

ECML 2007 PKDD
WARSAW POLAND

THE 18TH EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
THE 11TH EUROPEAN CONFERENCE ON PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

INTERNATIONAL WORKSHOP ON
**CONSTRAINT-BASED MINING AND
LEARNING**
AT ECML/PKDD 2007

CMILE'07

September 21, 2007

Warsaw, Poland

Editors:

Siegfried Nijssen

Katholieke Universiteit Leuven

Luc De Raedt

Katholieke Universiteit Leuven

Preface

In many applications of machine learning and data mining the most interesting results are not obtained by a single run of a single algorithm. It is often necessary that a user be able to express constraints and preferences on the models and patterns that the algorithms have to output. This workshop intends to bring together researchers in data mining and machine learning that have an interest in mining and learning algorithms that explicitly offer users the possibility to express multiple types of constraints and preferences, and who believe that general ways to deal with —sometimes conflicting— constraints are necessary. Results of this kind have been obtained in constraint-based pattern mining algorithms, algorithms that learn decision trees under constraints and constraint-based clustering algorithms; also issues such as the language in which to express constraints, and approaches for dealing with multiple conflicting constraints, are important.

This workshop is the successor of the workshop on Knowledge Discovery in Inductive Databases (KDID). Constraint-based mining methods are a core component of Inductive Databases. The new name of the workshop reflects that we believe that constraints are not only important in data mining, but also in machine learning.

We decided to have four types of presentations. Besides invited presentations, we have presentations of abstracts and of full papers. In the abstract presentations work is presented that was previously published in other conferences, but which is of interest to the topic of the workshop. By including abstract presentations we hope to reduce the barrier for submitting high quality work to the workshop. In the paper presentations work is presented that was submitted to CMILE and reviewed by the program committee of the workshop. Depending on the enthusiasm of the program committee, these papers were accepted either as full presentations or as short presentations.

We look forward to an interesting workshop.

Leuven, August 2007

Siegfried Nijssen
Luc De Raedt

Workshop Organization

Workshop Chairs

Siegfried Nijssen (Katholieke Universiteit Leuven)

Luc De Raedt (Katholieke Universiteit Leuven)

ECML/PKDD Workshop Chair

Marzena Kryszkiewicz (Warsaw University of Technology)

Workshop Program Committee

Hiroki Arimura

Hendrik Blockeel

Francesco Bonchi

Jean-François Boulicaut

Toon Calders

Ian Davidson

Saso Dzeroski

Peter Flach

Minos Garofalakis

Thomas Gärtner

Fosca Giannotti

Bart Goethals

Jiawei Han

Tomer Hertz

Kristian Kersting

Ross D. King

Joost N. Kok

Stefan Kramer

Tilman Lange

Taneli Mielikäinen

Jan Struyf

Rosa Meo

Shinichi Morishita

Céline Robardet

Arno Siebes

Kiri Wagstaff

Takashi Washio

Philip S. Yu

Xifeng Yan

Mohammed Zaki

Table of Contents

I Full Presentations – Abstracts

Anytime Learning of Cost-sensitive Decision Trees	3
<i>Saher Esmeir and Shaul Markovitch</i>	
Constraint Based Hierarchical Clustering for Text Documents	4
<i>Korinna Bade</i>	
The Chosen Few: On Identifying Valuable Patterns	5
<i>Björn Bringmann and Albrecht Zimmermann</i>	

II Full Presentations – Papers

A Fast Algorithm for Mining Utility-Frequent Itemsets	9
<i>Vid Podpečan, Nada Lavrač and Igor Kononenko</i>	
Mining Views: Database Views for Data Mining	21
<i>Hendrik Blockeel, Toon Calders, Elisa Fromont, Bart Goethals and Adriana Prado</i>	

III Short Presentations – Papers

Onto4AR: a framework for mining association rules	37
<i>Cláudia Antunes</i>	
Iterative Constraints in Support Vector Classification with Uncertain Information	49
<i>Jianqiang Yang and Steve Gunn</i>	
Multitarget Polynomial Regression	61
<i>Aleksandar Pečkov, Sašo Džeroski and Ljupčo Todorovski</i>	
Author Index	73

Part I

Full Presentations – Abstracts

Anytime Learning of Cost-sensitive Decision Trees

Saher Esmeir and Shaul Markovitch

Computer Science Department, Technion-IIT, Israel

Abstract. Machine learning techniques are increasingly being used to produce a wide-range of classifiers for complex real-world applications that involve different constraints both on the resources allocated for the learning process and on the resources used by the induced model for future classification. As the complexity of these applications grows, the management of these resources becomes a challenging task. In this work we introduce ACT (Anytime Cost-sensitive Tree learner), a novel framework for operating in such environments. ACT is an anytime algorithm that allows trading computation time for lower classification costs. It builds a tree top-down and exploits additional time resources to obtain better estimations for the utility of the different candidate splits. Using sampling techniques ACT approximates for each candidate split the utility of the subtree under it and favors a split with the best evaluation. Due to its stochastic nature ACT is expected to be able to escape local minima, into which greedy methods may be trapped. ACT can be applied in two anytime setups: the contract setup where the allocation of resources is known in advance, and the interruptible setup where the algorithm might be queried for a solution at any point of time. Experiments with a variety of datasets were conducted to compare the performance of ACT to that of the state of the art decision tree learners. The results show that for most domains ACT produces significantly better trees. In the cost-insensitive setup, where test costs are ignored, ACT could produce smaller and more accurate trees. When test costs are involved, the ACT trees were significantly more efficient for classification. ACT is also shown to exhibit good anytime behavior with diminishing returns.

S. Esmeir and S. Markovitch. Anytime Learning of Decision Trees. *Journal of Machine Learning Research (JMLR)*, 8, 2007.

S. Esmeir and S. Markovitch. Occam's Razor Just Got Sharper. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, 2007.

S. Esmeir and S. Markovitch. Anytime Induction of Decision Trees: An Iterative Improvement Approach. In *Proceedings of The 21st National Conference on Artificial Intelligence (AAAI-2006)*, 2006.

