ is concentrated around the number of attributes. In Figure 8.4(b) $ncd_A(D)$ is plotted as a function of μ , the average distance between two random points. The x -axis is $\mu = E[Z_D]$ and the y -axis is $ncd_A(D; 1/4)$.

Note that the estimate in Approximation 8.9 does not depend on α .

Example 8.10. We tested Approximation 8.8 with synthetic data set having independent columns and *20 Newsgroups*², a collection of approximately 20 000 newsgroup documents across 20 different newsgroups [Lan95]. Figure 8.5 shows that Approximation 8.8 yields a good estimate for the selected data sets. $\square$

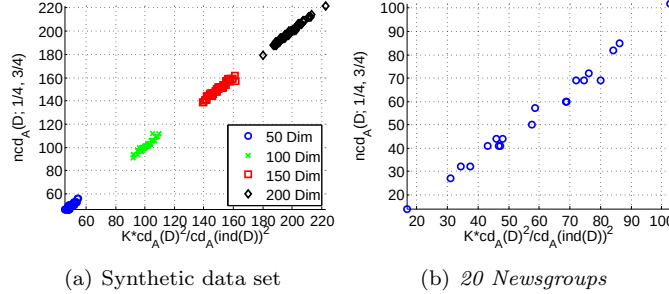


Figure 8.5: Normalised correlation dimension as a function of $K \cdot cd_A(D)^2 / cd_A(ind(D))^2$. Each point represents one data set. Figure 8.5(a) contains data sets with independent columns and Figure 8.5(b) contains data sets from the *20 Newsgroups* collection.

²<http://people.csail.mit.edu/jrennie/20Newsgroups/>

Appendix A

Proofs for the Theorems

A.1 Proof of Proposition 2.12

The existence of U follows directly from Theorem 2.8. To prove the invertibility of U let $B = \{b_1, \dots, b_L\}$ be a set of items. We need to show that there is a set of constraints u_C such that

$$S_B(\omega) = \sum_{C \subseteq B} u_C S_{\oplus C}(\omega).$$

We claim that $u_C = (-1)^{|C|-1} 2^{1-L}$. We first note that

$$\sum_{C \subseteq B} u_C S_{\oplus C}(1) = \sum_{C \subseteq B} 2^{1-L} [|C| \text{ is odd}] = 1 = S_B(1).$$

Now, let ω be a binary vector of length L having some elements as 0. Let $O = \{b_i; \omega_i\}$ be the subset of attributes. We can rewrite the sum as

$$\begin{aligned} \sum_{C \subseteq B} (-1)^{|C|-1} 2^{1-L} S_{\oplus C}(\omega) &= -2^{1-L} \sum_{X \subseteq O} \sum_{Y \subseteq B-O} (-1)^{|X|+|Y|} S_{\oplus X \oplus Y}(\omega) \\ &= -2^{1-L} \sum_{X \subseteq O} \sum_{Y \subseteq B-O} (-1)^{|X|+|Y|} S_{\oplus X}(\omega) \\ &= -2^{1-L} \sum_{X \subseteq O} (-1)^{|X|} S_{\oplus X}(\omega) \sum_{Y \subseteq B-O} (-1)^{|Y|} \\ &= -2^{1-L} \sum_{X \subseteq O} (-1)^{|X|} S_{\oplus X}(\omega) 0. \\ &= 0 = S_B(\omega) \end{aligned}$$

This completes the proof.

A.2 Proof of Proposition 4.9

Let

$$p(a_1, \dots, a_K) = \prod_{i=1}^N p_i(C_i) / \prod_{j=1}^{N-1} p_j(S_j).$$

Fix i and set C_i to be the root in T and define P_j to be the separator between the clique C_j and its parent (with respect to the root). Set $P_i = \emptyset$. The distribution p can now be written as

$$p(a_1, \dots, a_K) = \prod_{i=1}^N p_i(C_i; P_i). \quad (\text{A.1})$$

Let C_k be a leaf node. There is a node $v \in C_k$ such that $v \notin P_k$. Otherwise, p_k can be removed from the product. Note that v is not included in any other p_j since otherwise the running intersection property is violated. Hence, we can marginalise v out and still have the form of Eq. A.1. We repeat the marginalisation until we are left with p_i . This proves that the marginal distribution of p to C_i is equal to p_i .

A.3 Proof of Theorem 5.6

Before we state the regularity conditions, let us introduce some notation:

- By $\text{E}_\theta[\cdot]$ we denote the mean taking with respect to $p(\omega; \theta)$.
- The partial derivatives $\partial p(\omega; \theta) / \partial \theta_i$ and $\partial^2 p(\omega; \theta) / \partial \theta_i \partial \theta_j$ are shortened into $p_i(\omega; \theta)$ and $p_{ij}(\omega; \theta)$, respectively.
- The partial derivatives $\partial \log p(\omega; \theta) / \partial \theta_i$ and $\partial^2 \log p(\omega; \theta) / \partial \theta_i \partial \theta_j$ are shortened into $l_i(\omega; \theta)$ and $l_{ij}(\omega; \theta)$, respectively.
- A vector (depending on ω and θ) $l(\omega; \theta) = [l_1(\omega; \theta), \dots, l_K(\omega; \theta)]^T$ is called the score vector.
- A $K \times K$ matrix $I_\theta = \text{E}_\theta[l(\omega; \theta)l(\omega; \theta)^T]$ is called Fisher's information matrix.

The regularity conditions are:

1. θ_0 is an inner point of Θ , that is, there is an open K -dimensional ball B around θ_0 such that $B \subseteq \Theta$.
2. The family $p(\omega; \theta)$ is homogeneous in B , that is, if $\alpha \in B$ and $p(\omega; \alpha) = 0$, then $p(\omega; \beta) = 0$ for all $\beta \in B$.

3. Derivatives l_i and l_{ij} exist and are continuous (with respect to θ) for each $\omega \in \Omega$ and each $\theta \in B$.
4. θ_n is an efficient asymptotic normal estimate of θ_0 , that is, $\theta_n \rightsquigarrow \theta_0$ and $\sqrt{n}(\theta_n - \theta_0) \rightsquigarrow N(0, I_{\theta_0}^{-1})$, where $I_{\theta_0}^{-1}$ is the inverse of Fisher's information matrix. Note that we assume that I_{θ_0} is invertible.

Remark A.1. *The definition of weak convergence (or convergence in law) is given in [vdV98, Section 2.1]. We denote the weak convergence by $X_n \rightsquigarrow X$.*

Remark A.2. *We have assumed for simplicity that the sample space Ω is finite. The theorem also holds for general case under some additional regularity conditions.*

We need the following lemmata for proving the theorem:

Lemma A.3. $E_\theta[l_i(\omega; \theta)] = 0$.

Proof.

$$E_\theta[l_i(\omega; \theta)] = \sum_{\omega \in \Omega} p(\omega; \theta) \frac{p_i(\omega; \theta)}{p(\omega; \theta)} = \sum_{\omega \in \Omega} p_i(\omega; \theta) = \frac{\partial}{\partial \theta_i} \sum_{\omega \in \Omega} p(\omega; \theta) = \frac{\partial}{\partial \theta_i} 1 = 0.$$

□

Lemma A.4. $E_\theta[l_{ij}(\omega; \theta)] = -E_\theta[l_i(\omega; \theta)l_j(\omega; \theta)]$.

Proof.

$$\begin{aligned} E_\theta[l_{ij}(\omega; \theta)] &= \sum_{\omega \in \Omega} p(\omega; \theta) \frac{\partial}{\partial \theta_j} \frac{p_i(\omega; \theta)}{p(\omega; \theta)} \\ &= \sum_{\omega \in \Omega} p(\omega; \theta) \left[\frac{p_{ij}(\omega; \theta)}{p(\omega; \theta)} - \frac{p_i(\omega; \theta)p_j(\omega; \theta)}{p(\omega; \theta)^2} \right] \\ &= \sum_{\omega \in \Omega} p_{ij}(\omega; \theta) - \sum_{\omega \in \Omega} p(\omega; \theta) \frac{p_i(\omega; \theta)p_j(\omega; \theta)}{p(\omega; \theta)^2} \\ &= \frac{\partial^2}{\partial \theta_i \partial \theta_j} \sum_{\omega \in \Omega} p(\omega; \theta) - \sum_{\omega \in \Omega} p(\omega; \theta) l_i(\omega; \theta) l_j(\omega; \theta) \\ &= 0 - E_\theta[l_i(\omega; \theta)l_j(\omega; \theta)]. \end{aligned}$$

□

Lemma A.5 (Lemma 17.1 in [vdV98]). *Let Z be a random vector of length K distributed as $N(0, C)$, where C is invertible. Then $Z^T C^{-1} Z$ is distributed as χ^2 with K degrees of freedom.*

Proof of Theorem 5.6. Let $\alpha, \beta \in B$ be two vectors. Since l_i and l_{ij} are continuous, we can use Multidimensional Taylor's Theorem to obtain

$$\log p(\omega; \alpha) - \log p(\omega; \beta) = (\alpha - \beta)^T l(\omega; \beta) + \frac{1}{2} (\alpha - \beta)^T H(\omega; \gamma) (\alpha - \beta)^T,$$

where $\gamma \in B$ is a vector lying on a segment between α and β , and $H(\omega; \gamma)$ is a Hessian matrix $H_{ij}(\omega; \gamma) = l_{ij}(\omega; \gamma)$. By taking the mean and applying Lemma A.3 we obtain

$$-\text{KL}(\beta; \alpha) = \mathbb{E}_\beta[\log p(\omega; \alpha) - \log p(\omega; \beta)] = \frac{1}{2} (\alpha - \beta)^T \mathbb{E}_\beta[H(\omega; \gamma)] (\alpha - \beta)^T.$$

Assume for time being that $\theta_n \in B$. Let $\alpha = \theta_0$, $\beta = \theta_n$ and denote the resulting γ by η_n . If θ_n is outside B , set $\eta_n = 0$. Since $\theta_n \rightsquigarrow \theta_0$, we know from Theorem 2.7 in [vdV98] that $\eta_n \rightsquigarrow \theta_0$.

Define $g : \mathbb{R}^K \times \mathbb{R}^K \times \mathbb{R}^K \times \mathbb{R} \rightarrow \mathbb{R}$ to be

$$g(a, b, c, d) = \begin{cases} -a^T \mathbb{E}_b[H(\omega; c)] a & , \text{ if } b \in B \\ -\frac{2}{d} \text{KL}(b; \theta_0) & , \text{ if } b \notin B \end{cases}.$$

Note that

$$2n\text{KL}(\theta_n; \theta_0) = g(\sqrt{n}(\theta_n - \theta_0), \theta_n, \eta_n, 1/n).$$

Let $\sqrt{n}(\theta_n - \theta_0) \rightsquigarrow Z$, a random variable distributed as $N(0, I_{\theta_0}^{-1})$. Since g is continuous at $(Z, \theta_0, \theta_0, 0)$ we can apply Continuous Map Theorem [vdV98, Theorem 2.3] to obtain

$$2n\text{KL}(\theta_n; \theta_0) \rightsquigarrow g(Z, \theta_0, \theta_0, 0) = -Z^T \mathbb{E}_{\theta_0}[H(\omega; \theta_0)] Z.$$

An application of Lemma A.4 leads us to

$$2n\text{KL}(\theta_n; \theta_0) \rightsquigarrow Z^T I_{\theta_0} Z.$$

Since $I_{\theta_0}^{-1}$ is the covariance matrix of Z , we can apply Lemma A.5 to obtain the desired result. $\square$

A.4 Proof of Theorem 8.3

We begin by noting that $\mathbb{E}[Z_D] = \mathbb{E}[Z_{\text{ind}(D)}]$. Let $C(D)$ be the covariance matrix of the distance vector between two random points, that is,

$$C(D)_{ij} = \mathbb{E}[Z_i Z_j] - \mathbb{E}[Z_i] \mathbb{E}[Z_j],$$

where Z_i is the indicator variable having value 1 if two randomly chosen elements from D disagree at i th dimension. Note that $\text{Var}[Z_D] = \sum_{ij} C(D)$. Let $U =$

$C(D)$ and $V = C(\text{ind}(D))$. Note that V is a diagonal matrix having the diagonal equal to the diagonal of U . Hence, to prove the theorem we need to show that U contain only positive entries.

Fix i and j and abbreviate

$$x = \mathbb{P}(a_i = 1), \quad y = \mathbb{P}(a_j = 1), \quad z = \mathbb{P}(a_i = 1, a_j = 1).$$

The entry U_{ij} can be written as

$$\begin{aligned} U_{ij} &= 2z(1 - x - y + z) + 2(x - z)(y - z) - 4xy(1 - x)(1 - y) \\ &= 4z^2 + (2 - 4x - 4y)z + 2xy - 4xy(1 - x)(1 - y). \end{aligned}$$

The value of z minimising U_{ij} is

$$z = \frac{4x + 4y - 2}{8} = \frac{1}{2}(x + y) - \frac{1}{4}.$$

Since $x, y \leq \frac{1}{2}$, we have

$$z - xy = \frac{1}{2}(x + y) - \frac{1}{4} - xy = -\left(\frac{1}{2} - x\right)\left(\frac{1}{2} - y\right) \leq 0.$$

But we have assumed that $z \geq xy$, hence U_{ij} obtains its minimum value when $z = xy$, that is, a_i and a_j are independent. In this case $U_{ij} = 0$ and we have proved the statement.

A.5 Proof of Theorem 8.4

We need the following lemma.

Lemma A.6. *Assume sequences x_n , a_n , and b_n such that $x_n \rightarrow \infty$, $a_n \rightarrow a$ and $b_n \rightarrow b$. Then*

$$\left(\frac{x_n + a_n}{x_n + b_n}\right)^{x_n} \xrightarrow{n \rightarrow \infty} \exp(a - b).$$

Proof of Theorem 8.4. Recall that

$$\text{cd}_A(D; \alpha, 1 - \alpha) = \frac{\log(1 - \alpha) - \log \alpha}{\log r_2 - \log r_1},$$

where r_1 and r_2 are such that $\alpha = \mathbb{P}(Z_D < r_1)$ and $1 - \alpha = \mathbb{P}(Z_D < r_2)$. The numerator is $\log((1 - \alpha)/\alpha)$. Assume that K is large enough. Let $r_1(K)$ and $r_2(K)$ be the corresponding radii for the dataset D_K .

We next study the denominator $\log r_2 - \log r_1$. We have to analyse the distribution of the random variable Z_D , the L_1 distance between two randomly chosen points from D . For simplicity, we denote Z_{D_K} by Z_K in the sequel. Let Z_i be the indicator variable having value 1 if two randomly chosen elements from X_i disagree; then $Z_K = \sum_{i=1}^K Z_i$. Let

$$\mu_K = \mathbb{E}[Z_K], \quad \mu_i = \mathbb{E}[Z_i], \quad \sigma_K^2 = \mathbb{E}[(Z_K - \mu_K)^2].$$

Our goal is to estimate Z_K with the normal distribution. In order to do this we define

$$Y_{K,i} = \frac{Z_i - \mu_i}{\sigma_K}.$$

The sufficient condition for Lindeberg-Feller central limit theorem [vdV98, Theorem 2.27] is that for a fixed $\epsilon > 0$ there is L such that $|Y_{K,i}| \leq \epsilon$, whenever $K > L$. But this is true since $\sigma_K \rightarrow \infty$ and $|Z_i - \mu_i| \leq 1$. The central limit theorem implies that

$$\frac{Z_K - \mu_K}{\sigma_K} = \sum_{i=1}^K Y_{K,i} \rightsquigarrow N(0, 1).$$

This implies that

$$\frac{r_1(K) - \mu_K}{\sigma_K} \rightarrow -c \text{ and } \frac{r_2(K) - \mu_K}{\sigma_K} \rightarrow c,$$

where c is the inverse of the cumulative distribution function of the normal distribution with parameters 0 and 1, that is, $c = \Phi^{-1}(\alpha) = \sqrt{2} \operatorname{erf}^{-1}(2\alpha - 1)$.

Define $e_2(K) = r_2(K) - (\mu_K + c\sigma_K)$ and $e_1(K) = r_1(K) - (\mu_K - c\sigma_K)$. Also, let $z_K = \mu_K/\sigma_K$. Consider the ratio

$$\begin{aligned} \left(\frac{r_2(K)}{r_1(K)} \right)^{z_K} &= \left(\frac{\mu_K + c\sigma_K + e_2(K)}{\mu_K - c\sigma_K + e_1(K)} \right)^{z_K} \\ &= \left(\frac{z_K + c + e_2(K)/\sigma_K}{z_K - c + e_1(K)/\sigma_K} \right)^{z_K}. \end{aligned}$$

Note that $e_1(K)/\sigma_K \rightarrow 0$ and $e_2(K)/\sigma_K \rightarrow 0$. A straightforward calculation shows that $\mu_K \geq \sigma_K^2$. This implies that

$$z_K = \frac{\mu_K}{\sigma_K} \geq \frac{\sigma_K^2}{\sigma_K} = \sigma_K \rightarrow \infty$$

We can now apply the lemma to obtain

$$\left(\frac{r_2(K)}{r_1(K)} \right)^{z_K} \rightarrow \exp(2c).$$

Hence, we must have

$$\frac{\mu_K}{\sigma_K} (\log r_2(K) - \log r_1(K)) = z_K \log \frac{r_2(K)}{r_1(K)} \rightarrow 2c.$$

By setting

$$C(\alpha) = \frac{\log((1-\alpha)/\alpha)}{2c} = \frac{\log((1-\alpha)/\alpha)}{2\sqrt{2} \operatorname{erf}^{-1}(2\alpha-1)}$$

we have the desired result. □

Bibliography

- [AB97] Martin Anthony and Norman Biggs. *Computational Learning Theory: An Introduction*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.
- [AIS93] Rakesh Agrawal, Tomazc Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.
- [AMS⁺96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press/The MIT Press, 1996.
- [AY98] Charu C. Aggarwal and Philip S. Yu. A new framework for item-set generation. In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 18–24. ACM Press, 1998.
- [Bar88] Michael Barnsley. *Fractals Everywhere*. Academic Press, 1988.
- [Bas89] Michèle Baseville. Distance measures for signal processing and pattern recognition. *Signal Processing*, 18(4):349–369, 1989.
- [Ber73] Kenneth N. Berk. A central limit theorem for m-dependent random variables with unbounded m. *The Annals of Probability*, 1(2):352–354, Apr. 1973.
- [BFS03] Pierre Baldi, Paolo Frasconi, and Padhraic Smyth. *Modeling the Internet and the Web*. John Wiley & Sons, 2003.

- [BMS97] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 265–276. ACM Press, May 1997.
- [BMS02] Ella Bingham, Heikki Mannila, and Jouni K. Seppänen. Topics in 0–1 data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 450–455, New York, NY, USA, 2002. ACM.
- [BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, May 1997.
- [BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [BP03] Wray Buntine and Sami Perttu. Is multinomial PCA multi-faceted clustering or dimensionality reduction? In C.M. Bishop and B.J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 300–307, 2003.
- [BRRT05] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Trans. Inter. Tech.*, 5(1):231–297, 2005.
- [BSH04] Artur Bykowski, Jouni K. Seppänen, and Jaakko Hollmén. Model-independent bounding of the supports of Boolean formulae in binary data. In Pier Luca Lanzi and Rosa Meo, editors, *Database Support for Data Mining Applications: Discovering Knowledge with Inductive Queries*, LNCS 2682, pages 234–249. Springer Verlag, 2004.
- [BSS93] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, second edition, 1993.
- [BSVW99] Tom Brijs, Gilbert Swinnen, Koen Vanhoof, and Geert Wets. Using association rules for product assortment decisions: A case study. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA*, pages 254–260. ACM, 1999.