S. Esmeir and S. Markovitch. When a Decision Tree Learner Has Plenty of Time. In *Proceedings of The 21st National Conference on Artificial Intelligence (AAAI-2006)*, 2006.

S. Esmeir and S. Markovitch. Lookahead-based Algorithms for Anytime Induction of Decision Trees. In *Proceedings of the 21st International Conference on Machine Learning (ICML-2004)*, 2004.

Constraint Based Hierarchical Clustering for Text Documents

Korinna Bade

University of Magdeburg, Germany

Abstract. This presentation deals with constraint based clustering in a hierarchical clustering scenario. The explicit goal is to derive a hierarchical structure of clusters that is constrained by a partially known hierarchy. In the presentation, related work in the field of constraint based clustering is analyzed, pointing out major drawbacks of the current approaches, especially when applied to a hierarchical scenario. Furthermore, special attention is drawn to problems concerning the clustering of text documents. Once problems are identified, two different types of approaches as well as their combination are presented that specifically target at finding a hierarchical cluster structure for text documents under constraints. Results of a first evaluation on a hierarchical dataset of text documents are shown, considering different aspects like unevenly distributed constraints. The presentation ends with proposals for future research in this area.

K. Bade and A. Nürnberger. Personalized hierarchical clustering. In: *Proceedings of the 2006 IEEE/WIC/ACM Int. Conference on Web Intelligence*, 2006.

The Chosen Few: On Identifying Valuable Patterns

Björn Bringmann and Albrecht Zimmermann

Katholieke Universiteit Leuven, Belgium

Abstract. Pattern search is one of the main topics in data mining. In the last years different types of pattern languages were developed in order to be able to deal with the ever new challenges of new and hopefully valuable representations for all kind of data. Most algorithms developed handle the usually computationally intensive task in a decent way enabling us to extract millions of patterns from even small datasets. These patterns are then presented to the user or they are used as features such that each instance of the original database can be converted into a binary vector, each bit encoding the presence or absence of the pattern.

Due to the fact that interestingness (i.e. constraint satisfaction) of patterns is evaluated for each pattern individually, the amount of patterns to be considered by a user is often too large. Furthermore, when presenting the binary vector data to a machine learning technique, an overabundance of features does not help in the learning task, possibly even “confusing” the algorithm, leading to overfitting.

The aim of our work is to select a small, human-interpretable subset containing little redundancy from a larger set of patterns while retaining as much information as possible encoded in the full set.

We define the information of a pattern set as the partition it induces on the database, with all instances sharing the exact same subset of patterns belonging uniquely to one block. Thus, information is obtained from the composition of all patterns, contrary to e.g. the notion of closed patterns. Closedness only considers patterns having different covers; thus two mutually exclusive patterns are closed but induce the same partition.

Since the high amount of patterns leads to an exponentially large search space of possible subsets, we present a heuristic technique to tackle the problem. The pattern set is processed in a given order and only patterns that effect a change to the current partition are selected. Besides arbitrary orderings, our rather general algorithm furthermore allows for different techniques for measuring the effect of the pattern, thus allowing intuitive descriptions of the selection step or even of properties for the final subset.

We give some rather intuitive examples for selection criteria and combine them with straight-forward ordering strategies. When evaluated on several UCI data sets the techniques reduce the set of closed patterns severely.

B. Bringmann and A. Zimmermann. The Chosen Few: On Identifying Valuable Patterns. In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*, 2007.

Part II

Full Presentations – Papers

A Fast Algorithm for Mining Utility-Frequent Itemsets

Vid Podpečan¹, Nada Lavrač^{1,2}, and Igor Kononenko³

¹ Jozef Stefan Institute, Ljubljana, Slovenia

² University of Nova Gorica, Nova Gorica, Slovenia

³ University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

vid.podpecan@gmail.com

nada.lavrac@ijs.si

igor.kononenko@fri.uni-lj.si

Abstract. Utility-based data mining is a new research area interested in all types of utility factors in data mining processes and targeted at incorporating utility considerations in both predictive and descriptive data mining tasks. High utility itemset mining is a research area of utility-based descriptive data mining, aimed at finding itemsets that contribute most to the total utility. A specialized form of high utility itemset mining is utility-frequent itemset mining, which – in addition to subjectively defined utility – also takes into account itemset frequencies. This paper presents a novel efficient algorithm FUFM (Fast Utility-Frequent Mining) which finds all utility-frequent itemsets within the given utility and support constraints threshold. It is faster and simpler than the original 2P-UF algorithm (2 Phase Utility-Frequent), as it is based on efficient methods for frequent itemset mining. Experimental evaluation on artificial datasets show that, in contrast with 2P-UF, our algorithm can also be applied to mine large databases.

1 Introduction

Utility-based data mining [10–12] is a broad topic that covers all aspects of economic utility in data mining. It encompasses predictive and descriptive methods for data mining, among the later especially detection of rare events of high utility (e.g. high utility patterns). This paper describes methods for itemset mining or more specifically, mining utility-frequent itemsets which is a special form of high utility itemset mining [13, 14].

Standard methods for association rule mining [1, 14] are based on support and confidence measures. The goal of the first phase of assoc. rule mining is to find all frequent itemsets and the goal of the second phase is to build rules based of frequent itemsets. We use support measure because we assume that the user is interested only in statistically important patterns.

However, frequency of an itemset alone does not assure its interestingness because it does not contain information on its subjectively defined utility such

as profit in euros or some other variety of utility. Mining high utility itemsets thus upgrades the standard frequent itemset mining framework as it employs subjectively defined utility instead of statistics-based support measure. User-defined utility is based on information not available in the transaction dataset. It often reflects user preference and can be represented by an external utility table or utility function. Utility table (or function) defines utilities of all items in a given database (we can also treat them as weights). Besides subjective external utility we also need transaction dependent internal utilities (e.g. quantities of items in transactions). Utility function we use to compute utility of an itemset takes into account both internal and external utility of all items in a itemset. The most usual form that is also used in this paper is defined as a sum of products of internal and external utilities of present items. The goal of high utility itemset mining is to find all itemsets that give utility greater or equal to the user specified threshold.

The deficiency of this approach is that it does not consider the statistical aspect of itemsets. Utility-based measures should incorporate user-defined utility as well as raw statistical aspects of data [14]. Consequently, it is meaningful to define a specialized form of high utility itemsets, utility-frequent itemsets [15] which are a subset of high utility itemsets as well as frequent itemsets. Example 1 indicates differences between frequent, high utility and utility-frequent itemsets.

Example 1. As an example let us analyze sales in a large retail store. We can find that itemset {bread, milk} is frequent, itemset {caviar, champagne} is of high utility and itemset {beer} is utility frequent. A smart manager should pay special attention to itemset {beer} as it is frequent and of high utility. On the other side, itemset {bread, milk} is frequent but not of high utility and itemset {caviar, champagne} gives high utility but is not frequent.

First algorithm 2P-UF for mining utility-frequent itemsets was introduced together with formal definition of this novel area [15] of utility-based itemset mining. It is based on a quasi support, a special measure that solves the problem of nonexistence of anti-monotone property of joined support-utility measure. 2P-UF is proven to find all utility-frequent itemsets but it has some properties that render it impossible to use in practice on large databases. In this paper we present a new algorithm that treats utility-frequent itemsets as a special form of frequent itemsets which is in contrast with 2P-UF algorithm since it treats them as a special form of high utility itemsets. Our approach proves to be efficient because support measure has anti-monotone property and assures efficient mining approach. Moreover, it is possible to use existent, very efficient methods for mining frequent itemsets, that can significantly speed up the mining process.

The remainder of this paper is organized as follows. In Section 2 we briefly survey work on frequent and high utility itemset mining as it forms a formal theoretical background for our algorithm. In Section 3 we describe our new algorithm and its advantages in comparison with 2P-UF algorithm. Section 4 describes and comments our implementation and results of experiments on synthetical

databases. Finally, conclusions are drawn in Section 5 where we also indicate possible directions of future work.

2 Mining High Utility Itemsets

A frequent itemset is a set of items that appears at least in a pre-specified number of transactions. Formally, let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items and $DB = \{T_1, T_2, \dots, T_n\}$ a set of transactions where every transaction is also a set of items (i.e. itemset). Given a minimum support threshold $minSup$ an itemset S is frequent iff:

$$\frac{|\{T | S \subseteq T, T \subseteq DB, S \subseteq I\}|}{|DB|} \geq minSup.$$

Frequent itemset mining is the first and the most time consuming step of mining association rules. During the search for frequent itemsets the anti-monotone property is used.

Definition 1. *Let D be the domain of a function f . f has the anti-monotone property when $\forall x, y \in D : x \leq y \Rightarrow f(y) \leq f(x)$.*

In the case of mining frequent itemsets the anti-monotone property assures that no superset of an infrequent itemset is frequent. Consequently, infrequent candidates can be discarded during the candidate generation phase. The first efficient frequent itemset mining algorithm APriori was developed by Agrawal et. al. [1], but later many faster methods were developed (FP-trees [5], ECLAT [16], Relim [2]). For the sake of simplicity, our implementation uses the APriori algorithm, however, any other more efficient algorithm could be used.

2.1 High Utility Itemsets

A high-utility itemset mining model was defined by Yao, Hamilton and Butz [13]. It is a generalization of the share-mining model [3, 4]. The goal of high utility itemset mining process is to find all itemsets that give utility greater or equal to the user specified threshold. The following is the set of definitions given in [13] which we shall illustrate on a small example.

Definition 2. *The external utility of an item i_p is a numerical value y_p defined by the user. It is transaction independent and reflects importance (usually profit) of the item. External utilities are stored in an utility table. For example, external utility of item B in Table 2 is 10.*

Definition 3. *The internal utility of an item i_p is a numerical value x_p which is transaction dependent. In most cases it is defined as the quantity of an item in transaction. For example, internal utility of item E in transaction T_5 is 2 (see Table 1).*

Table 1. Database with 10 transactions and 5 distinct items.

TID	A	B	C	D	E
1	0	0	18	0	1
2	0	6	0	1	1
3	2	0	1	0	1
4	1	0	0	1	1
5	0	0	4	0	2
6	1	1	0	0	0
7	0	10	0	1	1
8	3	0	25	3	1
9	1	1	0	0	0
10	0	6	2	0	2

Table 2. External utilities of items from database in Table 1.

item	A	B	C	D	E
profit (€)	3	10	1	6	5

Definition 4. Utility function f is a function of two variables:
 $f(x, y) : (\mathbf{R}^+, \mathbf{R}^+) \rightarrow \mathbf{R}^+$. The most common form also used in this paper is the product of internal and external utility: $x_p * y_p$.

Definition 5. The utility of item i_p in transaction T is the quantitative measure computed with utility function from Definition 4: $u(i_p, T) = f(x_p, y_p)$, $i_p \in T$. For example: utility of item E in transaction T_5 is $2 * 5 = 10$.

Definition 6. The utility of itemset S in transaction T is defined as $u(S, T) = \sum_{i_p \in S} u(i_p, T)$, $S \subseteq T$. For example: utility of itemset $\{B, E\}$ in transaction T_2 is $u(\{B, E\}, T_2) = u(\{B\}, T_2) + u(\{E\}, T_2) = 6 * 10 + 1 * 5 = 65$.

Definition 7. The utility of item i_p in itemset S is defined as $u(i_p, S) = \sum_{T \in DB, S \subseteq T} u(i_p, T)$. For example, utility of item E in itemset $\{B, E\}$ is $u(E, \{B, E\}) = u(E, T_2) + u(E, T_7) + u(E, T_{10}) = 20$.

Definition 8. The utility of itemset S in database DB is defined as $u(S) = \sum_{T \in DB, S \subseteq T} u(S, T) = \sum_{T \in DB, S \subseteq T} \sum_{i_p \in S} f(x_p, y_p)$. For example, utility of itemset $\{A, E\}$ in database from Table 1 is $u(\{A, E\}) = u(\{A, E\}, T_3) + u(\{A, E\}, T_4) + u(\{A, E\}, T_8) = 33$.