-
- [Cal03] Toon Calders. *Axiomatization and Deduction Rules for the Frequency of Itemsets*. PhD thesis, University of Antwerp, Belgium, 2003.
- [Cal04] Toon Calders. Computational complexity of itemset frequency satisfiability. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database System*, 2004.
- [CDLS99] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and Davig J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, 1999.
- [Cha00] Bernhard Chazelle. *The Discrepancy Method*. Cambridge University Press, 2000.
- [Coo90] Gregory Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, Mar. 1990.
- [Csi75] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 3(1):146–158, Feb. 1975.
- [CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley, 1991.
- [CY92] Sangit Chatterjee and Mustafa R. Yilmaz. Chaos, fractals and statistics. *Statistical Science*, 7(1):49–68, Feb. 1992.
- [Dan51] George B. Dantzig. Programming of interdependent activities, II, mathematical model. In Koopmans [Koo51], pages 19–32.
- [Dan63] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
- [DF00] Adrian Dobra and Stephen E. Fienberg. Bounds for cell entries in contingency tables given marginal totals and decomposable graphs. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 97(22):11885–11892, Oct. 2000.
- [DHS00] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.

- [DP01] William DuMouchel and Daryl Pregibon. Empirical bayes screening for multi-item associations. In *Knowledge Discovery and Data Mining*, pages 67–76, 2001.
- [DR72] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.
- [DS40] W. Edwards Deming and Frederick F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11(4):427–444, Dec 1940.
- [EM97] Thomas Eiter and Heikki Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133, 1997.
- [FGJM06] Mikael Fortelius, Aristides Gionis, Jukka Jernvall, and Heikki Mannila. Spectral ordering and biochronology of european fossil mammals. paleobiology. *Paleobiology*, 32(2):206–214, 2006.
- [For05] Mikael Fortelius. Neogene of the old world database of fossil mammals (NOW). University of Helsinki, <http://www.helsinki.fi/science/now/>, 2005.
- [GCB07] Arianna Gallo, Nello Cristianini, and Tijl De Bie. Mini: Mining informative non-redundant itemsets. In *11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 438–445, 2007.
- [GGM04] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In Einoshin Suzuki and Setsuo Arikawa, editors, *Discovery Science*, volume 3245 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2004.
- [GHPT05] Aristides Gionis, Alexander Hinneburg, Spiros Papadimitriou, and Panayiotis Tsaparas. Dimension induced clustering. In Robert Grossman, Roberto Bayardo, and Kristin P. Bennett, editors, *KDD*, pages 51–60. ACM, 2005.
- [GKM03] Aristides Gionis, Teija Kujala, and Heikki Mannila. Fragments of order. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 129–136, New York, NY, USA, 2003. ACM.

-
- [GKP88] George Georgakopoulos, Dimitris Kavvadias, and Christos H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1–11, March 1988.
- [GWBV03] Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16, 2003*.
- [Hai65] Theodore Hailperin. Best possible inequalities for the probability of a logical function of events. *The American Mathematical Monthly*, 72(4):343–359, Apr. 1965.
- [HAK⁺02] Jiawei Han, Russ B. Altman, Vipin Kumar, Heikki Mannila, and Daryl Pregibon. Emerging scientific applications in data mining. *Communications of the ACM*, 45(8):54–58, 2002.
- [Hay98] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998.
- [HFEM07] Hannes Heikinheimo, Mikael Fortelius, Jussi Eronen, and Heikki Mannila. Biogeography of european land mammals shows environmentally distinct and spatially coherent clusters. *Journal of Biogeography*, 34(6):1053–1064, 2007.
- [HHM⁺07] Hannes Heikinheimo, Eino Hinkkanen, Heikki Mannila, Taneli Mielikäinen, and Jouni K. Seppänen. Finding low-entropy sets and trees from binary data. In *Knowledge Discovery and Data Mining*, 2007.
- [HMS01] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. The MIT Press, 2001.
- [HSM03] Jaakko Hollmén, Jouni K Seppänen, and Heikki Mannila. Mixture models and frequent sets: combining global and local methods for 0-1 data. In *Proceedings of the SIAM Conference on Data Mining (2003)*, 2003.
- [Jay57] Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, May 1957.
- [Jol02] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd edition, 2002.
- [Jor99] Michael I. Jordan, editor. *Learning in graphical models*. MIT Press, 1999.

- [JP95] Radim Jiroušek and Stanislav Přešil. On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics and Data Analysis*, 19:177–189, 1995.
- [JS02] Szymon Jaroszewicz and Dan A. Simovici. Pruning redundant association rules using maximum entropy principle. In *Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference, PAKDD'02*, pages 135–147, May 2002.
- [JS04] Szymon Jaroszewicz and Dan A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 178–186, New York, NY, USA, 2004. ACM.
- [JS05] Szymon Jaroszewicz and Tobias Scheffer. Fast discovery of unexpected patterns in data, relative to a bayesian network. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 118–127, New York, NY, USA, 2005. ACM.
- [KBF⁺00] Ron Kohavi, Carla Brodley, Brian Frasca, Llew Mason, and Zijian Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000.
- [Kha79] Leonid G. Khachian. A polynomial algorithm for linear programming. *Doklady Akad. Nauk USSR*, 244(5):1093–1096, 1979. Translated in *Soviet Math. Doklady*, 20, 191–194.
- [KL51] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, Mar. 1951.
- [Koo51] Tjalling C. Koopmans, editor. *Activity Analysis of Production and Allocation*. John Wiley & Sons, 1951.
- [KRS02] Ron Kohavi, Neal J. Rothleder, and Evangelos Simoudis. Emerging trends in business analytics. *Communications of the ACM*, 45(8):45–48, 2002.
- [Kru64] Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of t to a nonmetric hypothesis. *Psychometrika*, 29:1–26, 1964.
- [Kul68] Solomon Kullback. *Information Theory and Statistics*. Dover Publications, Inc., 1968.

-
- [Lan95] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.
- [LS90] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. pages 415–448, 1990.
- [LS01] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562, 2001.
- [Luk01] Thomas Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Transactions on Computational Logic (TOCL)*, 2(3):289–339, July 2001.
- [LV97] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer-Verlag, 3rd edition, 1997.
- [Meo00] Rosa Meo. Theory of dependence values. *ACM Trans. Database Syst.*, 25(3):380–406, 2000.
- [MMG⁺06] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Knowledge Discovery in Databases: PKDD 2006 – 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, Croatia, September 18–22, 2006, Proceedings*, Lecture Notes in Computer Science. Springer, 2006.
- [MPR95] Sylvia D. Monson, Norman J. Pullman, and Rolf Rees. A survey of clique and biclique coverings and factorizations of $(0, 1)$ -matrices. *Bulletin of the ICA*, 14:17–86, 1995.
- [MT96] Heikki Mannila and Hannu Toivonen. Multiple uses of frequent sets and condensed representations (extended abstract). In *Knowledge Discovery and Data Mining*, pages 189–194, 1996.
- [MTV97] Heikki Mannila, Hannu Toivonen, and Aino Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

- [Omi03] Edward R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, 2003.
- [Ott97] Edward Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 1997.
- [Pea88] Judea Pearl. *Probabilistic Inference in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [PKF00] Bernd-Uwe Pagel, Flip Korn, and Christos Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *ICDE*, pages 2589–2598. IEEE Computer Society, 2000.
- [PKG03] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. LOCI: fast outlier detection using the local correlation integral. In Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman, editors, *ICDE*, pages 315–326. IEEE Computer Society, 2003.
- [PMS03] Dmitry Pavlov, Heikki Mannila, and Padhraic Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1409–1421, 2003.
- [POP04] Paolo Palmerini, Salvatore Orlando, and Raffaele Perego. Statistical properties of transactional databases. In Hisham Haddad, Andrea Omicini, Roger L. Wainwright, and Lorie M. Liebrock, editors, *SAC*, pages 515–519. ACM, 2004.
- [PS91] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press, 1991.
- [PS98] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization Algorithms and Complexity*. Dover, 2nd edition, 1998.
- [Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
- [Sha48] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, Oct. 1948.

-
- [TdSL00] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- [TJTWF00] Caetano Traina Jr., Agma J. M. Traina, Leejay Wu, and Christos Faloutsos. Fast feature selection using fractal dimension. In Karin Becker, Adriano Augusto de Souza, Damires Yluska de Souza Fernandes, and Daniela Coelho Freire Batista, editors, *SBBD*, pages 158–171. CEFET-PB, 2000.
- [vdV98] Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

Index

- σ -frequent, [10](#)
- θ -safe, [41](#)
- absolutely continuous, [30](#)
- antimonotonic, [10](#)
- approximative correlation dimension, [60](#)
- attribute, [8](#)
- basic feasible solution, [18](#)
- basic solution, [18](#)
- binary data set, [7](#)
- chord, [23](#)
- chordless, [23](#)
- clique graph, [24](#)
- connected, [22](#)
- CONSISTENT, [40](#)
- constrained minimum (CM) distance, [50](#)
- correlation dimension, [58](#)
- cycle, [23](#)
- decomposable, [25](#)
- dependency graph, [41](#)
- dimension, [7](#)
- downward closed, [10](#)
- ELIMINATION Algorithm, [23](#)
- ELLIPSOID Algorithm, [19](#)
- empirical distribution, [8](#)
- entropy, [30](#)
- ENTRQUERY, [40](#)
- exponential form, [32](#)
- extension, [41](#)
- feasible set, [16](#)
- frequency, [9](#)
- frequency interval, [37](#)
- indicator function, [9](#)
- itemsets, [10](#)
- ITERATIVE SCALING Algorithm, [33](#)
- junction tree, [24](#)
- Kullback-Leibler divergence, [30](#)
- Mahanalobis distance, [53](#)
- Maximum Entropy, [32](#)
- MAXQUERY, [40](#)
- normalised correlation dimension, [62](#)
- parity formula, [12](#)
- permuted data set, [60](#)
- PRIMAL-DUAL PATH-FOLLOWING Algorithm, [19](#)
- project, [40](#)
- running intersection property, [24](#)
- safe, [41](#)
- sample space, [7](#)
- satisfies, [32](#)

satisfies the frequency, [9](#)

separator, [24](#)

SIMPLEX, [19](#)

standard form, [16](#)

transaction, [7](#)

triangulated, [23](#)

Set Covering with Our Eyes Closed*

Fabrizio Grandoni[†]

Anupam Gupta[‡]

Stefano Leonardi[§]

Pauli Miettinen[¶]

Piotr Sankowski[§]

Mohit Singh[‡]

Abstract

Given a universe U of n elements and a weighted collection $\mathcal{S}$ of m subsets of U , the *universal set cover* problem is to a-priori map each element $u \in U$ to a set $\mathbf{S}(u) \in \mathcal{S}$ containing u , so that $X \subseteq U$ is covered by $\mathbf{S}(X) = \cup_{u \in X} \mathbf{S}(u)$. The aim is finding a mapping such that the cost of $\mathbf{S}(X)$ is as close as possible to the optimal set-cover cost for X . (Such problems are also called *oblivious* or *a-priori* optimization problems.)

Universal algorithms are useful in distributed settings, where decisions are taken locally to minimize the communication overhead. Similarly, in critical applications one wants to pre-compute solutions for a family of scenarios, so as to react faster when the actual input shows up. Moreover, universal mappings can be translated into online algorithms. Unfortunately, for every universal mapping, if the set X is adversarially chosen, the cost of $\mathbf{S}(X)$ can be $\Omega(\sqrt{n})$ times larger than optimal (e.g., see Jia et al. [STOC'05]).

However, in many applications, the parameter of interest is not the worst-case performance, but instead the *performance on average*: what can we say when X is a set of randomly chosen elements from the universe? How does the expected cost of $\mathbf{S}(X)$ compare to the expected optimal cost? In this paper we present a polynomial-time $O(\log mn)$ -competitive universal algorithm. In fact, we give a slightly improved analysis and show that this is the best possible.

Our algorithm is based on interleaving two greedy algorithms: the standard greedy algorithm picking the set with the best ratio of cost to number of uncovered elements, and an even dumber greedy algorithm picking the cheapest subset that covers any uncovered element. The crux of our analysis is bounding the expected cost of the sets selected by the second algorithm, for which known techniques for worst-case analysis do not seem to apply; hence we develop novel counting arguments for our analyses.

We generalize the ideas for weighted set cover and show similar guarantees to (non-metric) facility location, where we have to balance the facility opening cost with the cost of connecting clients to the facilities. Additionally, we show applications of our set-cover result to universal multi-cut and disc-covering problems.

Finally, we show how all these universal mappings naturally give us *stochastic online algorithms* with the same competitive factors; this complements the $O(\log m \log n)$ *worst-case* online algorithm for set cover problem of Alon et al. [STOC'03].

*Part of this work was done when the non-Roman authors were visiting the “Sapienza” Università di Roma.

[†]Dipartimento di Informatica, Sistemi e Produzione, Università di Roma “Tor Vergata”, via del Politecnico 1, 00133, Roma, Italy. Partially supported by MIUR under project MAINSTREAM.

[‡]Carnegie Mellon University, Pittsburgh, PA 15213.

[§]Dipartimento di Informatica e Sistemistica, “Sapienza” Università di Roma, Via Ariosto 25, 00185 Rome, Italy. Partially supported by the EU within the 6th Framework Programme under contract no. 001907 “Dynamically Evolving, Large Scale Information Systems” (DELIS)

[¶]Helsinki Institute for Information Technology, University of Helsinki, P.O. box 68 (Gustaf Hållströmin katu 2b), 00014 Helsinki, Finland.

1 Introduction

In the classical *set cover* problem we are given a set X , taken from a *universe* U of n elements, and a collection $\mathcal{S} \subseteq 2^U$ of m subsets of U , with a cost function $c: \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$. (The pair $(U, \mathcal{S})$ is sometimes called a *set system*). The aim is to compute a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ which covers X , i.e., $X \subseteq \bigcup_{S \in \mathcal{S}'} S$, with minimum cost $c(\mathcal{S}') := \sum_{S \in \mathcal{S}'} c(S)$. Each feasible solution can also be interpreted as a mapping $\mathbf{S}: U \rightarrow \mathcal{S}$ which defines, for each $u \in X$, a subset $\mathbf{S}(u)$ which covers u (breaking ties in an arbitrary way). In particular, $\mathbf{S}(X) := \bigcup_{u \in X} \mathbf{S}(u)$ provides the desired sub-collection $\mathcal{S}'$, of cost $c(\mathbf{S}(X)) := \sum_{S \in \mathbf{S}(X)} c(S)$. In the *cardinality* (or *unweighted*) version of the problem, all the set costs are 1, and the goal is minimizing the number $|\mathbf{S}(X)|$ of subsets used to cover X .

In their seminal work, Jia et al. [JLN⁺05] define, among other problems, a *universal* variant of the set cover problem. Here the mapping $\mathbf{S}$ has to be provided a-priori, i.e., without knowing the actual value of $X \subseteq U$. The problem now is to find a mapping which minimizes the worst-case ratio $\max_{X \subseteq U} \{c(\mathbf{S}(X))/c(\text{opt}(X))\}$ between the cost of the set cover given by $\mathbf{S}$ (which is computed without knowing X), and the cost of the optimal “offline” solution $\text{opt}(X)$ (which is based on the knowledge of X). A universal algorithm is α -competitive if the ratio above is at most α .

Universal algorithms are useful for applications in distributed environments, where decisions have to be taken locally, with little communication overhead. Similarly, in critical or real-time applications we might not have enough time to run any approximation algorithm once the actual instance of the problem shows up. Hence we need to perform most of the computation beforehand, even if this might imply worse competitive factors and higher preprocessing time. Indeed, we might also think of applications where the solution computed a-priori is wired on a circuit. Eventually, universal problems have strong implications to online problems (where the instance is revealed gradually, and the solution is computed step-by-step). In particular, any universal algorithm provides an online algorithm with the same competitive ratio.

The standard competitive analysis for universal (and online) algorithms assumes that the input is chosen adversarially, and often this setting is too pessimistic: indeed, for universal set cover, Jia et al. [JLN⁺05] gave $\tilde{O}(\sqrt{n})$ bounds. However, in many situations it is often more reasonable to assume that the input is sampled according to some probability distribution. In other words, what if we are competing against nature and the lack of information about the future, and not against a malicious adversary out to get us? Can we give algorithms with a better performance in that case?

1.1 Our Results and Techniques

We formalize the questions above by defining a *stochastic* variant of the universal set cover problem. Here the input X is obtained by sampling k times a given probability distribution $\pi: U \rightarrow [0, 1]$. Let $\omega \in U^k$ be the random sequence of elements obtained (possibly with repetitions), and let us interpret ω as a set of elements when the ordering (and multiplicity) of elements in the sequence is not relevant. The aim is minimizing the ratio $\mathbf{E}_\omega[c(\mathbf{S}(\omega))]/\mathbf{E}_\omega[c(\text{opt}(\omega))]$ between the expected cost of the solution computed w.r.t. $\mathbf{S}$ and the expected optimal cost. We will sometimes omit ω when the meaning will be clear from the context.

An algorithm for the universal stochastic set cover problem is *length-aware* if it is given the length k of the sequence in input, and *length-oblivious* otherwise.

As a warm up for the reader, we present a lower bound on the quality of the mapping obtained by running on the set system $(U, \mathcal{S})$ the standard greedy algorithm for weighted set cover, which selects in each step the subset with the best ratio of cost to number of uncovered elements. This algorithm defines an order on the selected sets: let each element be mapped to the first set in the order covering it. Consider a set $S_{\text{all}} = U$ covering the whole universe, of cost $c(S_{\text{all}}) = \sqrt{n}$, and singleton sets $S_u = \{u\}$ for each $u \in U$, each of unit cost $c(S_u) = 1$. The greedy set cover algorithm maps all the elements into S_{all} . For a uniform distribution π and $k = 1$, the cost of

this mapping is $\sqrt{n}$, while the optimal mapping (assigning each $u \in U$ to the corresponding singleton set S_u) has always cost one. Note that, for $k \simeq n$, the situation changes drastically: now the greedy algorithm produces the optimal mapping with high probability. Indeed, essentially the same example shows that any length-oblivious universal algorithm for the (weighted) stochastic set cover problem must be $\Omega(\sqrt{n})$ -competitive (see [Section 3](#)).

Motivated by the example above, we developed an algorithm based on the interleaving of standard greedy with a second, *even more myopic*, greedy algorithm that selects the min-cost set which covers at least *one* uncovered element (disregarding the actual number of the covered elements). In each selection step we trust the *min-ratio* greedy algorithm if a subset with a sufficiently small ratio exists, and the *min-cost* one otherwise. The threshold ratio is derived from the length k of the sequence.

The main result of this paper can be stated as follows (see [Section 3](#)):

Theorem 1.1 *There exists a polynomial-time length-aware algorithm that returns a universal mapping $\mathbf{S}$ to the (weighted) universal stochastic set cover problem with $\mathbf{E}[c(\mathbf{S})] = O(\log mn)\mathbf{E}[c(\text{opt})]$.*

When m is polynomial in n , this is asymptotically the best possible due to the $o(\log n)$ -inapproximability of set cover (which extends to the universal stochastic case by choosing $k \gg n$). For values of $m \gg n$, the competitive factor can be improved to $O\left(\frac{\log m}{\log \log m - \log \log n}\right)$, and this bound is tight (see [Section 4](#)).

The crux of our analysis is bounding the cost of the min-cost sets selected by the algorithm when it cannot find good ratio sets. Here we use a novel counting argument to show that the number of sampled elements among the still-uncovered elements is sufficiently *small* compared to the number of sets used by the optimal solution to cover those elements. We then translate this into a convenient lower bound on the cost paid by the optimum solution to cover the mentioned elements.

In the unweighted case we can do better: here the standard greedy algorithm provides a *length-oblivious* universal algorithm with the same competitive ratio.

Theorem 1.2 *There exists a polynomial-time length-oblivious algorithm that returns a universal mapping $\mathbf{S}$ to the unweighted universal stochastic set cover problem with $\mathbf{E}[|\mathbf{S}|] = O(\log mn)\mathbf{E}[|\text{opt}|]$.*

Based on the proof of [Theorem 1.2](#), we also show that the dependence on n in the competitive factor can be removed if exponential time is allowed, or when the set system has a small VC-dimension. The latter result is especially suited for applications where $m \ll n$, one of which we highlight in [Appendix B](#).

Our results naturally extend to the stochastic version of the *online set cover* problem. Here the random sequence ω is presented to the algorithm element by element, and, each time a new element u is given, the algorithm is forced to define a set $\mathbf{S}(u) \ni u$. In other words, the mapping $\mathbf{S}$ is constructed in an online fashion. We remark that, once the value $\mathbf{S}(u)$ is chosen, it cannot be modified in the following steps. Moreover, the length k of the sequence is not given to the algorithm. Similarly to the universal stochastic case, the aim is to minimize $\mathbf{E}_\omega[c(\mathbf{S}(\omega))]/\mathbf{E}_\omega[c(\text{opt}(\omega))]$.

A length-oblivious universal algorithm would immediately imply an online algorithm with the same competitive factor. We achieve the same task by combining a family of universal mappings, computed via our (length-aware) universal algorithm for carefully-chosen sequence lengths (see [Section 5](#)):

Theorem 1.3 *There exists a polynomial-time $O(\log mn)$ -competitive algorithm for the online (weighted) stochastic set cover problem.*

Our techniques are fairly flexible, and can be applied to other covering-like problems. In [Section 6](#) we describe universal algorithms for the stochastic versions of (non-metric) *facility location*, *multi-cut*, and *disc covering* in the plane.

In this paper, $\log x$ denotes the logarithm at base 2 of x . In the remaining of this paper, we assume that π is a uniform distribution: This is w.l.o.g. by the standard reduction described in [Appendix A](#).

1.2 Related Work

Universal, Oblivious and A-Priori Algorithms. These are algorithms where a single solution is constructed which is evaluated given multiple inputs—and either the worst-case or the average-case performance is considered. E.g., the universal TSP problem, where one computes a permutation that is used for all possible inputs, has been studied both in the worst-case scenario for the Euclidean plane [PB89, BG89] and general metrics [JLN⁺05, GHR06, HKL06], as well as in the average-case [Jai88, BJO90, SS08, GGLS08, ST08]. (For the related problem of universal Steiner tree, see [KM00, JLN⁺05, GHR06, GGLS08].) For *universal set cover* and *facility location*, the previous results are in the worst-case: Jia et al. [JLN⁺05] introduced the problems, show that the adversary is very powerful in such models, and give nearly-matching $\Omega(\sqrt{n})$ and $O(\sqrt{n} \log n)$ bounds.

Another well-studied problem is *oblivious routing* [Räc02, HHR03, BKR03] (see, e.g., [VB81, Vöc01] for special cases): in the worst-case, a tight logarithmic competitive result as well as a polynomial-time algorithm to compute the best routing is known for undirected graphs [ACF⁺03, Räc08]. For *oblivious routing on directed graphs* the situation is surprisingly similar to our problem: in the worst case, the lower bound of $\Omega(\sqrt{n})$ [ACF⁺03] nearly matches upper bounds [HKLR05a]. However, for the average case on directed graphs, [HKLR05b] give an $O(\log^2 n)$ -competitive oblivious routing algorithm when demands are chosen randomly from a known demand-distribution. In fact, [HKLR05b] also use “demand-dependent” routings and show that these are necessary; this is similar to our use of “length-aware” universal maps in [Section 3](#). As far as we can see, the connection of our results to [HKLR05b] is mostly in spirit, and the techniques required are somewhat different.

Online Algorithms. Online algorithms have a long history (see, e.g., [BEY98, FW98]), and there have been many attempts to relax the strict worst-case notion of competitive analysis: see, e.g., [DLO05, AL99, GGLS08] and the references therein. Online algorithms with stochastic inputs (either i.i.d. draws from some distribution, or inputs arriving in random order) have been studied, e.g., in the context of optimization problems [Mey01, MMP01, GGLS08], secretary problems [Fre83], mechanism design [HKP04, Kle05, BIK07], and matching problems in Ad-auctions [MSVV07, BJN07, GM08].

Alon et al. [AAA⁺03] gave the first online algorithm for set cover with a competitive ratio of $O(\log m \log n)$; they used an elegant primal-dual-style approach that has subsequently found many applications (e.g., [AAA⁺04, BN05, AAG05]). This ratio is the best possible under complexity-theoretic assumptions [FK]; even unconditionally, no deterministic online algorithm can do much better than this [AAA⁺03]. Online versions of *metric facility location* are studied in both the worst case [Mey01, Fot03], the average case [GGLS08], as well as in the stronger *random permutation model* [Mey01], where the adversary chooses a set of clients unknown to the algorithm, and the clients are presented to us in a random order. It is easy to show that for our problems, the random permutation model (and hence any model where elements are drawn from an *unknown* distribution) are as hard as the worst case.

Offline problems: Set Cover and (non-metric) Facility Location. The set cover problem is one of the poster children for approximation algorithms, for which a $\Theta(\ln n)$ -approximation has been long known [Joh74, Chv79, Lov75], and this is the best possible [LY94, Fei98, RS97, AMS06]. For the special case of set systems with small VC-dimension, a better algorithm is known [BG95]. Other objective functions have also been used, e.g., the min-latency [FLT04] and min-entropy [HK05, CFJ06] set cover problems. The $O(\log n)$ approximation for non-metric facility location is due to Hochbaum [Hoc82].

Stochastic Optimization. Research in (offline) stochastic optimization gives results for k -stage stochastic set cover; however, the approximation in most papers [IKMM04, RS04, SS06] is dependent on the number of stages k . The work most relevant to this paper is that of Srinivasan [Sri07], who shows how to round an LP-relaxation

of the k -stage set cover problem with only an $O(\log n)$ loss, independent of k . These techniques can be used to obtain an $O(\log n)$ approximation to the expected cost of the *best online* algorithm for stochastic set cover in $\text{poly}(mn)$ time: this is in contrast to our results that gets within $O(\log nm)$ of the best expected *offline* cost. (It is unclear if these techniques imply results for the universal problem.)

2 A Universal Algorithm for Unweighted Set Cover

In this section, we present a $O(\log mn)$ -competitive algorithm for the universal stochastic set cover problem in the unweighted case (i.e., $c(S) = 1$ for all sets $S \in \mathcal{S}$). Moreover, the proof will introduce ideas and arguments which we will extend upon for the case of weighted set cover in the following section.

Our algorithm is the following natural adaptation of the standard greedy algorithm for the set cover problem. However, its analysis is completely different from the one for the classical offline greedy algorithm. We remark that our algorithm is length-oblivious, i.e., the mapping $\mathbf{S}$ computed by the algorithm works for any sequence length k .

Algorithm 1: Universal mapping for unweighted set cover.

Data: Set system $(U, \mathcal{S})$.
while $U \neq \emptyset$ **do**
 let $S \leftarrow$ set in $\mathcal{S}$ maximizing $|S \cap U|$;
 $\mathbf{S}(v) \leftarrow S$ for each $v \in S \cap U$;
 $U \leftarrow U \setminus S$;

For the analysis, fix some sequence length k and let $\mu = \mathbf{E}_{\omega \in U^k}[\text{opt}(\omega)]$ be the expected optimal cost. We first show that there are 2μ sets which cover all but δn elements from U , where $\delta = \mu \frac{3 \ln 2m}{k}$.

Lemma 2.1 (Existence of Small Almost-Cover) *Let $(U, \mathcal{S})$ be any set system with n elements and m sets. There exists 2μ sets in $\mathcal{S}$ which cover all but δn elements from U , for $\delta = \mu \frac{3 \ln 2m}{k}$.*

Proof: Let d denote the *median* of opt , i.e., in at least half of the scenarios from U^k , the optimal solution uses at most d sets to cover all the k elements occurring in that scenario. By Markov's inequality, $d \leq 2\mu$.

There are at most $p := \sum_{j=0}^d \binom{m}{j} \leq \binom{m}{d} 2^d \leq (2m)^d$ collections of at most d sets from $\mathcal{S}$: let these collections be $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p$, and let $\cup \mathcal{C}_i$ be the union of the sets in $\mathcal{C}_i$. We now show that $|\cup \mathcal{C}_i|$ is at least $(1 - \delta)n$ for some i .

Suppose for contradiction that $|\mathcal{C}_i| < n(1 - \delta) \leq ne^{-\delta}$ for each $1 \leq i \leq p$. Since half of the n^k scenarios have a cover with at most d sets, the k elements for any such scenario can be picked from some collection $\mathcal{C}_i$. Hence,

$$\sum_{i=1}^p |\cup \mathcal{C}_i|^k \geq \frac{1}{2} n^k.$$

Plugging in $p \leq (2m)^d = e^{d \ln 2m} \leq e^{2\mu \ln 2m}$ and the assumption that each $|\cup \mathcal{C}_i| \leq ne^{-\delta}$, we get

$$p(ne^{-\delta})^k > \frac{1}{2} n^k \implies e^{(2\mu \ln 2m) - k\delta} > \frac{1}{2} \implies e^{-\mu \ln 2m} > \frac{1}{2}.$$

Since $m \geq 1$ and $\mu \geq 1$, we also get $e^{-\mu \ln 2m} \leq \frac{1}{2}$, which gives the desired contradiction. ■

We can now use the fact that for the *partial coverage problem* (pick the fewest sets to cover some $(1 - \delta)$ fraction of the elements), the greedy algorithm is a $O(\log n)$ -approximation [Kea90, Thm 5.15] to get:

Corollary 2.2 *Algorithm 1 covers at least $n(1 - \delta)$ elements using the first $O(\mu \log n)$ sets.*

Finally, we can complete the analysis of [Algorithm 1](#). (A slightly improved result will be described in [Section 4](#).)

Proof of Theorem 1.2: The first $O(\mu \log n)$ sets picked by the greedy algorithm cover all except δn elements of U , by [Corollary 2.2](#). We count all these sets as contributing to $\mathbf{E}[|\mathbf{S}|]$; note that this is fairly pessimistic.

From the remaining at most δn elements, we expect to see $\frac{k}{n}\delta n = 3\mu \ln 2m$ elements in a random sequence of length k . Whenever such an element appears we use at most one new set to cover it. Hence, in expectation, we use at most $3\mu \ln 2m$ sets for covering the elements which show up from the δn remaining elements, making the total $O(\mu(\log n + \log m))$ as claimed. ■

An Exponential-Time Variant. Surprisingly, we can trade off the $\ln n$ factor in the approximation for a worse running time; this is quite unusual for competitive analysis where the lack of information rather than lack of computational resources is the deciding factor. Instead of running the greedy algorithm to find the first $4\mu \ln n$ sets which cover $(1 - \delta)n$ elements we can run an exponential-time algorithm which finds 2μ sets which cover $(1 - \delta)n$ elements (whose existence is shown in [Lemma 2.1](#)). Thus we obtain an exponential-time universal algorithm whose expected cost is at most $O(\mu \log m)$. In [Appendix B](#) we give a polynomial-time algorithm achieving an $O(\log m)$ -competitiveness when the set system has constant VC-dimension, and also give an application of this result to the disc-cover problem.

3 The Weighted Set Cover Problem

We now consider the general (weighted) version of the universal stochastic set cover problem. As mentioned in the introduction, and in contrast to the unweighted case where we could get a length-oblivious universal mapping $\mathbf{S}$, in the weighted case there is no mapping $\mathbf{S}$ that is good for all sequence lengths k .

Theorem 3.1 *Any length-oblivious algorithm for the (weighted) universal stochastic set cover problem has a competitive ratio of $\Omega(\sqrt{n})$.*

Proof: Consider a set $S_{all} = U$ covering the whole universe, of cost $c(S_{all}) = \sqrt{n}$, and singleton sets $S_u = \{u\}$ for each $u \in U$, each of unit cost $c(S_u) = 1$. Take any length-oblivious algorithm. If this algorithm maps more than half the elements to S_{all} then the adversary can choose $k = 1$ and the algorithm pays in expectation $\Omega(\sqrt{n})$ while the optimum is 1. Otherwise (the algorithm maps less than half the elements to S_{all}), the adversary chooses $k = n$ and the algorithm pays, in expectation, $\Omega(n)$ while the optimum is at most $\sqrt{n}$. ■

Hence, we do the next best thing: we give a $O(\log mn)$ -competitive universal algorithm, which is aware of the input length k .

We first present an algorithm for computing a universal mapping $\mathbf{S}$ when given the value of $\mathbf{E}[c(\text{opt})]$. This assumption will be relaxed later, by showing that indeed the value of k is sufficient.

Algorithm 2: Universal mapping for weighted set cover.

Data: Set system $(U, \mathcal{S})$, $c: \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$, and $\mathbf{E}[c(\text{opt})]$.

while $U \neq \emptyset$ **do**

let $S \leftarrow$ set in $\mathcal{S}$ minimizing $\frac{c(S)}{|S \cap U|}$;

if $\frac{c(S)}{|S \cap U|} > \frac{64\mathbf{E}[c(\text{opt})]}{|U|}$ then let $S \leftarrow$ set in $\mathcal{S}$ minimizing $c(S)$;

$\mathbf{S}(u) \leftarrow S$ for each $u \in S \cap U$;

$U \leftarrow U \setminus S$ and $\mathcal{S} \leftarrow$ all sets covering at least one element remaining in U ;

In each iteration of [Algorithm 2](#), we either choose a set with the best ratio of cost to number of uncovered elements (*Type I* sets), or simply take the cheapest set which covers at least one uncovered element (*Type II* sets). We remark that since the value of U changes at each step, we may alternate between picking sets of Type I and

II in an arbitrary way. We also observe that both types of sets are needed in general, as the proof of [Theorem 3.1](#) shows.

We bound the cost of sets of Type I and II separately. The following lemma shows that the total cost of Type I sets is small, even in the fairly pessimistic assumption that we use all such sets to cover the random sequence ω . Since Type I sets are *min-ratio* sets, their cost can be bounded using the standard greedy analysis of set cover.

Lemma 3.2 (Type I Set Cost) *The cost of Type I sets selected by [Algorithm 2](#) is $O(\log n) \cdot \mathbb{E}[c(\text{opt})]$.*

Proof: Let $S_1, \dots, S_\ell$ be the Type I sets picked by the algorithm in this order. Moreover, let U_i denote the set of uncovered elements just before S_i was picked. Since the algorithm picked a Type I set,

$$c(S_i) \leq |S_i \cap U_i| \frac{64 \mathbb{E}[c(\text{opt})]}{|U_i|}.$$

Hence, the total cost of the sets S_i can be bounded by

$$\sum_{i=1}^{\ell} c(S_i) \leq \sum_{i=1}^{\ell} \frac{64 |S_i \cap U_i| \times \mathbb{E}[c(\text{opt})]}{|U_i|} \leq 64 \mathbb{E}[c(\text{opt})] \cdot \sum_{i=1}^{\ell} \sum_{j=1}^{|S_i \cap U_i|} \frac{1}{|U_i| - j + 1} \leq 64 \mathbb{E}[c(\text{opt})] \sum_{t=1}^n \frac{1}{t},$$

which is at most $64 \mathbb{E}[c(\text{opt})] \ln n$. ■

It remains to bound the expected cost of the Type II sets, which is also the technical heart of our argument. Let $S_1, \dots, S_\ell$ be the Type II sets selected by [Algorithm 2](#) in this order. Observe that, since Type II sets are picked on the basis of their cost alone, $c(S_i) \leq c(S_{i+1})$ for each $1 \leq i \leq \ell - 1$. Before bounding the mentioned cost ([Lemma 3.6](#)), we need a few intermediate results.

Let U_i denote the set of uncovered elements just before S_i was picked. Define $n_i = |U_i|$ and let $k_i = n_i \frac{k}{n}$ be the expected number of elements sampled from U_i . Denote by ω_i the subsequence of the input sequence ω obtained by taking only elements belonging to U_i , and let $\text{opt}|_{\omega_i}$ be the subcover obtained by taking for each $u \in \omega_i$ the cheapest set in $\text{opt} = \text{opt}_\omega$ containing u . (Note that this is *not* the optimal set cover for ω_i .) As usual, $c(\text{opt}|_{\omega_i})$ and $|\text{opt}|_{\omega_i}|$ denote the cost and number of the sets in $\text{opt}|_{\omega_i}$. Let Ω_i^q be the set of scenarios ω 's such that $|\omega_i| = q$. The proof of the following technical lemma is given in [Appendix C](#).

Lemma 3.3 *For every $i \in \{1, \dots, \ell\}$, if $k_i \geq 8 \log 2n$ then there exists $q \geq k_i/2$ such that*

$$\Pr_{\omega \in \Omega_i^q} [c(\text{opt}|_{\omega_i}) \leq 8 \mathbb{E}[c(\text{opt})] \text{ and } |\text{opt}|_{\omega_i}| \leq 8 \mathbb{E}[|\text{opt}|]] \geq \frac{1}{2}.$$

The next lemma proves that if k_i is large enough, the optimal solution uses many sets to cover the remaining elements. The observation here is similar to [Lemma 2.1](#), but now the number of sets in the set cover is not equal to its cost. This is why we needed a careful restriction of the optimal solution to subproblems given by $\text{opt}|_{\omega_i}$.

Lemma 3.4 *For every $i \in \{1, \dots, \ell\}$, if $k_i \geq 8 \log 2n$ then $k_i \leq 16 \mathbb{E}[|\text{opt}|_{\omega_i}|] \log m$.*

Proof: For a contradiction, assume that $k_i > 16 \mathbb{E}[|\text{opt}|_{\omega_i}|] \log m$, and use [Lemma 3.3](#) to define q . There are exactly n_i^q equally likely different sequences ω_i corresponding to sequences in Ω_i^q .