Definition 9. The utility of transaction T is defined as $u(T) = \sum_{i_p \in T} u(i_p, T)$. For example: utility of transaction T_{10} is $u(T_{10}) = u(\{B\}, T_{10}) + u(\{C\}, T_{10}) + u(\{E\}, T_{10}) = 72$.

Definition 10. *The utility of database DB is defined as $u(DB) = \sum_{T \in DB} u(T)$. For example, utility of database DB from Table 1 is $u(DB) = u(T_1) + \dots + u(T_{10}) = 23 + \dots + 72 = 400$.*

Definition 11. *The utility share of itemset S in database DB is defined as $U(S) = \frac{u(S)}{u(DB)}$. For example, utility share of itemset {A, D, E} in database from Table 1 is $U(\{A, D, E\}) = \frac{46}{400} = 0.115 = 11.5\%$.*

Finally, on the basis of definitions 2–11 we can formally define high utility itemset and the general problem of high utility itemset mining.

Definition 12. *Itemset S is of high utility iff $U(S) \geq \text{minUtil}$ where minUtil is user defined utility threshold in percents of the total utility of the database.*

Definition 13. *High utility itemset mining is the problem of finding set H defined as $H = \{S | S \subseteq I, U(S) \geq \text{minUtil}\}$ where I is the set of items (attributes).*

Utility function from Definition 4 is neither monotone or anti-monotone which can be proven with a counter example based on database from Table 1: $\{A, E\} \subseteq \{A, D, E\}$, $u(\{A, E\}) \leq u(\{A, D, E\})$ and $\{B\} \subseteq \{B, C\}$, $u(\{B\}) \geq u(\{B, C\})$.

Because of the nonexistence of anti-monotone property of utility function efficient high utility itemset mining algorithms [7, 8] employ critical function, a special function that estimates the utility of all possible supersets of a given itemset. It has the anti-monotone property which ensures the existence of a systematic non-exhaustive mining method. Critical function of itemset S is simply utility (see Definition 10) of database DB_S (a subset of DB with only those transactions that contain S). It is used to reduce the number of candidates by discarding low quality itemsets (all their supersets are of low utility) during the candidate generation phase of the high utility itemset mining process.

2.2 Utility-Frequent Itemsets

Utility-frequent itemsets are a special form of high utility itemsets, therefore, all quoted definitions also apply. For a given utility threshold μ each itemset S is associated with a set of transactions defined as

$\tau_{S,\mu} = \{T | S \subseteq T \wedge u(S, T) \geq \mu \wedge T \in DB\}$. On the basis of this set of transactions an extended support measure can be identified: $\text{support}(S, \mu) = \frac{|\tau_{S,\mu}|}{|DB|}$.

Definition 14. *Itemset S is utility-frequent if for a given utility threshold μ and support threshold s the extended support measure $\text{support}(S, \mu)$ is greater or equal to s .*

The measure of extended support is obviously not anti-monotone as it is based on a non-monotone utility function (see Definition 4 and counter example). It is

possible to use critical function and algorithms for high utility itemset mining (i.e. DCG [7], ShFSM [8]), however, they are highly impractical and inefficient since the utility threshold is defined at transaction level instead of database level (see definition of the set $\tau_{S,\mu}$) and thus much smaller.

Because the utility threshold has direct influence on the number of candidates in DCG and ShFSM algorithm, they generate many high utility candidates (with respect to a given threshold). However, only a small fraction of them are also utility frequent. To tackle this problem, authors of the utility-frequent itemset mining model defined *quasi support*, the basis of their 2P-UF algorithm [15].

Definition 15. *Quasi support is a special form of extended support defined as $quasiSupport(S, \mu) = \frac{|\tau'_{S,\mu}|}{|DB|}$ where $\tau'_{S,\mu}$ is a set of transactions $\tau'_{S,\mu} = \{T | u(S, T) \geq \mu \wedge T \in DB\}$.*

We should remind the reader that it is not necessary for an itemset S to be a true subset of transaction T when computing its quasi support. It can be only partially contained as long as it gives desired minimum utility. Quasi support has the monotone property which ensures the existence of systematic non-exhaustive mining method. However, in comparison with the use of an anti-monotone function the process runs in the opposite direction.

Proof (monotonicity of the quasi support measure). Let X and Y be subsets of the set I (set of all items in a database) and let $X \subseteq Y$ and let X be utility-frequent. Obviously, for each transaction $T \subseteq \tau'_{X,\mu}$ holds $u(Y, T) \geq u(X, T)$ because X is a subset of Y. Thus, $|\tau'_{Y,\mu}| \geq |\tau'_{X,\mu}|$, $quasiSupport(Y, \mu) \geq quasiSupport(X, \mu)$ and Y is utility frequent. \square

2P-UF algorithm is based on the fact that every utility-frequent itemset is also quasi utility-frequent. Figure 1 briefly describes the 2P-UF algorithm. In the first phase of the algorithm all quasi frequent itemsets are collected and in the second phase all quasi utility-frequent but not utility-frequent are discarded. Function $QUF - APriori(\cdot, \cdot, \cdot)$ starts with itemsets of length $n - 1$ (n is the number of items in DB). It computes intersection of each itemset with all other itemsets. Candidates of length $n - 2$ which do not have quasi utility-infrequent supersets (monotone property) and satisfy the given utility threshold are appended to the set of quasi utility-frequent itemsets and used in the next iteration to find all quasi utility-frequent itemsets of length $n - 3$. The process repeats until candidates of length 1 are generated and checked or the new set of candidates is empty and no shorter candidates can be produced. Further details of the 2P-UF algorithm and detailed description of the function $QUF - APriori(\cdot, \cdot, \cdot)$ can be found in [15].

Mining of high utility and utility-frequent itemsets should be organized similar to mining of frequent itemsets. We start with conservative (high) thresholds and lower them as long as we are not contended with the number of itemsets found.

```

Algorithm 2P-UF
Input:
- database DB
- constraints minUtil and minSup
Output:
- all utility-frequent itemsets

/* Phase 1: find all quasi utility-frequent itemsets */
[1] CandidateSet = QUF-APriori(DB, minUtil, minSup)
/* Phase 2: prune utility-infrequent itemsets */
[2] foreach c in CandidateSet:
[3]   foreach T in DB:
[4]     if c in T and u(c,T) >= minUtil:
[5]       c.count += 1
[6] return {c in CandidateSet | c.count >= minSup}

```

Fig. 1. Pseudo code of the 2P-UF algorithm.