Let $\mathcal{S}_i$ be the family of sets $\{S \cap U_i \mid S \in \mathcal{S}\}$, and denote by $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p$ the collections of at most $8 \mathbb{E}[|\text{opt}|]$ sets from $\mathcal{S}_i$ with total cost at most $8 \mathbb{E}[c(\text{opt})]$; there are at most $(2m)^{8 \mathbb{E}[|\text{opt}|]}$ of these collections. As previously, let $\cup \mathcal{C}_j$ denote the union of the sets from $\mathcal{C}_j$. [Lemma 3.3](#) says that with probability at least $1/2$, the solution $\text{opt}|_{\omega_i}$ uses at most $8 \mathbb{E}[|\text{opt}|]$ sets and costs at most $8 \mathbb{E}[c(\text{opt})]$, hence

$$\sum_{j=1}^p |\cup \mathcal{C}_j|^q \geq \frac{1}{2} n_i^q.$$

Analogously to the proof of [Lemma 2.1](#), we can infer that there is a collection $\mathcal{C}_j$ with

$$|\cup \mathcal{C}_j| \geq \frac{n_i}{2(2m)^{8 \mathbb{E}[|\text{opt}|]/q}} \geq \frac{n_i}{2(2m)^{1/\log m}} \geq \frac{n_i}{8},$$

due to the assumption $q \geq k_i/2 > 8\mathbf{E}[|\text{opt}|_{\omega_i}] \log m$. Since the total cost of sets in $\mathcal{C}_j$ is at most $8\mathbf{E}[c(\text{opt})]$ and they cover $n_i/8$ elements from U_i , there is a set $S \in \mathcal{C}_j$ with

$$\min_{S \in \mathcal{C}_j} \frac{c(S)}{|S \cap U_i|} \leq \frac{\sum_{S \in \mathcal{C}_j} c(S)}{\sum_{S \in \mathcal{C}_j} |S \cap U_i|} \leq \frac{8\mathbf{E}[c(\text{opt})]}{n_i/8} = \frac{64\mathbf{E}[c(\text{opt})]}{n_i}.$$

However, the Type II set S_i was picked by the algorithm because there were no sets for which $\frac{c(S)}{|S \cap U_i|} < \frac{64\mathbf{E}[c(\text{opt})]}{|U_i|}$, so we get a contradiction and the lemma follows. $\blacksquare$

The following lemma relates the cost of the sets S_i with the cardinality and cost of the optimum solution.

Lemma 3.5 *For each $1 \leq i \leq \ell$,*

$$c(S_i)\mathbf{E}[|\text{opt}|_{\omega_{i+1}}] \leq \mathbf{E}[c(\text{opt})_{\omega_{i+1}}] \quad \text{and} \quad c(S_i)(\mathbf{E}[|\text{opt}|_{\omega_i}] - \mathbf{E}[|\text{opt}|_{\omega_{i+1}}]) \leq \mathbf{E}[c(\text{opt})_{\omega_i}] - \mathbf{E}[c(\text{opt})_{\omega_{i+1}}].$$

Proof: The set S_{i+1} is the cheapest set covering any element of U_{i+1} , and hence $c(S_{i+1})$ is a lower bound on the cost of the sets in $\text{opt}|_{\omega_{i+1}}$. Since by definition $c(S_i) \leq c(S_{i+1})$,

$$c(S_i)|\text{opt}|_{\omega_{i+1}} \leq c(S_{i+1})|\text{opt}|_{\omega_{i+1}} \leq c(\text{opt})_{\omega_{i+1}}.$$

Analogously, the number of sets opt uses to cover the elements $U_i \setminus U_{i+1}$ covered by S_i is given by $|\text{opt}|_{\omega_i} - |\text{opt}|_{\omega_{i+1}}$, and to cover each of those elements opt pays at least $c(S_i)$. Thus,

$$c(S_i)(|\text{opt}|_{\omega_i} - |\text{opt}|_{\omega_{i+1}}) \leq c(\text{opt})_{\omega_i} - c(\text{opt})_{\omega_{i+1}}.$$

Taking expectations on the inequalities gives the lemma. $\blacksquare$

Finally, we can bound the expected cost of Type II sets: recall that we incur the cost of some set S_i only if one of the corresponding elements $S_i \cap U_i$ is sampled.

Lemma 3.6 (Type II Set Cost) *The expected cost of Type II sets selected by Algorithm 2 is $O(\log mn) \mathbf{E}[c(\text{opt})]$.*

Proof: Recall that the Type II sets were $S_1, S_2, \dots, S_\ell$. Set $k_{\ell+1} = 0$ and $c(S_0) = 0$ for notational convenience. Moreover, let j be such that $k_j \geq 8 \log 2n$ but $k_{j+1} < 8 \log 2n$. Hence, in expectation we see at most $8 \log 2n$ elements from U_{j+1} , and since each of these elements is covered by a set that does not cost more than the one covering it in opt , the cost incurred by using the sets $S_{j+1}, \dots, S_\ell$ is bounded by $8 \log 2n \mathbf{E}[c(\text{opt})]$.

By Lemma 3.4, the expected cost incurred by using the remaining sets $S_1, \dots, S_j$ is at most

$$\begin{aligned} \sum_{i=1}^j c(S_i) \Pr[\omega \cap (S_i \cap U_i) \neq \emptyset] &\leq \sum_{i=1}^j c(S_i) \mathbf{E}[|\omega \cap (S_i \cap U_i)|] = \sum_{i=1}^j c(S_i) \mathbf{E}[|\omega \cap (U_i \setminus U_{i+1})|] \leq \sum_{i=1}^j c(S_i) (k_i - k_{i+1}) \\ &\leq \sum_{i=1}^j k_i (c(S_i) - c(S_{i-1})) \leq \sum_{i=1}^j 16 \mathbf{E}[|\text{opt}|_{\omega_i}] \log m \cdot (c(S_i) - c(S_{i-1})) \\ &= 16 \log m \cdot \left(c(S_j) \mathbf{E}[|\text{opt}|_{\omega_{j+1}}] + \sum_{i=1}^j c(S_i) (\mathbf{E}[|\text{opt}|_{\omega_i}] - \mathbf{E}[|\text{opt}|_{\omega_{i+1}}]) \right). \quad (3.1) \end{aligned}$$

It follows by Lemma 3.5 that the expected cost due to the sets $S_1, \dots, S_j$ is at most

$$16 \log m \cdot (\mathbf{E}[c(\text{opt})_{\omega_{j+1}}] + \sum_{i=1}^j (\mathbf{E}[c(\text{opt})_{\omega_i}] - \mathbf{E}[c(\text{opt})_{\omega_{i+1}}])) = 16 \mathbf{E}[c(\text{opt})_{\omega_1}] \log m \leq 16 \mathbf{E}[c(\text{opt})] \log m,$$

concluding the proof of the lemma. $\blacksquare$

We have all the ingredients to prove the main result of this section.

Proof of Theorem 1.1: Lemmas 3.2 and 3.6 together imply that Algorithm 2 is $O(\log mn)$ -competitive. It remains to show how this result can be adapted to the situation when, instead of $\mathbf{E}[c(\text{opt})]$, we are given in input the sequence length k .

Algorithm 2 uses the value of $\mathbf{E}[c(\text{opt})]$ only in comparison with $\frac{c(S) \cdot |U|}{|S \cap U|}$ for different sets S . This fraction can take at most nm different values, and thus the algorithm can generate at most $nm + 1$ different mappings $\{\mathbf{S}_i\}_{i=1}^{nm+1}$. For any such map $\mathbf{S}$, computing the expected cost $\mathbf{E}[c(\mathbf{S})]$ is easy: indeed, if $\mathbf{S}^{-1}(S)$ is the pre-image of $S \in \mathcal{S}$, then

$$\mathbf{E}[c(\mathbf{S})] = \sum_{S \in \mathcal{S}} c(S) \cdot \Pr[\omega \cap \mathbf{S}^{-1}(S) \neq \emptyset].$$

The value of k is sufficient (and necessary) to compute the probabilities above. Hence, we can select the mapping $\mathbf{S}_i$ with the minimum expected cost for the particular value k ; this cost is at most the cost of the mapping generated with the knowledge of $\mathbf{E}[c(\text{opt})]$. ■

4 Matching Bounds

In this section we present slightly refined upper bounds and matching lower bounds for universal stochastic set cover.

If we stay within polynomial time, and if $m = \text{poly}(n)$, then the resulting $O(\log mn) = O(\log n)$ competitive factor is asymptotically the best possible given suitable complexity-theoretic assumptions. However, for the cases when $m \gg n$, we can show a better dependence on the parameters.

Let us slightly modify the universal algorithm for weighted set cover as follows: fixing a value $0 < x \leq \log m$, the set S minimizing $c(S)/|S \cap U|$ is selected only if $c(S)/|S \cap U| > 64 \cdot 2^x \mathbf{E}[c(\text{opt})]/|U|$. By adapting the analysis, the cost of Type I sets is bounded by $O(2^x \log n) \mathbf{E}[c(\text{opt})]$, and the expected cost of Type II sets is $O(\log n + \frac{\log m}{x}) \mathbf{E}[c(\text{opt})]$. A similar result can be shown for Algorithm 1, in the unweighted (length-oblivious) case. Setting x suitably (details appear in the full version), we get:

Theorem 4.1 *For $m > n$, there exists a polynomial-time length-aware (resp. length-oblivious) $O\left(\frac{\log m}{\log \log m - \log \log n}\right)$ -competitive algorithm for the weighted (resp. unweighted) universal stochastic set cover problem.*

The following theorem (which extends directly to *online* stochastic set cover) shows that the bounds above are tight.

Theorem 4.2 *There are values of m and n such that any mapping $\mathbf{S}$ for the (unweighted) universal stochastic set cover problem satisfies $\mathbf{E}[|\mathbf{S}|] = \Omega\left(\frac{\log m}{\log \log m - \log \log n}\right) \mathbf{E}[|\text{opt}|]$.*

Proof: Consider an n element universe $U = \{1, \dots, n\}$ with the uniform distribution over the elements, and $\mathcal{S}$ consisting of all $m = \binom{n}{\sqrt{n}}$ subsets of U of size $\sqrt{n}$; hence $\log m = \Theta(\sqrt{n} \log n)$ and $\log \log m - \log \log n = \Theta(\log n)$. Let the sequence length be $k = \sqrt{n}/2$. Consider any mapping. The sets included in the solution covering the first i elements cover at most $i\sqrt{n} \leq \frac{n}{2}$ of the total elements. Hence, with probability at least half, the mapping must pick a new set to cover the $(i+1)$ -th element. Hence, in expectation the mapping picks $\frac{\sqrt{n}}{4}$ sets while the optimal solution is always one proving the lemma. ■

5 Online Stochastic Set Cover

The universal algorithm for (weighted) stochastic set cover can be turned into an online algorithm with the same $O(\log mn)$ competitive ratio. The basic idea is using the universal mapping from Section 3 to cover each new element, and update the mapping from time to time. The main difficulty is choosing the update points properly:

indeed, the standard approach of updating the mapping each time the number of elements doubles does not work here.

Let ω^i denote a random sequence of i elements, and let $\mathbf{S}_i$ be the mapping produced by the universal set cover algorithm from Section 3 for a sequence of length i . Our algorithm works as follows. Let k be the current number of samplings performed. The algorithm maintains a variable k' , initially set to 1, which is larger than k at any time. For a given value of k' , the mapping used by the online algorithm is the universal mapping $\mathbf{S}_{k'}$. When $k = k'$, we update k' to the smallest value $k'' > k'$ which satisfies $\mathbf{E}[c(\mathbf{S}_{k''}(\omega^{k''}))] > 2\mathbf{E}[c(\mathbf{S}_{k'}(\omega^{k'}))]$ and modify the mapping consequently (we set $k' = \infty$ if such value k'' does not exist). We remark that the algorithm above takes polynomial time per sample, and does not assume any knowledge of the final number of samplings. The proof of Theorem 1.3 is given in Appendix D.

6 Extensions and Applications

Our techniques can be applied to other *covering-like* problems. In this section we sketch three such applications.

6.1 Universal Stochastic Facility Location

In this section we consider the universal stochastic version of (non-metric) facility location, a generalization of set cover. For this problem, we provide a $O(\log n)$ -competitive algorithm, where n is the total number of clients and facilities, under the assumption that the sequence length k is known.

The *universal stochastic facility location* problem is defined as follows. An instance of the problem is a set of clients C and a set of facilities F , with a (possibly non-metric) *distance* function $d: C \times F \rightarrow \mathbb{R}_{\geq 0}$. Each facility $f \in F$ has an opening cost $c(f) \geq 0$. We let $n = |F| + |C|$. Given a mapping $\mathbf{S}: C \rightarrow F$ of clients into facilities, and a subset $X \subseteq C$, we define $c(\mathbf{S}(X))$ as the total cost of opening facilities in $\mathbf{S}(X) = \cup_{u \in X} \mathbf{S}(u)$ plus the total distance from each $u \in X$ to the closest facility in $\mathbf{S}(X)$. We also denote by $|\mathbf{S}(X)|$ the number of facilities in $\mathbf{S}(X)$. With the usual notation, the aim is finding a mapping which minimizes $\mathbf{E}_\omega[c(\mathbf{S}(\omega))]/\mathbf{E}_\omega[c(\text{opt}(\omega))]$, where ω is a random sequence of k clients.

Our algorithm is an extension of the algorithm from Section 3, where the new challenge is to handle the connection costs for clients. As for weighted set cover, we first assume that the algorithm is given as input (a constant approximation for) the value $\mathbf{E}[c(\text{opt})]$; we later show how to remove this assumption.

Algorithm 3: Algorithm for the (weighted) stochastic facility location problem.

Data: $C, F, d: C \times F \rightarrow \mathbb{R}_{\geq 0}, c: F \rightarrow \mathbb{R}_{\geq 0}, k$ and $\mathbf{E}[c(\text{opt})]$.

while $C \neq \emptyset$ **do**

let $f \in F$ and $S \subseteq C$ minimize $\text{avg} := \frac{c(f) + \frac{k}{n} \sum_{v \in S} d(v, f)}{|S \cap C|}$;
if $\text{avg} > \frac{192\mathbf{E}[c(\text{opt})]}{|C|}$ **then** let $f \in F$ and $S = \{v\} \subseteq C$ minimize $c(f) + d(v, f)$;
 $\mathbf{S}(u) \leftarrow S$ for each $u \in S \cap C$;
 $C \leftarrow C \setminus S$;

The first step in the while loop can be implemented in polynomial time even if the number of candidate sets S is exponential, since it suffices to consider, for each facility f , the closest i clients still in C , for every $i = 1, \dots, |C|$. Due to space limitations, the proof of the following lemma is given in Appendix E.

Lemma 6.1 *Given k and a constant approximation to $\mathbf{E}[c(\text{opt})]$, Algorithm 3 returns a universal mapping $\mathbf{S}$ to the universal stochastic facility location problem with $\mathbf{E}[c(\mathbf{S})] = O(\log n)\mathbf{E}[c(\text{opt})]$.*

Now, using the above lemma it is easy to prove the main theorem.

Theorem 6.2 *There exists a polynomial-time length-aware algorithm that returns a universal mapping $\mathbf{S}$ to the universal stochastic facility location problem with $\mathbf{E}[c(\mathbf{S})] = O(\log n)\mathbf{E}[c(\text{opt})]$.*

Proof: First, note that the value of $\mathbf{E}_{\omega \in C^1}[c(\text{opt}(\omega))]$ can be easily computed, by finding for each $v \in C$ the facility f minimizing $c(f) + d(c, f)$. Trivially, $\mathbf{E}_{\omega \in C^1}[c(\text{opt}(\omega))] \leq \mathbf{E}_{\omega \in C^k}[c(\text{opt}(\omega))] \leq \mathbf{E}_{\omega \in C^n}[c(\text{opt}(\omega))]$ for $1 \leq k \leq n$. Moreover, by subadditivity, $\mathbf{E}_{\omega \in C^n}[c(\text{opt}(\omega))] \leq n\mathbf{E}_{\omega \in C^1}[c(\text{opt}(\omega))]$.

Hence one of the values $x_i := 2^i \mathbf{E}_{\omega \in C^1}[c(\text{opt}(\omega))]$ for $0 \leq i \leq \log n$ is a 2-approximation for $\mathbf{E}_{\omega \in C^k}[c(\text{opt}(\omega))] = \mathbf{E}[c(\text{opt})]$. Therefore, we can run Algorithm 3 for all $\log n$ values x_i in order to obtain $\log n$ different mappings. Afterwards, we can choose the one with the smallest expected cost, which is guaranteed to be $O(\log n)$ approximate. The expected costs above can be computed analogously to the set cover case. ■

Essentially the same reduction as in Section 5 leads to an $O(\log n)$ -competitive algorithm for the online version of the problem.

Theorem 6.3 *There is an $O(\log n)$ -competitive algorithm for the online stochastic facility location problem.*

6.2 Universal Stochastic Multi-Cut

In an instance of the *universal multi-cut* problem we are given a graph $G = (V, E)$ with edge costs $c: E \rightarrow \mathbb{R}_{\geq 0}$, and a set of demand pairs $D = \{(s_i, t_i) : 1 \leq i \leq m\}$. The task is to return a mapping $\mathbf{S}: D \rightarrow 2^E$ so that $\mathbf{S}((s_i, t_i)) \subseteq E$ disconnects s_i from t_i . The cost of the solution for a sequence $\omega \in D^k$ is defined as usual to be $c(\mathbf{S}(\omega))$ —the total cost of edges in $\mathbf{S}(\omega)$. The universal and online stochastic versions are defined analogously, and again the goal is to minimize the ratio $\mathbf{E}_\omega[c(\mathbf{S}(\omega))]/\mathbf{E}_\omega[c(\text{opt}(\omega))]$.

Notice first, that multi-cut in trees (i.e., G is a tree) is a special case of weighted set cover: each demand pair (s_i, t_i) is an element in U , each edge e corresponds to a set S_e , and an element (s_i, t_i) is contained in a set S_e if e is in the unique path from s_i to t_i . Thus we can use the algorithm from Section 3 to obtain a $O(\log n)$ -competitive algorithm for stochastic universal multi-cut in trees. Using results from Räcke [Räc08], we can generalize this result to general graphs obtaining a $O(\log^2 n)$ -competitive algorithm. The proof of the following theorem is omitted due to space constraints.

Theorem 6.4 *There exists an $O(\log^2 n)$ -competitive polynomial-time algorithm for the online multi-cut problem, and a polynomial-time algorithm that, given the length of the input sequence, is $O(\log^2 n)$ -competitive for the universal multi-cut problem.*

6.3 Disc Covering in the Plane

Consider a region $U \subseteq \mathbb{R}^2$ of the 2-dimensional plane, and a set of m “base-stations” $v_i \in \mathbb{R}^2$, each with a coverage radius r_i , such that $U \subseteq \cup_i \mathbf{B}(v_i, r_i)$; i.e., the discs cover the entire region. Given a set $X \subseteq U$, the goal is to find a small set cover, i.e., to map each point $x \in X$ to a base-station covering it so that not too many base-stations are in use. This problem was studied by Hochbaum and Maas [HM85], and by Bronnimann and Goodrich [BG95]: among other results, they gave a constant-factor approximation for the problem based on set cover for set systems with small VC-dimension.

However, one might want to hard-wire this mapping from locations in the plane to base-stations, so that we do not have to solve a set-cover problem each time a device wants to access a base-station; i.e., we want a *universal map*. For ease of exposition, let us discretize the plane into n points by placing a fine-enough mesh on the plane. We can then use the arguments from Appendix B to show that for the case of points chosen randomly from some known distribution from the plane (or more precisely, from this mesh), there *exists* a universal map whose expected set-cover cost is at most $O(\log m)$ times the expected optimum. Moreover, using the k -coverage algorithm for set systems of finite VC-dimension from the same section, we can also find such a universal map in randomized polynomial-time.

References

- [AAA⁺03] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The Online Set Cover Problem. In *STOC'03: Proceedings of the 35th Annual ACM Symposium on the Theory of Computation*, pages 100–105, 2003.
- [AAA⁺04] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. A general approach to online network optimization problems. In *SODA'04: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 577–586, 2004.
- [AAG05] Noga Alon, Yossi Azar, and Shai Gutner. Admission control to minimize rejections and online set cover with repetitions. In *SPAA'05: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 238–244, 2005.
- [ACF⁺03] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *STOC'03: Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 383–388, 2003.
- [AL99] Susanne Albers and Stefano Leonardi. On-line algorithms. *ACM Comput. Surv.*, 31(3es):4, 1999.
- [AMS06] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.
- [BEHW86] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Classifying Learnable Geometric Concepts with the Vapnik-Chervonenkis Dimension (Extended Abstract). In *STOC'86: Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1986.
- [BEY98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, 1998.
- [BG89] Dimitris Bertsimas and Michelangelo Grigni. Worst-case examples for the spacefilling curve heuristic for the Euclidean traveling salesman problem. *Oper. Res. Lett.*, 8(5):241–244, 1989.
- [BG95] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14(4):463–479, 1995. ACM Symposium on Computational Geometry (Stony Brook, NY, 1994).
- [BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA'07: Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 2007.
- [BJN07] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA'07: Proceedings of the 15th Annual European Symposium on Algorithms*, pages 253–264, 2007.
- [BJO90] Dimitris J. Bertsimas, Patrick Jaillet, and Amedeo R. Odoni. A priori optimization. *Oper. Res.*, 38(6):1019–1033, 1990.
- [BKR03] Marcin Bienkowski, Mirosław Korzeniowski, and Harald Räcke. A practical algorithm for constructing oblivious routing schemes. In *SPAA'03: Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 24–33, 2003.
- [BN05] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing problems. In *ESA'05: Proceedings of the 13th Annual European Symposium on Algorithms*, pages 689–701, 2005.
- [CFJ06] Jean Cardinal, Samuel Fiorini, and Gwenaél Joret. Tight results on minimum entropy set cover. In *APPROX'06: Proceedings of the 9th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems*, pages 61–69, 2006.
- [Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
- [DLO05] Reza Dorrigiv and Alejandro Lopez-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.
- [Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

- [FK] Uriel Feige and Simon Korman. On the use of randomization in the online set cover problem. *Technical Report*.
- [FLT04] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.
- [Fot03] Dimitris Fotakis. On the competitive ratio for online facility location. In *ICALP'03: Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, pages 637–652. 2003.
- [Fre83] P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.
- [FW98] Amos Fiat and Gerhard J. Woeginger, editors. *Online algorithms*, volume 1442 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998. The state of the art, Papers from the Workshop on the Competitive Analysis of On-line Algorithms held in Schloss Dagstuhl, June 1996.
- [GGLS08] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA'08: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 942–951, 2008.
- [GHR06] Anupam Gupta, Mohammad Taghi Hajiaghayi, and Harald Räcke. Oblivious network design. In *SODA'06: Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 970–979, 2006.
- [GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA'08: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 982–991, 2008.
- [HHR03] Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *SPAA'03: Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 34–43, 2003.
- [HK05] Eran Halperin and Richard M. Karp. The minimum-entropy set cover problem. *Theoret. Comput. Sci.*, 348(2-3):240–250, 2005.
- [HKL06] Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Frank Thomson Leighton. Improved lower and upper bounds for universal tsp in planar metrics. In *SODA'06: Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 649–658, 2006.
- [HKLR05a] Mohammad T. Hajiaghayi, Robert D. Kleinberg, Tom Leighton, and Harald Räcke. Oblivious routing on node-capacitated and directed graphs. In *SODA'05: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 782–790, 2005.
- [HKLR05b] Mohammad Taghi Hajiaghayi, Jeong Han Kim, Tom Leighton, and Harald Räcke. Oblivious routing in directed graphs with random demands. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 193–201, 2005.
- [HKP04] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *EC'04: Proceedings of the 5th ACM conference on Electronic Commerce*, pages 71–80, 2004.
- [HM85] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.
- [Hoc82] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, Dec 1982.
- [IKMM04] Nicole Immorlica, David Karger, Maria Minkoff, and Vahab Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *SODA'04: Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 684–693, 2004.
- [Jai88] Patrick Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Oper. Res.*, 36(6):929–936, 1988.
- [JLN⁺05] Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for tsp, steiner tree, and set cover. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 386–395, 2005.

- [Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.
- [Kea90] Michael J. Kearns. *The Computational Complexity of Machine Learning*. MIT Press, 1990.
- [Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA’05: Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 630–631, 2005.
- [KM00] David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. In *FOCS’00: Proceedings of the 41th Symposium on the Foundations of Computer Science*, pages 613–623, 2000.
- [Lov75] László Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13(4):383–390, 1975.
- [LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.
- [Mey01] Adam Meyerson. Online facility location. In *FOCS’01: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431. 2001.
- [MMP01] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *FOCS’01: Proceedings of the 42nd Symposium on the Foundations of Computer Science*, pages 406–415, 2001.
- [MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. AdWords and generalized online matching. *J. ACM*, 54(5):Art. 22, 19 pp. (electronic), 2007.
- [PB89] Loren K. Platzman and John J. Bartholdi, III. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36(4):719–737, 1989.
- [Räc02] Harald Räcke. Minimizing congestion in general networks. In *FOCS’02: Proceedings of the 43rd Symposium on the Foundations of Computer Science*, pages 43–52, 2002.
- [Räc08] Harald Räcke. Optimal Hierarchical Decompositions for Congestion Minimization in Networks. In *STOC’08: Proceedings of the 40th Symposium on Theory of Computing*, 2008.
- [RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC’97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.
- [RS04] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *IPCO’04: Proceedings of the 10th Integer Programming and Combinatorial Optimization Conference*, pages 101–115, 2004.
- [Sri07] Aravind Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *SODA’07: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1305–1313, 2007.
- [SS06] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006.
- [SS08] Frans Schalekamp and David B. Shmoys. Algorithms for the universal and a priori tsp. *Oper. Res. Lett.*, 36(1):1–3, Jan 2008.
- [ST08] David B. Shmoys and Kunal Talwar. A constant approximation algorithm for the a priori traveling salesman problem. In *IPCO’08: Proceedings of the 15th Integer Programming and Combinatorial Optimization Conference*, 2008.
- [VB81] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *STOC’81: Proceedings of the 13th Annual ACM symposium on Theory of computing*, pages 263–277, 1981.
- [Vöc01] Berthold Vöcking. Almost optimal permutation routing on hypercubes. In *STOC’01: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 530–539, 2001.

A Handling Non-Uniform Probability Distributions

We show that given an arbitrary distribution we can convert it to the uniform distribution. The only assumption that we need is that the algorithms have polynomial approximation ratios in the worst case, which is the case for all algorithms presented here. Assume we are given an $\alpha(n, m)$ -competitive algorithm that is n^β -worst-case competitive, for some function α and a constant β . We replace each element $u \in U$ with $\lceil n^{\beta+1} \pi(u) \rceil$ copies of u in all the sets containing u . Denote the set system obtained this way by $(U', \mathcal{S}')$. Note that a uniform distribution on U' simulates the given distribution π in such a way that elements with $\pi(u) \geq \frac{1}{n^{\beta+1}}$ are generated with probability changed only by a factor of at most 2. Hence, the algorithm gives an $2\alpha(n, m)$ approximation on the sequences when only such elements are generated. The cost of all other sequences can be bounded by $n^\beta \times n \frac{1}{n^{\beta+1}} \mathbf{E}[c(\text{opt}_\omega)] = \mathbf{E}[c(\text{opt}_\omega)]$. Hence, finally we get an $2\alpha(n^{\beta+1}, m) + 1$ competitive algorithm. In particular, for $\alpha(n, m) = O(\log(mn))$ the competitive factor is $O(\log(mn))$.

The following lemmas allow us to use this idea to the algorithms presented here.

Lemma A.1 *Any universal mapping for the unweighted set cover problem is n -approximate in the worst case.*

Proof: This is trivial the optimal solution needs at least one set whereas the mapping returns at most n sets. ■

Lemma A.2 *The universal mapping $\mathbf{S}$ generated by Algorithm 2 is n^2 -approximate in the worst case for the set cover problem.*

Proof: Consider an element x assigned by $\mathbf{S}$ to the *Type I* set S . It is easy to see that S weights no more than n times the cost of the cheapest set covering x . Moreover, the *Type II* sets are the cheapest sets. We get the n^2 bound by observing that the optimal solution cost no more than a cheapest sets covering each element. ■

A similar argument holds for the facility location problem as well. This justifies our assumption that π is a uniform probability distribution.

Lemma A.3 *The universal mapping $\mathbf{S}$ generated by Algorithm 3 is n^3 -approximate in the worst case for the facility location problem.*

Proof: Consider elements in S assigned by $\mathbf{S}$ to the *Type I* facility f . Let f_v be the facility minimizing $c(f) + d(v, f)$ over all f . In the worst case $\mathbf{S}$ for connecting v pays

$$\begin{aligned} c(f) + d(v, f) &\leq c(f) + \sum_{v \in S} d(v, f) \leq n \cdot \left(c(f) + \frac{k}{n} \sum_{v \in S} d(v, f) \right) \leq \\ &\leq n^2 \cdot \left(\frac{c(f) + \frac{k}{n} \sum_{v \in S} d(v, f)}{|S \cap C|} \right) \leq n^2 \cdot (c(f_v) + d(v, f_v)). \end{aligned}$$

For elements assigned to *Type II* sets we pay no more than $c(f_v) + d(v, f_v)$. Now, optimal solution pays at least $c(f_v) + d(v, f_v)$ for each v , hence $\mathbf{S}$ pays no more than n^3 times more. ■

B Constant VC-Dimension

Here we achieve $O(\log m)$ -competitive factor in polynomial time when the set system has constant VC-dimension developing on the algorithm in Section 2.

The main bottleneck step in achieving a polynomial-time algorithm is finding the 2μ sets which cover all but δn elements. When the set system has constant VC-dimension we implement this step in polynomial time. To be precise, we give a polynomial-time algorithm which finds $O(d\mu \log d\mu)$ sets which cover all but $2\delta n$ elements

when the set system has VC-dimension d . Combining this with the argument in [Theorem 1.2](#) this implies a polynomial-time $O(\log m)$ -competitive algorithm.

The algorithm builds on the set cover algorithm given by Bronnimann and Goodrich [BG95] for fixed VC-dimension. The only difference is that in our problem we want a *partial cover*: a small collection of sets which cover $(1 - \delta)n$ elements and not all the elements. We will briefly describe here the differences in our case and the reader is assumed to be familiar with [BG95].

Following [BG95], we describe an algorithm for the dual *hitting set* problem. The dual set system $\mathcal{S}^* = \{\mathbb{R}_x : x \in U\}$ where $\mathbb{R}_x$ is the set of all sets R of $\mathcal{S}$ which contain x . It is well-known that the VC-dimension of the dual is at most 2^{d+1} where the VC-dimension of the primal is at most d . In the dual problem, sets correspond to elements of the primal problem and elements correspond to sets of the dual problem. The cover of size 2μ sets which cover all but δn elements corresponds to a hitting set of size 2μ which hits all but δn sets. Given a weight function w on elements and parameter $\varepsilon > 0$, an ε -net A is a set of elements which hits all sets with weight at least $\varepsilon w(U)$ where the weight of any set is defined to be the sum of the weight of the elements in the set.

We now describe the algorithm. The algorithm proceeds in iterations. To initialize, we give a weight of one to each element and set $\varepsilon = \frac{1}{4|H|}$ where H is the set of size at most 2μ which hits all but δn sets. In each iteration, we find a ε -net A . Such a ε -net of size $O(d\mu \log d\mu)$ exists and can be found in polynomial time [BEHW86]. If the set A hits all but $2\delta n$ sets we stop. Else, we pick a set R at random which is not hit and double the weight of elements in R and go to the next iteration.

Clearly, the hitting set returned by the algorithm is of appropriate size. It remains to show that the algorithm terminates in polynomial time.

Lemma B.1 *The doubling procedure can proceed for at most $O(\mu \log \frac{n}{\mu})$ iterations.*

Proof: Let t be the total number of iterations done by the algorithm. Any set R which is not hit by A must have weight at most $\varepsilon w(U)$ since A is an ε -net. In each iteration we double the weight of one such set R and therefore, the weight of U increases by at most a factor of $(1 + \varepsilon)$ in each iteration. Thus,

$$w(U) \leq n(1 + \varepsilon)^t \leq ne^{\varepsilon t}.$$

Let H denote the set of size at most 2μ which hit all but δn sets which is guaranteed by Lemma 2.1. Since, at least $2\delta n$ sets are not hit by A , at least half of the sets not hit by A are hit by H . Therefore, with probability at least $\frac{1}{2}$ we will double the weight of some element in H in each iteration. For each $h \in H$, let Z_h be the random variable denoting the number of iterations where the weight of the element h is doubled and $E[Z_h]$ denote the expectation of Z_h . Thus we have

$$w(H) = \sum_{h \in H} 2^{Z_h}, \text{ where } \sum_{h \in H} E[Z_h] \geq \frac{t}{2}.$$

Using the fact that $E[2^Y] \geq 2^{E[Y]}$ for any random variable Y (from Holder's inequality) and using convexity of the exponential function and $E[w(H)] \leq E[w(U)]$ we have

$$|H|2^{\frac{t}{2|H|}} \leq ne^{\frac{t}{4|H|}}$$

from which we have $t \leq 8|H| \log \frac{n}{|H|} \leq 16\mu \log \frac{n}{2\mu}$ as required. ■

C Proofs from Section 3

Proof of Lemma 3.3: We restrict our attention to scenarios in $\Omega_i^{\geq k_i/2} := \uplus_{p \geq k_i/2} \Omega_i^p$, i.e., scenarios where the sampled k elements contain at least $\frac{k_i}{2}$ elements from U_i . Let d_i be the upper quartile of $|\text{opt}|_{\omega_i}|$, i.e., in at least

three-quarters of the scenarios in Ω_i , the optimal solution $\text{opt} = \text{opt}(\omega)$ uses at most d_i sets to cover the elements in the scenario. A Chernoff's bound implies that $\Pr[|\omega_i| < k_i/2] \leq \exp\left(-\frac{(1/2)^2 8 \log 2n}{2}\right) \leq \frac{1}{2n}$. Hence, conditioning on the event $\omega \in \Omega_i^{\geq k_i/2} = \{\omega : |\omega_i| \geq k_i/2\}$ and observing that $\frac{1}{1-(1/2n)} \leq 2$, we obtain

$$\mathbf{E}[c(\text{opt}|_{\omega_i}) \mid \omega \in \Omega_i^{\geq k_i/2}] \leq 2\mathbf{E}[c(\text{opt}|_{\omega_i})] \quad \text{and} \quad \mathbf{E}[|\text{opt}|_{\omega_i}| \mid \omega \in \Omega_i^{\geq k_i/2}] \leq 2\mathbf{E}[|\text{opt}|_{\omega_i}|]. \quad (\text{C.2})$$

By (C.2) and the definition of $\text{opt}|_{\omega_i}$,

$$d_i \leq 4\mathbf{E}[|\text{opt}|_{\omega_i}| \mid \omega \in \Omega_i^{\geq k_i/2}] \leq 8\mathbf{E}[|\text{opt}|_{\omega_i}|] \leq 8\mathbf{E}[|\text{opt}_\omega|].$$

An analogous argument shows that the cost $c(\text{opt}|_{\omega_i})$ is at most $8\mathbf{E}[c(\text{opt}_\omega)]$ with probability at least $3/4$. Hence, a trivial union bound implies that

$$\Pr_{\omega \in \Omega_i^{\geq k_i/2}} [c(\text{opt}|_{\omega_i}) \leq 8\mathbf{E}[c(\text{opt})] \wedge |\text{opt}|_{\omega_i} \leq 8\mathbf{E}[|\text{opt}|]] \geq \frac{1}{2}.$$

Since $\Omega_i^{\geq k_i/2} = \uplus_{p \geq k_i/2} \Omega_i^p$, an averaging argument implies that some $q \geq k_i/2$ satisfies the lemma. $\blacksquare$

D Proofs from Section 5

Proof of Theorem 1.3: Let $k \geq 1$ be the final number of samplings performed, and $\mathbf{S}$ be the actual mapping computed by the algorithm. Let moreover $1 = k_1, k_2, \dots, k_h > k$ be the sequence of different values of k' computed by the algorithm. For $h = 1$ the analysis is trivial, so assume $h \geq 2$ and hence $k_h \geq 2$. By the choice of the k_i 's,

$$\begin{aligned} \mathbf{E}[c(\mathbf{S}(\omega^k))] &\leq \mathbf{E}[c(\mathbf{S}_{k_1}(\omega^{k_1}))] + \mathbf{E}[c(\mathbf{S}_{k_2}(\omega^{k_2-k_1}))] + \dots + \mathbf{E}[c(\mathbf{S}_{k_h}(\omega^{k-k_{h-1}}))] \\ &\leq \mathbf{E}[c(\mathbf{S}_{k_1}(\omega^{k_1}))] + \mathbf{E}[c(\mathbf{S}_{k_2}(\omega^{k_2}))] + \dots + \mathbf{E}[c(\mathbf{S}_{k_h}(\omega^{k_h}))] \\ &\leq 2\mathbf{E}[c(\mathbf{S}_{k_{h-1}}(\omega^{k_{h-1}}))] + \mathbf{E}[c(\mathbf{S}_{k_h}(\omega^{k_h}))]. \end{aligned}$$

By construction, $\mathbf{E}[c(\mathbf{S}_{k_{h-1}}(\omega^{k_{h-1}}))] \leq 2\mathbf{E}[c(\mathbf{S}_{k_{h-1}}(\omega^{k_{h-1}}))]$: this is trivially true for $k_h = \infty$ and holds by the minimality of k_h otherwise.

We need the following technical Lemma.

Lemma D.1 $\forall i \geq 1, \quad \mathbf{E}[c(\mathbf{S}_i(\omega^i))] \leq \mathbf{E}[c(\mathbf{S}_{i+1}(\omega^{i+1}))] \quad \text{and} \quad \mathbf{E}[c(\mathbf{S}_i(\omega^i))] \leq 2\mathbf{E}[c(\mathbf{S}_{\lceil i/2 \rceil}(\omega^{\lceil i/2 \rceil}))].$

Proof: We observe that the pool of possible universal mappings from which each $\mathbf{S}_i$ is chosen is the same for every value of i (i.e. one for every possible breaking point). Moreover, the expected cost of each such mapping is an increasing function of the length of the sequence. As a consequence, $\mathbf{E}[c(\mathbf{S}_i(\omega^i))] \leq \mathbf{E}[c(\mathbf{S}_{i+1}(\omega^{i+1}))] \leq \mathbf{E}[c(\mathbf{S}_{i+1}(\omega^{i+1}))]$. The second claim follows along the same line. $\blacksquare$

It follows from Lemma D.1 that

$$\mathbf{E}[c(\mathbf{S}_{k_h}(\omega^{k_h}))] \leq 2\mathbf{E}[c(\mathbf{S}_{\lceil k_h/2 \rceil}(\omega^{\lceil k_h/2 \rceil}))] \leq 2\mathbf{E}[c(\mathbf{S}_{k_{h-1}}(\omega^{k_{h-1}}))] \leq 4\mathbf{E}[c(\mathbf{S}_{k_{h-1}}(\omega^{k_{h-1}}))].$$

We can conclude by the properties of the universal stochastic set cover algorithm that

$$\mathbf{E}[c(\mathbf{S}(\omega^k))] \leq 6\mathbf{E}[c(\mathbf{S}_{k_{h-1}}(\omega^{k_{h-1}}))] = O(\log mn)\mathbf{E}[c(\text{opt}(\omega^{k_{h-1}}))] = O(\log mn)\mathbf{E}[c(\text{opt}(\omega^k))]. \quad \blacksquare$$

E Proof of Lemma 6.1

Analogously to Section 3, we partition the pairs (f, S) computed by the algorithm in two subsets: The pairs computed in the first step of the while loop are of Type I, and the remaining pairs of Type II. The cost paid by the solution for a pair (f, S) is zero if no element in S is sampled, and otherwise is $c(f)$ plus the sum of the distances from the sampled elements in S to f . We next bound the cost of the pairs of Type I.