3 A Fast Algorithm for Mining Utility-Frequent Itemsets

2P-UF utility-frequent itemset mining algorithm described in Section 2 is proven to find all utility-frequent itemsets. However, due to the monotone property of quasi support measure it has a few disadvantages which render it unusable for mining of large datasets.

The first weak point is the reversed way of candidate generation. 2P-UF algorithm wastes time checking long itemsets that are highly unusual to be utility-frequent. For example, when mining a database with 1000 distinct items (attributes) 2P-UF algorithm first generates and checks all itemsets of length 999, then itemsets of length 998 etc. Short itemsets which have fairly large probability to be utility-frequent, come at the very end.

Candidate generation function is also slow and inefficient as it computes intersection of every pair of candidates in each iteration. Moreover, computation of quasi support measure is also inefficient because special data structures (hash trees) can not be used and we have to scan database once for every candidate.

Finally, the two-phase form of the algorithm is space consuming since we have to store all quasi utility-frequent candidates from the first phase to filter them in the second phase. It is possible to avoid this waste of space by merging both phases and filter non utility-frequent candidates in every iteration of the algorithm.

It is obvious that 2P-UF algorithm can not be used in practice where databases usually consist of millions of transactions and thousands of items. Our new algorithm [9] FUFM (**F**ast **U**tility-**F**requent **M**ining) is based on the fact that utility-frequent itemsets are a special form of frequent itemsets. Moreover, the support measure is always greater or equal to the extended support measure. Proof is trivial because when computing extended support we count only those transaction containing given itemset S that also gives minimum utility on S ,

but when computing "ordinary" support we count all transactions containing S . The practical consequence of this statement is that frequent itemset mining algorithms can be used to mine utility-frequent itemsets. These algorithms are well studied and also very efficient.

For this reason, our FUFM algorithm is very simple and fast because the main part is the "external" frequent itemset mining algorithm. It is straightforward to find utility-frequent itemsets among frequent itemsets because all that is needed is to build a hash tree. This data structure [1] is used to compute supersets (i.e. transactions) for all candidates and, with this information, utilities of all candidates. Figure 2 shows pseudo code of our FUFM algorithm.

Algorithm FUFM

Input:

- database DB
- constraints minUtil and minSup

Output:

- all utility-frequent itemsets

- [1] $L = 1$
- [2] find the set of candidates of length L with support $\geq \text{minSup}$
- [3] compute extended support for all candidates and output utility-frequent itemsets
- [4] $L += 1$
- [5] use the frequent itemset mining algorithm to obtain new set of frequent candidates of length L from the old set of frequent candidates
- [6] stop if the new set is empty otherwise go to [3]

Fig. 2. Pseudo code of the FUFM algorithm.

Clearly, FUFM algorithm does not have disadvantages and inefficiencies of the 2P-UF algorithm as its generation phase (step 5 on Fig. 2) is based on frequent itemset mining methods. Filtering non-utility frequent candidates is also efficient because we only need to build a hash tree from candidates and push all transactions down the tree to compute subsets. Consequently, time and space complexity are both fully determined with the complexity of the frequent itemsets mining method used.

Comparison of the number of candidates from consequent iterations of 2P-UF and FUFM algorithms is not trivial, but intuitively we can conclude that in first iterations of the 2P-UF algorithm there are lots of candidates since quasi support measure overestimates longer itemsets. In fact, 2P-UF algorithm is efficient only in case when utility threshold is very high and result is an empty set. In such case the mining process stops at the very first iterations. Because in our new algorithm utility threshold does not have influence on the candidate generation

phase, FUFM performs worse in this special case as it has to inspect all frequent itemsets regardless of their utility.

Let us compare the sets of generated candidates with 2P-UF and FUFM algorithm on a small database with 200 transactions. Utility threshold was set to 17.94 € (0.5% of the total utility of the database) and support threshold was fixed on 10%. Both algorithms found all 21 utility-frequent itemsets, FUFM in 8.5 seconds and 2P-UF in 1567.8 seconds. Number of remaining candidates (after pruning low support / non quasi utility-frequent candidates from all generated candidates) is represented with a graph on Fig. 3.

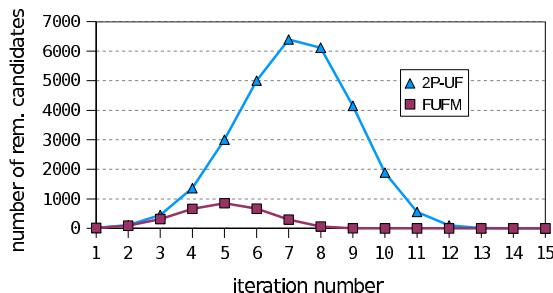


Fig. 3. Number of remaining candidates using 2P-UF and FUFM.

Let us point out again that 2P-UF algorithm generates utility-frequent itemsets in reverse order. Therefore, iteration i of the FUFM algorithm structurally equals iteration $15 - i$ of the 2P-UF algorithm.

4 Experiments

Both algorithms were implemented in the Python programming language within the Orange [17] data mining framework. Our choice of interpreted programming language results in some severe constraints concerning the size of used databases. However, our implementation is for testing purposes only and not for use in practice. All experiments were performed on a PC with AMD 3000+ processor, 512MB of main memory, version 2.5 of Python interpreter and Orange 0.99b (17th may 2007).

We used IBM Quest synthetic data generator [1, 6]. It is highly advanced and considers typical properties of real transactional databases such as high frequencies of some itemsets, mean length of transactions, etc. This generator can produce only a binary form of transactional databases. Therefore, internal and external utilities were generated separately from the log-normal distribution in range $[1, \dots, 10]$ (internal) and $[1, \dots, 20]$ (external). Figure 4 shows rounded external utilities for 500 items.

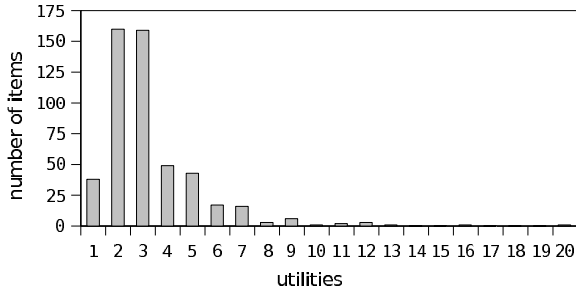


Fig. 4. External utility distribution with 500 distinct items (log-normal distribution, shape parameter = 0.7).

Two databases with 50 000 and 100 000 transactions were used. They can be formalized in Quest notation as follows: $T10.I4.D50000.N750$, $T10.I4.D100000.N1000$ where T is the mean length of transaction, I is the mean length of potentially frequent itemsets, D is the number of transactions and N is the number of distinct items (attributes). 2P-UF algorithm is completely unusable on such large databases, therefore, we do not show its results as the execution was terminated manually due to extraordinary time complexity. We point the reader to the former example with 200 transactions to compare time complexity of both algorithms.

Figure 5 shows the performance of the FUFM algorithm on database $T10.I4.D50000.N750$ with support threshold set to 0.5%.

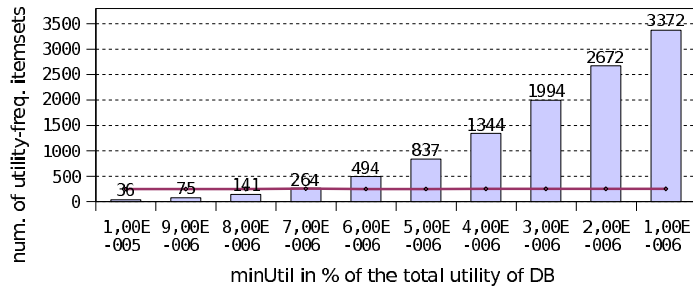


Fig. 5. Number of utility-frequent itemsets and running time of the FUFM algorithm (database $T10.I4.D50000.N750$, minSup = 0.5%).

Red line connecting the columns shows execution time in seconds. As we already mentioned, the actual number of utility-frequent itemsets does not noticeably

influence the total running time of the algorithm. For that reason, the red line is straight and indicates the total running time of approx. 250 seconds.

Table 3 shows the execution of FUFM algorithm on database *T10.I4.-D100000.N1000*. With support threshold set to 0.1% and utility threshold to 147,95 € (0.001% of the total utility of the database) our algorithm finds all 121 utility-frequent itemsets in 1754 seconds. Number of generated candidates can be quite large (second row of the table), but after pruning all low support candidates the remaining number of high support candidates is perfectly acceptable (third row). For practical use the total time of 1754 seconds would be unacceptable, but it should be noted that implementation in C++ could be many times faster and appropriate for even larger databases. As a further improvement a faster frequent itemset mining method could be used instead of APriori.

Table 3. Summary of execution of the FUFM algorithm on database *T10.I4.-D100000.N1000*. minSup = 0.1%, minUtil = 147,95 € = 0.001% of total utility of DB.

iteration	1	2	3	4	5	6	7	8	9	10	11
generated candidates	870	320400	42011	5721	3292	1468	500	131	23	2	0
remaining candidates	801	8783	7156	5563	3233	1441	494	131	23	2	0
utility-frequent itemsets	5	3	12	25	26	22	17	10	1	0	0

5 Conclusions and Further Work

In this paper we introduced a novel, fast algorithm for mining all utility-frequent itemset. It is considerably faster than first algorithm 2P-UF and also much simpler to implement. Because it is based on efficient methods for mining frequent itemset it also performs well on real-sized databases.

Our FUFM algorithm and 2P-UF algorithm were both implemented in Python and tested on a few synthetic databases generated with IBM Quest data generator.

We plan to implement both algorithms in C++ together with a more advanced method for mining frequent itemsets. We also intend to test our algorithm on a real dataset from a retail store and analyze the results which could be used in practice.

References

1. Agrawal R., Imielinski T., Swami A.: Mining association rules between sets of items in large databases. Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data, Washington, D.C., may 1993, pp. 207–216.
2. Borgelt C.: Keeping Things Simple: Finding Frequent Item Sets by Recursive Elimination. Workshop Open Source Data Mining Software, ACM Press, New York, pp. 66-70, 2005.

3. Carter C, Hamilton H. J., Cercone N.: Share based measures for itemsets. In Proc. First European Conf. on the Principles of Data Mining and Knowledge Discovery, pp. 14–24, 1997.
4. Hilderman R. J., Carter C. L., Hamilton H. J., Cercone N.: Mining market basket data using share measures and characterized itemsets. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 159–170, 1998.
5. Han J., Pei J., Yin Y.: Mining frequent patterns without candidate generation. In Proceedings of the Int. Conf. on Management of Data, pp. 1–12, 2000.
6. IBM Almaden research center: Synthetic data generation code for associations and sequential patterns. Available at:
http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html/#assocSynData
7. Li Y. C., Yeh J. S., Chang, C. C.: Direct candidates generation: a novel algorithm for discovering complete share-frequent itemsets. In Proceedings of the 2nd Intl. Conf. on Fuzzy Systems and Knowledge Discovery, pp. 551–560, 2005.
8. Li Y. C., Yeh J. S., Chang, C. C.: Efficient algorithms for mining share-frequent itemsets. In Proceedings of the 11th World Congress of Intl. Fuzzy Systems Association, pp. 543–539, 2005.
9. Podpečan V.: Utility-based Data Mining. BSc Thesis (in Slovene), University of Ljubljana, 2007.
10. ACM SIGKDD Workshop on utility-based data mining, 2005. Available at:
<http://storm.cis.fordham.edu/~gweiss/ubdm-kdd05.html>
11. 11.ACM SIGKDD Workshop on utility-based data mining, 2006. Available at:
<http://www.ic.uff.br/~bianca/ubdm-kdd06.html>
12. Weiss G., Zadrozny B., Saar-Tsechansky M.: Utility-based data mining 2006 workshop report. SIGKDD Explorations, volume 8, issue 2.
13. Yao H., Hamilton H. J., Butz C. J.: A Foundational Approach to Mining Itemset Utilities from Databases. In The Fourth SIAM International Conference on Data Mining SDM, pp. 428–486, 2004.
14. Yao H., Hamilton H. J., Geng L.: A Unified framework for Utility based Measures for Mining Itemsets. Second International Workshop on Utility-Based Data Mining, Philadelphia, Pennsylvania, 2006.
15. Yeh J. S., Li, Y. C., Chang C. C.: A Two-Phase Algorithm for Utility-Frequent Mining. To appear in Lecture Notes in Computer Science, International Workshop on High Performance Data Mining and Applications, 2007.
16. Zaki M. J., Parthasarathy S., Ogihara M., Li W.: New Algorithms for Fast Discovery of Association rules. In Proceedings of the 3rd Intl. Conf. on Knowledge discovery and Data Mining, Newport Beach, California, pp. 283–286, 1997.
17. Zupan B., Demšar J., Leban G.: Orange: From Experimental Machine Learning to Interactive Data Mining. White Paper, (available at: www.aillab.si/orange), Faculty of Computer and Information Science, University of Ljubljana, 2004.