Lemma E.1 *The expected total cost of Type I pairs is $O(\log n \mathbf{E}[c(\text{opt})])$.*

Proof: Let (f_i, S_i) be the i -th pair of Type I selected by the algorithm, $i = 1, \dots, l$. Moreover, let C_i denote the set C before f_i was selected. The expected cost paid by our solution for buying f_i and connecting the clients in S_i to f_i is

$$c(f_i)Pr[S_i \cap \omega \neq \emptyset] + \sum_{v \in S_i} d(v, f_i)Pr[v \in \omega] \leq c(f_i) + \frac{k}{n} \sum_{v \in S_i} d(v, f_i).$$

Since, f_i and S_i are selected in this step, this quantity is at most $192\mathbf{E}[c(\text{opt})]|S_i \cap C_i|/|C_i|$. The lemma follows by the same argument as in Lemma 3.2. $\blacksquare$

In the following we denote by $(f_i, S_i) = (f_i, \{v_i\})$ the i -th pair of Type II selected by the algorithm, $i = 1, \dots, l$. We let C_i denote the set C before f_i was selected, $n_i = |C_i|$, and $k_i = n_i \frac{k}{n}$. We also let ω_i be the subsequence obtained from the random sequence ω by taking only elements belonging to C_i , and let $\text{opt}|_{\omega_i}$ be the set obtained from opt by taking for each $v \in \omega_i$ the facility f minimizing $c(f) + d(v, f)$. We define $c(\text{opt}|_{\omega_i})$ and $|\text{opt}|_{\omega_i}|$ in the usual way. Additionally, for each client $v \in C$, let D_v denote the connection cost paid by opt given that v appears in the random sequence ω and let $D := \sum_{v \in C} D_v$ denote the total expected connection cost paid by opt . Denote the set of ω 's such that $|\omega_i| = q$ as Ω_i^q . Moreover, let us denote by $\Omega_i^{\geq q}$ the sequences ω such that $|\omega_i| \geq q$. We focus on scenarios $\Omega_i^{\geq \frac{k_i}{2}}$.

The following lemma is analogous to Lemma 3.3 and can be proved in the same way.

Lemma E.2 *For every $i \in \{1, \dots, \ell\}$, if $k_i \geq 8 \log 2n$ then there exists $q \geq k_i/2$ such that*

$$Pr_{\omega \in \Omega_i^q} \left[c(\text{opt}|_{\omega_i}) \leq 16\mathbf{E}[c(\text{opt})] \text{ and } |\text{opt}|_{\omega_i}| \leq 16\mathbf{E}[|\text{opt}|] \right] \geq \frac{1}{4}.$$

Next lemma extends Lemma 3.4.

Lemma E.3 *For every $i \in \{1, \dots, l\}$, if $k_i \geq 8 \log 2n$ then $k_i \leq 128\mathbf{E}[|\text{opt}|_{\omega_i}|] \log n$.*

Proof: Suppose the lemma does not hold, i.e. $k_i > 128\mathbf{E}[|\text{opt}|_{\omega_i}|] \log n$ and apply the previous lemma in order to define q . We show a contradiction by providing a pair (f, S) which violates the condition in the second step of the while loop of Algorithm 3. Consider the n_i^q different outcomes for a sequence of q clients in C_i . Disconnect v to its closest facility in any scenario where $D_v \geq 4\mathbf{E}[D_v | \omega \in \Omega_i^q]$.

Observe that in many scenarios we will connect fewer than q elements. Remove all scenarios where we connect fewer than $\frac{q}{4}$ elements. At least $1 - \frac{1}{4} - \frac{1}{4} \geq \frac{1}{2}$ scenarios will satisfy the conditions of Lemma E.2 and additionally connect at least $\frac{q}{4}$ elements.

For each facility f , let S_f denote the set of clients v within distance at most $4\mathbf{E}[D_v | \omega \in \Omega_i^q]$ from f . Consider the collections, of at most $d = 16\mathbf{E}[|\text{opt}|]$ facilities with total weight of opening all the d facilities less or equal to $16\mathbf{E}[c(\text{opt})]$. Let A_j , $1 \leq j \leq p \leq (2n)^d$, be the j -th such collection, and let $\cup A_j$ be the union of the S_f over the facilities of the collection. We must have:

$$\sum_{j=1}^p |\cup A_j|^{\frac{1}{4}q} n_i^{\frac{3}{4}q} \geq \frac{1}{2} n_i^q.$$

As a consequence, there must be one A_j such that:

$$|\cup A_j| \geq \frac{n_i}{2^{\frac{4}{q}} (2n)^{\frac{4d}{q}}} \geq \frac{n_i}{2^{\frac{2}{\log n}} n^{\frac{1}{\log n}}} \geq \frac{n_i}{8},$$

due to the assumption that $q \geq \frac{k_i}{2} \geq 64\mathbf{E}[|\text{opt}|_{\omega_i}|] \log n \geq 4d \log n$. The cost of opening all the facilities A_j is at most $16\mathbf{E}[c(\text{opt})]$. Furthermore, we have

$$\sum_{f \in A_j} \sum_{v \in S_f} \frac{k}{n} d(v, f) \leq \sum_{f \in A_j} \sum_{v \in S_f} \frac{k}{n} 4\mathbf{E}[D_v | \omega \in \Omega_i^q] \leq 4\mathbf{E}[D | \omega \in \Omega_i^q] \leq 4\mathbf{E}[c(\text{opt}) | \omega \in \Omega_i^q] \leq 8\mathbf{E}[c(\text{opt})].$$

Hence, by an averaging argument, there must exist a facility f such that

$$\frac{c(f) + \sum_{v \in S_f} \frac{k}{n} d(v, f)}{|S_f|} \leq \frac{16\mathbf{E}[c(\text{opt})] + 8\mathbf{E}[c(\text{opt})]}{n_i/8} \leq 192\mathbf{E}[c(\text{opt})],$$

which contradicts the fact that a pair of Type II is selected in the iteration considered. $\blacksquare$

The following lemma is analogous to [Lemma 3.5](#)

Lemma E.4 For each $1 \leq i \leq l$,

$$c_i \mathbf{E}[|\text{opt}|_{\omega_{i+1}}|] \leq \mathbf{E}[c(\text{opt}|_{\omega_{i+1}})] \quad \text{and} \quad c_i(\mathbf{E}[|\text{opt}|_{\omega_i}|] - \mathbf{E}[|\text{opt}|_{\omega_{i+1}}|]) \leq \mathbf{E}[c(\text{opt}|_{\omega_i})] - \mathbf{E}[c(\text{opt}|_{\omega_{i+1}})]$$

Proof: For each facility in f used in $\text{opt}|_{\omega_{i+1}}$, $c(\text{opt}|_{\omega_{i+1}})$ is charged by at most $c(f)$ plus the shortest distance $d(v, f)$ between f and the closest sampled client $v \in C_{i+1}$. Trivially, $c_i \leq c_{i+1} \leq c(f) + d(v, f)$, and hence

$$c_i |\text{opt}|_{\omega_{i+1}}| \leq c_{i+1} |\text{opt}|_{\omega_{i+1}}| \leq c(\text{opt}|_{\omega_{i+1}}).$$

Taking expectations we obtain the first inequality.

Similarly, $|\text{opt}|_{\omega_i}| - |\text{opt}|_{\omega_{i+1}}|$ is the number of facilities $\mathcal{F}'$ covering the sampled elements C' in $C_i \setminus C_{i+1}$. For a given $f \in \mathcal{F}'$, consider the pair (f, S_f) , where S_f are the clients in C' covered by f . For each such pair $\text{opt}|_{\omega_i}$ pays at least c_i . Thus

$$c_i(|\text{opt}|_{\omega_i}| - |\text{opt}|_{\omega_{i+1}}|) \leq c(\text{opt}|_{\omega_i}) - c(\text{opt}|_{\omega_{i+1}}),$$

which implies the second inequality. $\blacksquare$

We now bound the cost of the Type II pairs in the solution constructed by the algorithm.

Lemma E.5 The expected cost of Type II pairs selected by [Algorithm 3](#) is $O(\log n) \cdot \mathbf{E}[c(\text{opt})]$.

Proof: Let $c_i := c(f_i) + d(v_i, f_i)$. Observe that $c_1 \leq \dots \leq c_l$. For notational convenience, we set $k_{l+1} = 0$ and $c_0 = 0$. Moreover, let j be such that $k_j \geq 8 \log 2n$ and $k_{j+1} < 8 \log 2n$. Note that in expectation we see at most $8 \log 2n$ elements in C_{j+1} . Each of these elements is connected in opt to a facility f for which $c(f) + d(v_i, f)$ is not smaller than c_i . Hence, the cost of connecting v_i 's to f_i 's for $j < i \leq l$ is bounded by $8 \log 2n \mathbf{E}[c(\text{opt})]$.

By [Lemma E.3](#) and [Lemma E.4](#), the cost of the pairs $(f_1, S_1), \dots, (f_j, S_j)$ is upper bounded by

$$\begin{aligned} \sum_{i=1}^j c_i \Pr[v_i \in \omega] &= \sum_{i=1}^j c_i (k_i - k_{i+1}) \leq \sum_{i=1}^j k_i (c_i - c_{i-1}) \leq \sum_{i=1}^j 128 \mathbf{E}[|\text{opt}|_{\omega_i}|] \log n (c_i - c_{i-1}) \\ &= 128 \log n \left(c_j \mathbf{E}[|\text{opt}|_{\omega_{j+1}}|] + \sum_{i=1}^j c_i (\mathbf{E}[|\text{opt}|_{\omega_i}|] - \mathbf{E}[|\text{opt}|_{\omega_{i+1}}|]) \right) \\ &\leq 128 \log n \left(\mathbf{E}[c(\text{opt}|_{\omega_{j+1}})] + \sum_{i=1}^j (\mathbf{E}[c(\text{opt}|_{\omega_i})] - \mathbf{E}[c(\text{opt}|_{\omega_{i+1}})]) \right) \\ &= 128 \log(n) \mathbf{E}[c(\text{opt}|_{\omega_1})] \leq 128 \log(n) \mathbf{E}[c(\text{opt})]. \end{aligned}$$

The lemma follows. $\blacksquare$

F Offline Algorithm for Universal Mappings

In this section, we give a $O(\log n)$ -approximation for the (unweighted) universal stochastic set cover problem. The mapping returned by the algorithm is the greedy mapping. We show that the expected cost of this mapping is within a $O(\log n)$ -factor of the optimal universal mapping.

Given any mapping π , We call a set S *large* wrt π if $|\pi^{-1}(S)| \geq \frac{n}{10k}$ else we call the set *small* wrt π . Let $\mathbb{L}_\pi$ and $\mathbb{S}_\pi$ denote the set of all large and small sets, respectively. Moreover, let T_π denote the set of elements mapped to large sets, i.e., $T_\pi = \{u : \pi(u) \in \mathbb{L}\}$. The expected cost of the the solution given by mapping π is

$$c(\pi) = \sum_{S \in \mathcal{C}} 1 - \left(1 - \frac{|\pi^{-1}(S)|}{n}\right)^k$$

We now use the fact that

$$\begin{aligned} \frac{1}{20} &\leq 1 - (1-x)^k \leq 1 & \text{if} & \quad kx \geq \frac{1}{10} \\ \frac{kx}{2} &\leq 1 - (1-x)^k \leq kx & \text{if} & \quad kx \leq \frac{1}{10} \end{aligned}$$

to obtain that

$$c(\pi) \geq \sum_{S \in \mathbb{L}_\pi} \frac{1}{20} + \sum_{S \in \mathbb{S}_\pi} \frac{k}{2n} |\pi^{-1}(S)| \geq \frac{|\mathbb{L}_\pi|}{20} + \frac{k}{2n} (n - |T_\pi|)$$

and

$$c(\pi) \leq |\mathbb{L}_\pi| + \frac{k}{n} (n - |T_\pi|)$$

Let ω denote the optimal mapping and σ denote the mapping returned by the greedy algorithm. Observe that the greedy algorithm uses at most $\mathbb{L}_\omega \log n$ sets to cover T_ω elements. We pay for these sets in all scenarios. Rest of the $n - |T_\omega|$ elements use at most one set each to be covered for a total expected cost of $\frac{k}{n}(n - |T_\omega|)$. Hence, the cost of the greedy solution is at most

$$\mathbb{L}_\omega \cdot \log n + \frac{k}{n} (n - |T_\omega|) \leq 20 \log n c(\omega)$$

as required.

Accepted Manuscript

On the Positive-Negative Partial Set Cover Problem

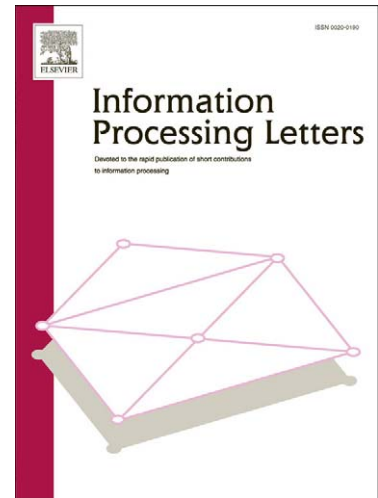
Pauli Miettinen

PII: S0020-0190(08)00159-2
DOI: [10.1016/j.ipl.2008.05.007](https://doi.org/10.1016/j.ipl.2008.05.007)
Reference: IPL 3885

To appear in: *Information Processing Letters*

Received date: 22 August 2007

Revised date: 17 April 2008



Please cite this article as: P. Miettinen, On the Positive-Negative Partial Set Cover Problem, *Information Processing Letters* (2008), doi: 10.1016/j.ipl.2008.05.007

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

On the Positive-Negative Partial Set Cover Problem

Pauli Miettinen

*Helsinki Institute for Information Technology, Department of Computer Science,
PO Box 68 (Gustaf Hållströmin katu 2b), FI-00014 University of Helsinki,
Finland*

Abstract

The Positive-Negative Partial Set Cover problem is introduced and its complexity, especially the hardness-of-approximation, is studied. The problem generalizes the Set Cover problem, and it naturally arises in certain data mining applications.

Key words: Approximation algorithms, Combinatorial problems, Hardness of approximation, Set cover

1 Introduction

The Positive-Negative Partial Set Cover ($\pm$ PSC) problem is a generalization of the Red-Blue Set Cover (RBSC) problem [2], which, for one, is a generalization of the classical Set Cover (SC) problem. The RBSC problem is, however, much harder than SC admitting the *strong inapproximability* property [6]. In this paper we will prove the strong inapproximability of $\pm$ PSC. The reductions

Email address: pamietti@cs.helsinki.fi (Pauli Miettinen).

used will also lead to an approximation algorithm for $\pm\text{PSC}$, and to results about its parameterized complexity.

1.1 Notation and Problem Definitions

In RBSC, we are given disjoint sets R and B of red and blue elements, respectively, and a collection $\mathcal{S} = \{S_1, \dots, S_n\} \subseteq 2^{R \cup B}$. The goal is to find a collection $\mathcal{C} \subseteq \mathcal{S}$ that covers all blue elements, i.e., $B \subseteq \cup \mathcal{C}$, while minimizing the number of covered red elements. The cost of a solution $\mathcal{C}$ is defined as $\text{cost}_{\text{RBSC}}(R, \mathcal{C}) = |R \cap (\cup \mathcal{C})|$, where $\cup \mathcal{C}$ is the union over $\mathcal{C}$'s sets (i.e., $\cup \mathcal{C} = \bigcup_{C \in \mathcal{C}} C$); a shorthand we use throughout this paper. We will use $\text{cost}_{\text{RBSC}}(\mathcal{C})$ when R is clear from the context. Finally, let $\rho = |R|$ and $\beta = |B|$.

In $\pm\text{PSC}$, the requirement of covering all blue elements is relaxed; instead, the goal is to find the best balance between covering the blue elements and not covering the red ones. In the context of $\pm\text{PSC}$, the red and blue elements are called *negative* and *positive* elements, respectively.

An instance of $\pm\text{PSC}$ is a triplet $(N, P, \mathcal{Q})$ with $|N| = \nu$, $|P| = \pi$, and $\mathcal{Q} = \{Q_1, \dots, Q_m\} \subseteq 2^{P \cup N}$. A solution of a $\pm\text{PSC}$ instance is a collection $\mathcal{D} \subseteq \mathcal{Q}$, and its cost is defined to be

$$\text{cost}_{\pm\text{PSC}}(N, P, \mathcal{D}) = |P \setminus (\cup \mathcal{D})| + |N \cap (\cup \mathcal{D})|, \quad (1)$$

namely the number of uncovered positive elements plus the number of covered negative elements. Again we will omit N and P when they are clear from the context.

27 1.2 Related Work

28 The RBSC problem was presented by Carr et al. [2] who also gave two hardness-
 29 of-approximation results to it: (i) unless $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$, there exist
 30 no polynomial-time approximation algorithms to approximate RBSC to within
 31 a factor of $2^{(4 \log n)^{1-\varepsilon}}$ for any $\varepsilon > 0$, and (ii) there are no polynomial-time
 32 approximation algorithms to approximate RBSC to within $2^{\log^1 - (\log \log \beta)^{-c}} \beta$ for
 33 any constant $c < 1/2$ unless $\text{P} = \text{NP}$. The first result was independently
 34 proved by Elkin and Peleg [4], and the latter result was based upon a result
 35 by Dinur and Safra [3]. The best upper bound for RBSC is due to Peleg [6],
 36 who recently presented a $2\sqrt{n \log \beta}$ -approximation algorithm for it.

37 To the best of the author's knowledge, there are no previous hardness results
 38 for the $\pm\text{PSC}$ problem, nor any approximation algorithms for it. The problem
 39 itself appears in some data mining applications (e.g., [1]), but its complexity
 40 and the existence of efficient approximation algorithms for it have not been
 41 studied previously.

42 2 Results

43 The main result of this paper relates the upper and lower bounds for the
 44 $\pm\text{PSC}$'s approximability to the respective bounds for RBSC.

45 **Theorem 1** *RBSC is approximable to within a factor of $f(\rho, \beta, n)$ if $\pm\text{PSC}$*
 46 *is approximable to within a factor of $f(\rho, \beta/\rho_{\max}, n)$, where $\rho_{\max}$ is the max-*
 47 *imum number of red elements in any set of the RBSC instance. Vice versa,*
 48 *$\pm\text{PSC}$ is approximable to within a factor of $g(\nu + \pi, \pi, m + \pi)$ if RBSC can be*

49 approximated to within a factor of $g(\nu, \pi, m)$.

50 Theorem 1 and the results from Section 1.2 provide the following corollaries.

51 **Corollary 2** *For any $\varepsilon > 0$, (i) there exists no polynomial-time approxima-*
 52 *tion algorithm for $\pm\text{PSC}$ with an approximation factor of $\Omega(2^{\log^{1-\varepsilon} m^4})$ unless*
 53 *$\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$, and (ii) there exists no polynomial-time approxi-*
 54 *mation algorithm for $\pm\text{PSC}$ with an approximation factor of $\Omega(2^{\log^{1-\varepsilon} \pi})$ unless*
 55 *$\text{P} = \text{NP}$.*

56 **Corollary 3** *There exists a polynomial-time approximation algorithm for $\pm\text{PSC}$*
 57 *that achieves an approximation factor of $2\sqrt{(m + \pi) \log \pi}$.*

58 The first part of Corollary 2 follows from the result by Carr et al. [2], and
 59 Corollary 3 follows from Peleg's algorithm [6]. The second part of Corollary 2
 60 follows from a result by Dinur and Safra [3] applied to RBSC: there exists
 61 an instance of RBSC where $\rho_{\max} = O\left(2^{\log^{1-(\log \log \beta)^{-c'}} \beta} (\log \log \beta)^{c'}\right)$ for some
 62 constant $c' < 1/2$, and unless $\text{P} = \text{NP}$ there are no polynomial-time approx-
 63 imation algorithms for it with an approximation factor of $2^{\log^{1-(\log \log \beta)^{-c}} \beta}$ for
 64 any constant $c < 1/2$. Thus, if we let $g_c(x) = 2^{\log^{1-(\log \log x)^{-c}} x}$ for all $c < 1/2$,
 65 then assuming that $\text{P} \neq \text{NP}$, there exists no polynomial-time approximation
 66 algorithm to $\pm\text{PSC}$ achieving an approximation factor of $g_c\left(\frac{\pi}{O(g_{c'}(\pi)(\log \log \pi)^{c'})}\right)$,
 67 which is $\Omega(2^{\log^{1-\varepsilon} \pi})$ for all $\varepsilon > 0$.

68 Theorem 1 is proved in the following two subsections, while Section 2.3 studies
 69 the parameterized complexity of $\pm\text{PSC}$. Notice that both RBSC and $\pm\text{PSC}$ have
 70 instances that have an optimal solution with zero cost. However, there are
 71 trivial polynomial-time algorithms to identify such instances and to find their
 72 optimal solutions. It is thus to be understood that henceforth all instances are

such that the cost of their optimal solution is at least 1.

2.1 From RBSC to $\pm$ PSC

Consider an instance of RBSC, i.e., a triplet (R, B, S) . We map this instance to an instance of $\pm$ PSC. Let the negative elements be exactly the red elements, $N = R$. For each blue element b_i , create $\rho_{\max} = \max_{S \in \mathcal{S}} |R \cap S|$ positive elements in P . Create the set collection $\mathcal{Q}$ so that all negative elements belong to the same subsets Q_j as their corresponding red elements, and all positive elements corresponding to a blue element b_i belong to the same subsets as b_i .

Let $\mathcal{D}$ be a solution of this instance of $\pm$ PSC. If $\mathcal{D}$ covers all positive elements, then the same subsets also cover all blue elements in RBSC, and $\mathcal{D}$ is a feasible solution of (R, B, S) . Moreover, $\text{cost}_{\pm\text{PSC}}(\mathcal{D}) = \text{cost}_{\text{RBSC}}(\mathcal{D})$, i.e., $\mathcal{D}$ induces same costs in both problems. If, on the other hand, there exists a positive element p not covered by $\mathcal{D}$, then there must be at least $\rho_{\max}$ positive elements not covered by $\mathcal{D}$. Thus we can add any set S with $p \in S$ to $\mathcal{D}$ without increasing the cost of the solution, as we cannot cover more than $\rho_{\max}$ negative elements with any S . If $\mathcal{C}$ is the (possibly extended) solution to RBSC induced by $\mathcal{D}$, we see that $\text{cost}_{\pm\text{PSC}}(\mathcal{D}) \geq \text{cost}_{\text{RBSC}}(\mathcal{C})$.

Finally, it is clear that the optimal solution of a $\pm$ PSC instance will cover exactly the negative elements corresponding to the red elements covered by the optimal solution of RBSC, i.e., the costs of the optimal solutions are equal. Denoting the optimal solutions to the instances of $\pm$ PSC and RBSC by $\mathcal{D}^*$ and $\mathcal{C}^*$, respectively, we see that $\frac{\text{cost}_{\pm\text{PSC}}(\mathcal{D})}{\text{cost}_{\pm\text{PSC}}(\mathcal{D}^*)} \geq \frac{\text{cost}_{\text{RBSC}}(\mathcal{C})}{\text{cost}_{\text{RBSC}}(\mathcal{C}^*)}$, and thus if we can approximate $\pm$ PSC to within a factor of $f(\rho, \beta/\rho_{\max}, n)$, then we can approx-

imate RBSC to within a factor of $f(\rho, \beta, n)$. This concludes the proof of the first part of Theorem 1.

2.2 From $\pm$ PSC to RBSC

Consider an instance of $\pm$ PSC: $(N, P, \mathcal{Q})$. For each $n_i \in N$, let there be a red element $r_i^- \in R$, and for each $p_i \in P$, let there be a blue element $b_i \in B$ and a red element $r_i^+ \in R$. For each set $Q_j \in \mathcal{Q}$, let there be a set $S_j^+ \in \mathcal{S}$ and for each positive element $p_i \in P$, let there be a set $S_i^- \in \mathcal{S}$. Define these sets as

$$S_j^+ = \{r_k^- \mid n_k \in Q_j\} \cup \{b_k \mid p_k \in Q_j\} \quad \text{and} \\ S_i^- = \{r_i^+, b_i\}.$$

Let $\mathcal{C}$ be a solution of the thus created RBSC instance. Create $\mathcal{D}$, a solution of the $\pm$ PSC instance, by adding each Q_j to $\mathcal{D}$ if the corresponding set S_j^+ is in $\mathcal{C}$.

To show that this reduction preserves the approximability, we start by considering the cost induced by $\mathcal{D}$. First, let n_k be a negative element in $\cup \mathcal{D}$. That is, there is a set Q_j so that $n_k \in Q_j$ and $Q_j \in \mathcal{D}$. But this means that the corresponding set S_j^+ must be in $\mathcal{C}$, and therefore the red element r_k^- corresponding to n_k is in $\cup \mathcal{C}$.

Second, let p_k be a positive element that is not in $\cup \mathcal{D}$, so none of the sets Q_j that contain p_k are in $\mathcal{D}$. This means that none of the sets S_j^+ that contain b_k are in $\mathcal{C}$. But as b_k must be covered by $\mathcal{C}$, it must be that S_k^- is in $\mathcal{C}$, and so r_k^+ is in $\cup \mathcal{C}$. Hence $\text{cost}_{\pm\text{PSC}}(\mathcal{D}) \leq \text{cost}_{\text{RBSC}}(\mathcal{C})$.

Consider then $\mathcal{D}^*$, the optimal solution of $(N, P, \mathcal{Q})$. We show that the cost

116 of the optimal solution of the RBSC instance created from the $\pm$ PSC instance
 117 is at most that of $\mathcal{D}^*$. Create $\mathcal{C}$ so that S_j^+ is in $\mathcal{C}$ if $Q_j \in \mathcal{D}^*$. For all blue
 118 elements b_i not yet covered by $\mathcal{C}$, add S_i^- in $\mathcal{C}$. It is straightforward to see that
 119 $\text{cost}_{\pm\text{PSC}}(\mathcal{D}^*) = \text{cost}_{\text{RBSC}}(\mathcal{C}) \geq \text{cost}_{\text{RBSC}}(\mathcal{C}^*)$. Therefore, $\frac{\text{cost}_{\text{RBSC}}(\mathcal{C})}{\text{cost}_{\text{RBSC}}(\mathcal{C}^*)} \geq \frac{\text{cost}_{\pm\text{PSC}}(\mathcal{D})}{\text{cost}_{\pm\text{PSC}}(\mathcal{D}^*)}$,
 120 so that if we can approximate RBSC to within a factor of $g(\nu, \pi, m)$, then we
 121 can approximate $\pm$ PSC to within a factor of $g(\nu + \pi, \pi, m + \pi)$.

122 2.3 Parameterized Complexity

123 We denote the parameterized versions of $\pm$ PSC and RBSC by p - $\pm$ PSC and
 124 p -RBSC. The parameter for both problems is the cost of the solution. The
 125 p -RBSC problem is W[2]-hard due to the results in [2] and [5].

126 In the reduction from RBSC to $\pm$ PSC (Section 2.1) the costs of the optimal
 127 solutions are equal, and in the reduction from $\pm$ PSC to RBSC (Section 2.2) the
 128 cost of the optimal solution to RBSC is at most the cost of the optimal solution
 129 to $\pm$ PSC. This proves that both reductions are indeed fpt-reductions [5], and
 130 gives rise to the following proposition.

131 **Proposition 4** *The p - $\pm$ PSC problem is equivalent to the p -RBSC problem un-*
 132 *der the fpt-reductions; especially, the p - $\pm$ PSC problem is W[2]-hard.*

133 3 Conclusions

134 This paper studied the $\pm$ PSC problem, proving both upper and lower bounds
 135 for its approximability. In addition to being important results as such, these
 136 bounds also provide new insights into the hardness of certain data mining

137 problems. Bounding the approximability of $\pm\text{PSC}$ (and RBSC) in terms of ν
138 (and ρ) remains an open problem.

139 Acknowledgements

140 The author thanks Heikki Mannila and Niina Haiminen for their comments.

141 References

- 142 [1] F. Afrati, A. Gionis, H. Mannila, Approximating a collection of frequent sets, in:
143 Proc. 10th KDD, 2004, pp. 12–19.
- 144 [2] R. D. Carr, S. Doddi, G. Konjevod, M. Marathe, On the red-blue set cover
145 problem, in: Proc. 11th ACM-SIAM SODA, 2000, pp. 345–353.
- 146 [3] I. Dinur, S. Safra, On the hardness of approximating label-cover, Inf. Process.
147 Lett. 89 (2004) 247–254.
- 148 [4] M. Elkin, D. Peleg, The hardness of approximating spanner problems, in: Proc.
149 17th STACS, vol. 1770 of LNCS, 2000, pp. 370–381.
- 150 [5] J. Flum, M. Grohe, Parameterized Complexity Theory, an EATCS Series,
151 Springer-Verlag, 2006.
- 152 [6] D. Peleg, Approximation algorithms for the label-cover_{MAX} and red-blue set
153 cover problems, J. Discrete Algorithms 5 (2007) 55–64.

A Functional Programming Approach to Distance-based Machine Learning

Darko Aleksovski¹, Martin Erwig², Sašo Džeroski¹

¹Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia

²School of EE and CS, Oregon State University, Corvallis, Oregon, USA

darko.aleksovski@gmail.com, Saso.Dzeroski@ijs.si

ABSTRACT

Distance-based algorithms for both clustering and prediction are popular within the machine learning community. These algorithms typically deal with attribute-value (single-table) data. The distance functions used are typically hard-coded.

We are concerned here with generic distance-based learning algorithms that work on arbitrary types of structured data. In our approach, distance functions are not hard-coded, but are rather first-class citizens that can be stored, retrieved and manipulated. In particular, we can assemble, on-the-fly, distance functions for complex structured data types from pre-existing components.

To implement the proposed approach, we use the strongly typed functional language Haskell. Haskell allows us to explicitly manipulate distance functions. We have produced a SW library/application with structured data types and distance functions and used it to evaluate the potential of Haskell as a basis for future work in the field of distance-based machine learning.

1. General Framework for Data Mining

A general framework for data mining should elegantly handle different types of data, different data mining tasks, and different types of patterns/models. Džeroski (2007) proposes such a framework, which explicitly considers different types of structured data and so-called generic learning algorithms that work on arbitrary types of structured data. The basic components of different types of such algorithms (such as distance or kernel-based ones) are discussed. Taking the inductive database (Imielinski and Mannila 1996) philosophy that proposes that patterns/models are first-class citizens that can be stored and manipulated, Džeroski proposes to store and manipulate basic components of data mining algorithms, such as distance functions.

Structured data

Complex data types are built from simpler types by using type constructors. To be more precise, we start with primitive data types, such as Boolean, Discrete(S) and Real. These serve as the basic building blocks for structured data types, composed by using type constructors. A minimal set of type constructors might be Set(), Tuple() and Sequence(): These take as arguments a data type: Set(T) is the type of sets of elements of type T.

Generic distance-based machine learning algorithms

Distance-based algorithms are popular within the machine learning community. They can be used for both clustering and prediction. Examples of such algorithms are hierarchical agglomerative clustering (HAC) and the (k)nearest neighbor algorithm (kNN) for prediction.

The above mentioned algorithms (HAC and kNN) are generic in the sense that they can work for arbitrary types of data, be it attribute-value (tuples of primitive data types) or structured data. We only need a distance function to be provided on the underlying data type. The distance function and the underlying data type are then parameters to the generic algorithm.

A distance function on type T is a function from pairs of objects of the type T to the set of non-negative reals $d :: T \times T \rightarrow \mathbb{R}^{0+}$. The three important properties this function has to satisfy are:

- 1) $d(x,y) \geq 0$
- 2) $d(x,y)=0$ iff $x=y$
- 3) $d(x,y)=d(y,x)$

A distance functions that besides these three satisfies the triangle property

- 4) $d(x,z) \leq d(x,y)+d(y,z)$

is called a metric.

In this work, we propose to use generic distance-based learning algorithms (GDBLA). These would be used in conjunction with a number of data types and corresponding distance functions from the domain of use, which can be passed as parameters to the GDBLAs. We propose to explicitly store and manipulate data types and distance functions for these. In particular, we propose to assemble distance functions for complex structured data types from pre-existing components.

2. Functional programming in Haskell

Since we are interested in storing, retrieving and manipulating distance functions, we consider the use of functional programming, i.e., Haskell (Thompson 1999).

Basics

There are a many features of functional programming and especially the language chosen (Haskell) that help users create succinct and easily understandable code. The code (as stated by people without extensive programming experience), is easily understandable, or at least the concepts, since they resemble the mathematical ones, are easy to grasp. Another desirable property of Haskell is its

expressiveness, which allows the programmer/user to spend more time on thinking and reasoning about the application domain in question, rather than trying to conform to the language's style of programming.

The key feature of functional programming languages, including Haskell, is the way of using functions and function compositions. Functions are first-class citizens and as such can be manipulated, passed as parameters, used as return values. Such functions are called higher-order functions.

In the context of our work, higher-order functions are clearly needed. We want to assemble a distance function for a complex data type (output), using distance functions on component simpler types (input). Here, functions are clearly present both as input and as output, and a higher-order function is needed to perform the assembly.

Haskell uses pure functions and nothing else. This means that a Haskell function resembles a mathematical function in the way that for every execution the same result is returned, that is no side-effects are allowed. The interpreters can (because of this lack of side-effects) more efficiently reorder executions. Moreover, some functional languages, such as Haskell, have adopted a lazy evaluation strategy, which supports infinite data structures and which can avoid unnecessary evaluations. This is desirable, since the user can define, for example, a sequence of infinite length and not worry about evaluation of unnecessary elements of the sequence, until they are needed in the program.

Strong Typing

Haskell is strongly typed. This means, e.g., that you can't freely use an Int instead of a Float, but rather have to explicitly convert the Int to a Float. Strong typing helps to find many programming errors. In particular, when combined with static typing, many programming errors can be caught before the program is run.

The type system of Haskell is polymorphic, allowing values of different data types to be handled using a uniform interface. A function that can be applied to values of different types is known as a polymorphic function. An example of a polymorphic data type is List (with elements of arbitrary type).

In our work, we make use of Haskell's fine grained set of types, both in terms of strong typing and polymorphism. These are very powerful features of Haskell. Types are automatically inferred wherever possible, which can help avoiding mistakes in code, and can help inferring the most general type for some polymorphic function.

3. The anatomy of distances for structured data

Distances on primitive data types

The currently considered list of primitive data types is Boolean, Discrete(S), and Real. We use the delta distance function on the first two, and absolute difference for Real. Delta yields zero given two identical inputs, one otherwise.

Distances on complex/structured types

Structured/complex data types are obtained through the (recursive) application of type constructors to simpler/primitive data types, with primitive types as base cases. The type constructors used here are Set(), Tuple() and Sequence(). Given distances on simpler (primitive) data types, we can compose distances for more complex structured types.

Distances on complex objects can be calculated through recursively inspecting the structure of the type. For this, we need (a) a function to generate pairs of objects of the simpler constitutive types, (b) distance functions on (objects of) the simpler types and (c) an aggregation function that we apply to the distance values obtained by applying (b) to the pairs produced by (a) to obtain a single (non-negative real) value of the distance between the complex objects.

In essence, the tree structure of the complex data type is inspected and for that type tree the following holds:

- every internal node represents a type constructor
- every leaf node is a primitive data type

Every internal node of this tree gets a pairing function and an aggregation function attached to it and every leaf node gets a distance function. The way of applying these functions to get a distance value (non-negative real) as a result is discussed next through an example.

For instance, given the Set(Char) type, and a distance function d() over the simple type Char, the distance of two sets of this type could be calculated as follows. If A and B are sets of this type, $A = \{a_i \mid i=1..n\}$ and $B = \{b_j \mid j=1..m\}$, a choice can be made whether $A \times B = \{(a_i, b_j) \mid i=1..n, j=1..m\}$ or just a subset thereof will be taken into account when determining the distance between the sets.

A function of the form

$$p :: [T] \rightarrow [T] \rightarrow [(T, T)]$$

can be used to determine the so-called important pairs of elements of the two complex objects, which will have the distance function d() applied to them. The function with this signature will be called a pairing function.

The second choice to be made is about the function that takes the computed distances between the pairs and produces a non-negative real, which is to be the distance between sets A and B. So, an additional function, called the aggregation function is to be defined, with the signature

$$\text{agg} :: [\text{Float}] \rightarrow \text{Float}$$

The third and last choice to make concerning the distance calculation on this complex type is which distance measure on our simple type Char to use. If we consider Char as a discrete type, the delta() function is the obvious choice. If we consider Char as an ordinal type (which we haven't discussed here), an absolute difference function which compares the two Chars after converting them to numbers (according to some character conversion table) may be used.