Mining Views: Database Views for Data Mining

Hendrik Blockeel¹, Toon Calders², Elisa Fromont¹,
Bart Goethals³, and Adriana Prado³

¹ Katholieke Universiteit Leuven, Belgium

² Technische Universiteit Eindhoven, The Netherlands

³ Universiteit Antwerpen, Belgium

Abstract. We propose a relational database model towards the integration of data mining into relational database systems, based on the so called virtual mining views. We show that several types of patterns and models over the data, such as itemsets, association rules, decision trees and clusterings, can be represented and queried using a unifying framework. We describe an algorithm to push constraints from SQL queries into the specific mining algorithms. Several examples of possible queries on these mining views, using the standard SQL query language, show the usefulness and elegance of this approach.

1 Introduction

Data mining is an iterative and interactive process. During the whole discovery process, typically, many different data mining tasks are performed, their results are combined, and possibly used as input for other data mining tasks. To support this knowledge discovery process, there is a need for integrating data mining with data storage and management. The concept of inductive databases (IDB) has been proposed as a means of achieving such integration [6].

In an IDB, one can not only query the data stored in the database, but also the patterns that are implicitly present in these data. The main advantages of integrating data mining into database systems are threefold: first of all, the data is mined where the data is: in the database. Hence, the need for transforming data into an appropriate format is completely removed. Second, in database systems, there is a clear separation between the logical and the physical level. This separation shields the user from the physical details, making the technology much more accessible for a non-specialist. Ideally, the user of an inductive database should not be involved with selecting the right algorithm and parameter setting, storage format of the patterns, etc., but should instead be able to specify, in a declarative way, the patterns in which he or she is interested. The third advantage of an inductive database is the flexibility of ad-hoc querying. In an inductive database, the user is not limited by the functionality offered by a limited set of tools. Instead, he or she can specify new types of patterns and constraints. Notice that data mining suites such as, e.g., Weka [19] and Yale [11] only share the first advantage of inductive databases by imposing one uniform data format for a group of algorithms.

In this paper, we focus our attention on determining how such an inductive database can be designed in practice. The solution proposed in this paper builds upon our preliminary work in [2, 3]. In contrast to the numerous proposals for data mining query languages [4, 7, 8, 10, 15, 17, 18], we propose to integrate data mining into database systems without extending the query language. Instead, we extend the database schema with new tables that contain, for instance, association rules, decision trees, or other descriptive or predictive models.

One might argue against this approach that tables containing all possible patterns and models over the data would in most cases be huge. These tables, however, are in fact implemented as views, called *virtual mining views*. Whenever a query is formulated that selects for instance association rules from these tables, this triggers a run of a data mining algorithm (e.g., Apriori [1]) that computes the result of the query, in exactly the same way that normal views in databases are only computed at query time, and only to the extent necessary for answering the query.

This querying approach naturally integrates constraint-based mining. Within the query, one can impose conditions on the kind of patterns that one wants to find. In many cases, these constraints can be pushed into the mining process.

The paper is organized as follows. Section 2 focuses on the related work. We introduce our new framework and the mining views it uses in Section 3. In Section 4 we illustrate how standard SQL queries on these views allow us to search for models fulfilling certain constraints or to apply a given model to a new dataset. In Section 5, we extend the constraint extraction algorithm from [2] for decision trees and clusterings. We conclude in Section 6.

2 Related Work

There already exist multiple proposals for extending a query language with some data mining primitives. The most well-known examples are the SQL-like operator MINE RULE of Meo et al. [10] for mining association rules, and the data mining query language DMQL by Han et al. [4]. In both studies, however, the language constructions only allow to specify the desired output, but this output is not integrated again into the database. Our proposal goes beyond this, by also allowing the results to be used as input for further data mining queries, as they are treated as regular database tables.

In Microsoft's Data Mining extensions (DMX) of SQL server [15], a classification model can be created. This model can be trained and used afterwards to give predictions, via the so-called prediction joins. However, this framework does not provide any operations other than browsing and prediction for manipulating the model, and there is no notion of composing mining operations in their framework. Although the philosophy behind the predictor join is somewhat related to our proposal, the work presented in this paper goes much further.

Siebes [14] argues in favour of making patterns and models first-class citizens, and suggests to extend for instance the relational algebra with operations on models. In our framework, predictive models are already first-class citizens in the

relational algebra itself, as they are simply relations. As such, the operation of applying a predictive model M to an instance x simply corresponds to a selection and projection from M : $\pi_Y(\sigma_{X=x}(M))$; the composition of two predictive models is their join, etc.