Pairing functions

The pairing functions are of more importance to the complex types using the Set() and Sequence() constructors. For the Tuple() constructor, given that it's heterogeneous, the pairing functions given below are most often in use.

```

p2 (Tuple2 a1 b1, Tuple2 a2 b2) =
[(a1,a2), (b1,b2)]

p3 (Tuple3 a1 b1 c1, Tuple3 a2 b2 c2) =
[(a1,a2), (b1,b2), (c1,c2)]

p4 (Tuple4 a1 b1 c1 d1, Tuple4 a2 b2 c2
d2) = [(a1,a2), (b1,b2), (c1,c2), (d1,d2)]

```

In the case of the Set(T) type constructor, a number of pairing functions can be used (Kalousis et al. 2006):

- all-to-all - every element from the first set is paired with every element from the second one
 - minimum distance - an element from one set is paired with the closest element of the other set
 - surjection pairing - considering all the surjections that map the larger set to the smaller, the "minimal" (with the distance Δ on T) such surjection is used
- $$d_s(S_1, S_2) = \min_{\eta} \sum_{(e_1, e_2) \in \eta} \Delta(e_1, e_2)$$
- linking - a mapping of one set to the other, all elements of each set participate in at least one pair
 - matching - each element of the two sets associated with at most one element of the other set

Aggregation functions

An aggregation function has the signature:
type Agg = [Float] -> Float

Examples of functions that can be used are:

- square-root of the sum of squares (Euclidian distance)
sqrt (sum [x*x|x<-xl])
- plain sum
sum [x|x<-xl]
- minimum (or maximum)
- median

The first three functions give equal weight to all the distances that they aggregate, while the last three only take into account one (or sometimes two, in the case of median) values. All of the above are special cases of the so-called ordered weighted aggregation functions (OWA, Yager and Kacprzyk 1997), which first sort the values to be aggregated, then apply a set of weights before aggregating. Assuming the list is sorted in ascending order, minimum gives a weight of one to the first element, maximum to the last, and median to the middle element (or weights of 1/2 to the two middle elements if the number of elements is even).

Haskell makes available an interesting and powerful feature when implementing the above. If the following piece of Haskell code is explored, taking into account the definition of the Agg type signature:

```

wSum :: [Float] -> Agg
wSum weights elems =
sum [a*b | (a,b) <- zip weights elems ]

```

it can be concluded that this aggregation function, weighted sum aggregation, has an additional parameter - an array of real values, that is weights. Haskell in this case allows the user to evaluate and use throughout the code the construct wSum weights, which is a specific aggregation function obtained through partial evaluation of the wSum function: the evaluation is partial as not all parameters are provided, in

particular the values to be aggregated. For this function definition to work properly it is required that the list weights is at least as long as the list elems.

4. A small database of distance function components

Populating the individual aspects

We have implemented a small database DDTD (database of data types and distances) where the definitions of data types and their corresponding distance functions are stored. We start with the primitive data types mentioned above and the basic distance functions on these. We also store additional distance functions on the primitive data types, as well as aggregation and pairing functions.

We have also implemented a generic version of the kNN algorithm, for demonstration purposes as well as for testing the Haskell implementation of the concepts discussed above (structured data and distances thereon). Datasets conforming to type definitions stored in the DDTD can be loaded from a database or from an XML file. This allows us to experiment with machine learning algorithms that work with structured data.

The process of populating DDTD with data type definitions, distance definitions, additional aggregation or pairing functions can be carried out either from the command line, or from a graphical interface (currently supporting a subset of the actions listed below). We can

- create definitions of new data types (composing complex data types out of simpler ones)
- create a definition of a distance over some data type (either using built-in functions or additional custom aggregation, pairing and distance functions)
- add a new distance function on a primitive type
- add a new aggregation function
- add a new pairing function

The data types can be described using XML or Haskell code. The additional functions have to be in Haskell syntax. The reason behind using the Haskell syntax is that it provides extensive support for mathematical functions (mostly defined in its Prelude), as well as support for processing lists, which can be easily learned, grasped and reused.

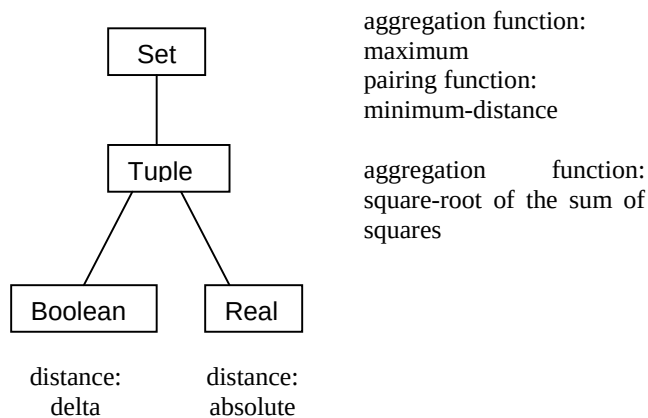
For every function to be added into the system, its signature (expected input data) has to be defined first, since some functions could use additional parameters (as was the case with the wSum aggregation function described in section 3 of this text). The definitions of the new functions have to be first checked for errors and then, if they produce the expected results, will be imported into the database, for further use. The possibility for additional functions and custom data types greatly increases the potential for use of DDTD.

DDTD usage scenario

Let us take for example the data type
t :: Set (Tuple2 Bool Float)

Note that this is the true Haskell definition of the type: Here we use Tuple2 instead of Tuple, as Tuple is really

a class of type constructors of varying arity (Tuple1, Tuple2, ...), rather than a type constructor. Two type constructors are used (Set and Tuple) and two primitive types (Bool and Real) in the above definition.



Picture 1. A custom data type with a distance defined for it

Using DDTD this custom data type is first declared. Then using a plain XML editor, the dataset that is to consist of this kind of objects is defined or imported from a database (another possibility could be to load it directly from a file). Once we have defined a data type, we can define a distance function on this data type (covered in the next section): As soon as this is done, distances on selected pairs of objects are calculated, for the purpose of checking if the results are as expected (in terms of types). Finally, a machine learning algorithm (like the implementation of k-NN mentioned above) can be invoked on the dataset.

Custom creating distance functions

For the data type in our example, a distance function could be defined in the following way (see Figure 1):

- For the Set type constructor, the aggregation function maximum and the pairing function minimum-distance are used
- For the Tuple type constructor, the aggregation function square-root of the sum of squares and the default pairing function are used
- For the Bool primitive type the distance-delta is used
- For the Real primitive type the distance-absolute is used

This distance function definition is converted into an appropriate XML code and stored in database, for later use.

If the functions supported by DDTD by default are not suitable for our complex distance definition, additional functions can be added. For instance, if an additional aggregation function is needed, for, say the median of a list of real values, the following should be carried out. Since the new function is going to be an aggregation function, the signature for aggregation function should be:

```
type Agg = [Float] -> Float
```

Finally, the body of the function should be written:

```
median xl = s!!(length `div` 2)
  where s = Data.List.sort xl
```

This definition will be checked for errors and then imported into the database and can later be used accordingly.

5. Conclusions and related work

We have been here concerned with distance-based machine learning, and in particular in such approaches that can handle arbitrary types of structured data. We have followed the approach proposed by Dzeroski (2007) to develop generic algorithms (in our case kNN), complemented with a database of definitions of data types and distance functions on these types. Moreover, the database contains basic building blocks for constructing distance functions on structured data and allows the user to custom create new ones, as well as choose from existing distance functions. To implement this, we have chosen a functional programming approach, which supports the higher-order nature of the operations that manipulate functions necessary for this.

Our work is related to inductive databases (IDBs, Imielinski and Mannila 1996): IDBs store patterns (and models) in addition to data. Most of the work in this area has focused on storing (and querying) local (frequent) patterns expressed in logical form. Our DDTD can be viewed as an inductive database storing global predictive models: the combination of a dataset, a distance function and a generic algorithm (such as kNN) yields a predictive model.

Allison (2004) also considers a functional programming approach to machine learning. He uses functional programming to define data types and type classes for models (where models include probability distributions, mixture models and decision trees) that allow for models to be manipulated in a precise and flexible way. However, he does not consider distance-based learning.

Finally, we consider the work on modular domain-specific languages and tools (Hudak 1998) relevant to our approach, especially for further work. Namely, we believe our approach can be extended to arrive at domain-specific languages for data mining. These might be coupled with domain-specific languages in a specific area of interest, e.g., a multi-media language. We believe that this would greatly facilitate the development of domain-specific data mining approached and their practical applications.

Acknowledgement. This work was supported by the EU funded project IQ (Inductive Queries for Mining Patterns and Models).

References

- [1] Allison, L.: Models for machine learning and data mining in functional programming. *Journal of Functional Programming* 15: 15–32, 2004.
- [2] Dzeroski, S: Towards a General Framework for Data Mining. In S. Dzeroski, J. Struyf (eds.) *Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Revised Selected and Invited Papers*, pp. 259-300. Springer, Berlin, 2007.
- [3] Hudak, P: Modular domain specific languages and tools. In *Proceedings of Fifth International Conference on Software Reuse*, pp. 134-142. IEEE Computer Society Press, 1998.
- [4] Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. *Comm. of the ACM* 39: 58–64, 1996.
- [5] Kalousis, A., Woznica, A., Hilario, M.: A unifying framework for relational distance-based learning founded on relational algebra. Technical Report, Computer Science Dept. Univ. of Geneva (2006)
- [6] Thompson, S.: *Haskell: The Craft of Functional Programming*. Add. Wesley (1999)
- [7] Yager, R, Kacprzyk, J: *The Ordered Weighted Averaging Operators: Theory and Applications*. Springer, Berlin, 1997.

An Inductive Query Language Proposal

(Extended Abstract)

IQ Confidential

Luc De Raedt and
Hendrik Blockeel and Heikki Mannila

Department of Computer Science
Katholieke Universiteit Leuven

Abstract. Some ideas for an inductive query language are formulated. The key idea is that joint probability distributions can be used in a similar way as tables in a relational database. This idea is then combined with the idea of mining views to yield a flexible query language.

1 Introduction

In the quest for an inductive query language, it has been argued that we need:

- to support the mining and manipulation of local patterns and global models, to treat patterns and models as “first class citizens”,
- to have the *closure* property, which states that the result of any query, whether inductive or deductive, yields (part of) an inductive database that can be used in further queries,
- to make abstraction of specific representations of patterns and models; for instance, it should not matter for querying whether we use a Markov net or a Bayesian net because both define a probabilistic model (a joint probability distribution); this calls for a separation of semantics and syntax; semantics should be covered by the language while the syntactic details should be hidden in the implementation; for the logical case, this is akin to the idea of the “mining views”,
- to support both logical (symbolic) and probabilistic reasoning.

If one accepts these points, it seems clear that we need a uniform treatment of patterns, models and data.

The key contribution of this note is that we argue that a joint probability distribution can be used for these purposes. Although this idea is unsurprising, it provides us with a surprisingly flexible mechanism for realizing data mining. Furthermore, the key operations on joint probability distributions make a lot of sense from both a data base and a data mining perspective.

2 Distributions as Tables

A DAT (distribution as table) is a multi-set $M = \{(I, p_I) | I \subseteq \mathcal{I} \text{ is a traditional item-set and } p \in]0, 1]\}$ such that $\sum_{I \in M} p_I = 1$.

So, a DAT specifies a joint probability distribution, that is, a set of possible worlds, where each possible world is an item-set, transaction or a (propositional) logical interpretation. In this preliminary note we work with boolean attributes and logic, though it seems easy to extend the ideas towards the standard relational case.

A DAT thus specifies a joint probability distribution $P(\mathcal{I})$. There are two different dimensions along which a DAT can be interpreted. First, DATs can be represented *extensionally*, that is, as explicit lists of possible worlds or item-sets together with their probability, or *intentionally*, that is, by using a model, such as a Markov or Bayesian net. Second, DATs can be viewed from a *probabilistic* perspective, which takes into account the probabilities, or from a purely *logical* perspective, which ignores the probabilities of the item-sets. The logical or deterministic interpretation of a DAT is as the set of item-sets with non-zero probability. This can be represented intentionally by a logic formula that has exactly these item-sets as models.

This resulting possibilities are illustrated in Table 1. Notice that so far we

	extension	intention
logical	set of models	logical formula
probabilistic	set of possible worlds	joint probability distribution model

Table 1. The four dimensions of DATs

have made abstraction of the syntax of the representations used and only provided a semantic definition. The reason is that when combining data mining results it will be necessary to work with different representations of models and patterns and hence a general framework should work at the semantic level. Even though in practice a Markov Net might be used to store a specific DAT, it can still be queried as if it were a list of possible worlds. This is in line with the “mining views”.

The resulting dimensions are also instructive, because they allow us to switch between different points of view and to transform a DAT across the dimensions. Indeed, at least the following transformations exist:

Intentional to extensional . Given an extensional DAT (whether probabilistic or logical), one can enumerate all the possible worlds (for the probabilistic case together with their corresponding probability values). This operation is called *enumeration*. A variant of this operation is *sampling*; it would only generate a representative collection of possible worlds at random.

Probabilistic to deterministic As already indicated above, a probabilistic DAT can be projected on a deterministic one by selecting all possible worlds with non-zero probability.

Deterministic to probabilistic A multi-set of item-sets can be interpreted as a set of possible world in which the probabilities are determined as the relative frequencies with which the item-sets occur in the data.

Extensional to intentional Going from an extensional representation of a data set to an intentional one is what data mining algorithms and approaches do. Therefore, transformations from an extensional to an intentional representation correspond to data mining operations. We discuss these in detail in the Section 4.

It turns out that the key transformations correspond in a natural way to useful operations from a data mining perspective. Before discussing these in depth, we first sketch the (deductive) queries that can be posed to a DAT.

3 Standard Queries on DATs

Because in the probabilistic case, a DAT is nothing else than a joint probability distribution, any query that can be posed to a joint probability distribution can be posed to a DAT. This will typically be queries of the form $P(J|K)$ where J and K are assignments of truth-values to the items in J and K .

In a similar manner, for the deterministic case, because a deterministic DAT is nothing else than a logical theory (a set of models), any query that can be posed to logical theory can be posed to a DAT. This includes queries for satisfiability and entailment.

Standard operations on joint probability distributions can be used as well on DATs. This includes marginalization (realizing projection) and conditioning (selection). The same holds for the logical counterparts.

Finally, operations on multiple DATs are also natural. One can for instance employ define a mixture model $\text{DAT3} = w_1 \times \text{DAT1} + w_2 \times \text{DAT2}$ where DAT1 and DAT2 are DATs over the same set of item-sets and $w_1 + w_2 = 1$. For the deterministic case, this operations are simpler and correspond to simple set operations (such as union).

4 Partial DATs

Because many data mining and machine learning techniques often specify a set of syntactic or semantic constraints over the models, patterns, or theories to be mined, it is useful to make also abstraction of specific classes of DATs. To realize this, we introduce the notion of a *partial DAT* (pDAT). A partial DAT is a set of DATs over the same set of items $\mathcal{I}$. Partial DATs can specify

Model classes or hypotheses languages , for instance, that the DAT can be represented as a fully connected Bayesian network or as a k-CNF theory,

Semantic constraints , for instance, that the DAT satisfies a set of conditional independence assumptions (such as the structure of a Bayesian net being given),

Syntactic constraints , for instance, that the DAT must correspond to a set of Horn-clauses.

As for normal DATs, pDATs can be specified along the two dimensions sketched (extensional or intentional, and probabilistic or deterministic). Ideally, intentional pDATs will be specified using a kind of (probabilistic) logic because this allows one to reason about pDATs and their instances. As one example of such a logic, consider a set of conditional independence assumptions. As another example of such a logic, consider Nilsson's probabilistic logic. Nilsson's probabilistic logic (AIJ 86) assigns probability values to logical statements. A probabilistic logical formula $p : f$ (where p denotes the probability) is then satisfied by a DAT M if and only if

$$\sum_{(i,p_i) \in M, i \models f} p_i = p$$

That is, the probability mass assigned by the DAT to models of the formula f should be equal to p . A DAT then satisfies a PSAT theory (a set of probabilistic logic formulas) if and only if satisfies all the formulas in the PSAT theory.

Nilssons' logic is useful because it allows us to model frequent pattern mining as deductive inference in our framework (cf. also Calders, PODS 2004, and below) and can be used to represent missing values in a data set.

To represent a traditional data-set consisting of item-sets we can use a deterministic DAT. Such a deterministic DAT can be represented in Nilssons' logic as a set of statements such as $1/n : x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4$ where the x_i are the items, and n is the number of transactions in our dataset. Modeling that the value for, say x_2 , is unknown can now be realized as $1/n : x_1 \wedge \neg x_3 \wedge x_4$. It specifies the constraint that the sum of probabilities of the two item-sets that satisfy this conjunction must be $1/n$.

5 Data Mining using DATs

Now that we have introduced all the ingredients of our framework, we can discuss how it can be used to support different types of inductive queries and data mining tasks.

Concept-learning Concept-learning (as formalized by Valiant [CACM 84]) is the task of finding a hypothesis H (a logical formula) that is consistent with a set of positive and negatives examples. In our framework, this corresponds to finding a consistent logical formula (an intentional deterministic DAT) consistent with the positive and negative examples. These example sets can be represented as two different (deterministic) DATs (over the same set of items), or else as one such DAT (with a special item denoting the class). A pDAT could then specify the bias, that is, the language constraints or even a prior over possible hypotheses (using Nilsson's logic by assigning a probability to each possible formula).

Local pattern mining A set of transactions can be conveniently regarded as a PSAT theory, where each transaction is a row in the table, and the probability reflects the relative frequency with which the item-set appears in the data set. Toon Calders (in his PODS 04 paper) shows that local pattern mining then corresponds to some kind of deduction. Indeed, an item-set is a conjunction, and provided that the DAT entails that the probability of the conjunction (in a kind of PSAT theory) is larger than the minimum (relative) frequency threshold, it is a solution to the frequent pattern mining problem. Thus frequent pattern mining is the problem of finding all such item-sets, realizing some kind of probabilistic entailment. Calders also introduced the inference problem, the so-called FREQSAT problem, which consists of deciding whether there exists a DAT that probabilistically entails a given set of item-sets (with probabilities).

Probabilistic Modelling This is the classical task of statistics. It is concerned with inferring an intentional (probabilistic) DAT starting from an extensional DAT. It applies to both generative and discriminative learning.

Clustering Clustering is the process whereby one starts from a DAT, adds a new item representing a latent variable. The task of (binary) hard clustering is then to find an assignment of truth-values to this item for all the examples that maximizes some kind of scoring function. Soft clustering is the variant of this setting, where each example is split into two examples (one for each truth-value of the new item) together with a probability value. In this way, the result is that each example is a conjunction + a probability value, and hence, the resulting pDAT can be interpreted as a pSAT theory. There is one subtlety though. If an example e is split into $p_1 : e(true)$ and $p_2 : e(false)$ such that $p_1 + p_2 = 1$ and this is aggregated.

6 Conclusions

The above arguments show that DATs and pDATs can be useful as a theoretical framework for inductive querying.¹ Nevertheless, there are still many more open questions, than answered ones. The most important ones are:

- Does the framework allow for theoretical analysis ? The view on concept-learning is in line with the well-known PAC-learning framework due to Valiant (CACM 84), which is widely accepted as the formal basis for statistical and computational learning theory. The above framework raises the question as to whether using DATs and pDATs one might also formulate a computational learning framework for probabilistic model learning.
- Can the notion of a pDAT be used to reason about constraints, optimized algorithms etc. ? pDATs specify constraints on the possible solutions of the data mining queries, and hence it would be interesting to reason about them, and possible to define an algebra or a logic for realizing this. Reasoning could be of interest for optimization and for combining different types of pDATs.

¹ Although some of the technical details, especially for local pattern mining, have been omitted.

- How to turn these ideas into a proper query language and implementation?
Here some ideas of the mining views are probably useful.