The closest to the work presented in this paper are \mathcal{LDL}^{++} [17] and ATLaS [9, 18, 20]. \mathcal{LDL}^{++} and ATLaS are extensions of respectively \mathcal{LDL} and SQL that add the ability of defining new *user defined aggregates* (UDAs), making them suitable for data mining. Especially ATLaS is very interesting with respect to our proposal, as it is also based on the principles of relational databases and query languages. In ATLaS, however, the query language is much more powerful (even Turing complete). In fact, ATLaS is rather a programming language based on SQL, enabling data mining operations, on top of relational databases. Hence, in ATLaS, the results of mining have to be encoded into the relational model, and subsequent queries of found patterns have to deal with decoding and encoding the found patterns. Also, the ATLaS query language is much less declarative, making it less attractive for query optimization.

As already pointed out in the introduction, the work presented here builds upon our own preliminary work on the integration of association rule mining and decision tree learning into database systems [2, 3]. This paper significantly improves upon these works in the following way. The representations of association rules and decision trees, as proposed in our earlier work, are fairly complex. In this work, we propose a new unifying representation that is more elegant and simpler than the originally proposed representations. It focuses more on the semantics of learned models rather than their structure, and as such allows us to handle conceptual models such as association rules, decision trees and clusterings in a general way. For instance, applying a model to classify a new example now amounts to a simple join operation, while involving much more complex queries using the previous representation. Furthermore, the constraint extraction algorithm of [2] is extended to support queries about predictive models as well.

3 Framework Representation

Given a table $T(A_1, \dots, A_n)$, let $Dom(T) = Dom(A_1) \times \dots \times Dom(A_n)$ denote the domain of T . We create a *Concept Table* $Concepts_T(Cid, A_1, \dots, A_n)$, such that for every tuple t in T , there exist 2^n unique tuples $\{t'_1, \dots, t'_{2^n}\}$ in $Concepts_T$ such that $t'_i.A_j = t.A_j$ or $t'_i.A_j = '?'$ for all $i \in [1, 2^n]$ and $j \in [1, n]$. We denote the special value '?' as the *wildcard value* and assume it doesn't exist in the domain of any attribute. As each of the concepts can actually cover more than one tuple in T , a unique identifier Cid is associated to each concept.

A tuple, or *concept*, $(cid, a_1, \dots, a_n) \in Concepts_T$ represents all tuples from $Dom(T)$ satisfying the condition $\bigwedge_{i|a_i \neq '?'} A_i = a_i$.

Figure 1 shows a data table for the classic *PlayTennis* example [12], together with a sample of its corresponding Concepts table.

Day	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
...

Cid	Day	Outlook	Temp	Humidity	Wind	Play
1	?	Sunny	?	High	?	No
2	?	Sunny	?	Normal	?	Yes
3	?	Overcast	?	?	?	Yes
4	?	Rain	?	?	Strong	No
5	?	Rain	?	?	Weak	Yes
6	?	?	?	?	?	?
...

Fig. 1. The PlayTennis data table and its corresponding Concepts table.

3.1 Representing models as sets of concepts

Given a data table T , and its corresponding Concepts table $Concepts_T$, we now explain how a variety of models can be represented using so called *virtual mining views* [2]. Although all mining views are defined over T , we omit the subscript T when it is clear from the context.

Itemsets and association rules Obviously, as itemsets in a relational database are conjunctions of attribute-value pairs, they can be represented as concepts. The result of frequent itemset mining can therefore be represented by a view $Sets(cid, supp)$. For each itemset, there is a tuple with cid the identifier of the itemset (concept) and $supp$ its support. Also other attributes, such as χ^2 or any correlation measure, could be added to the view to describe the itemsets. Similarly, association rules can be represented by a view $Rules(rid, cida, cidc, cid, conf)$, where rid is the rule identifier, $cida$ and $cidc$ are the identifiers for the concepts representing the antecedent and the consequent of the rule respectively, cid is the union (disjunction) of these, and $conf$ is the confidence of the rule. Again, many other attributes, such as lift, conviction, or gini index, could be added to describe the rules.

Predictive models In association rule discovery, results typically describe the dataset itself, but in inductive learning, one is interested in building from the training set a model of a broader population, from which the training set is a representative sample. Therefore, it is useful to distinguish T , the table from which we learn (the training set), $Dom(T)$, the domain of that table (the set of all conceivable instances), and P , the actual population from which T is a random sample. P may be a strict subset of $Dom(T)$: not every conceivable tuple may exist in the real world. For instance, if $T(Gender, Age, Pregnant?)$ is the set of all patients currently in some hospital, which is a subset of the set P of all possible patients, then $P \subset Dom(T)$: a tuple with $Gender=$ male and $Pregnant?=$ true would be in $Dom(T)$ but not in P .

From this point of view, we can describe inductive learning (whether it is descriptive or predictive) as deriving P from T . Generally, P can be described as a probability distribution over $Dom(T)$, but here we will focus on the simpler case where P is a subset of $Dom(T)$. P can then be represented in tabular form, using exactly the same schema as T , since both are subsets of $Dom(T)$.

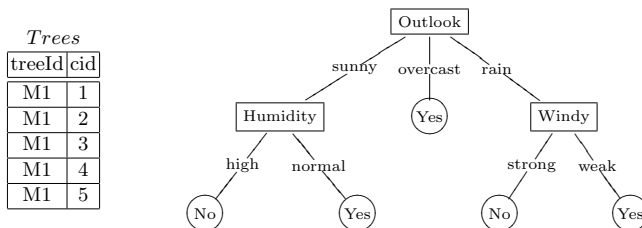


Fig. 2. Decision Tree built to predict the attribute Play.

In practice, machine learning systems would obviously learn P not in tabular format but, e.g., in the form of a decision tree. Such a tree represents a function from some attributes of T to a target attribute; this function is a relation, it is this relation that we call P .

Note that the term “model” may be used for both the representation of a model and its semantics (the relation it represents). In this text, we will use both meanings; it will usually be clear from the context what is meant.

We will now use concepts from the *Concepts* table to describe the semantics, and where possible also part of the structure, of the models learned. For instance, we represent all decision trees that can be learned from T , by the view *Trees*(*treeId*, *cid*). A unique identifier *treeId* is associated to each tree and each of the trees is described using a set of concepts (each concept describing one leaf). Figure 2 shows a decision tree built to predict the attribute Play using all other attributes in the data table, and its representation in the mining view *Trees*, using the *Concepts* table from Figure 1. If the user wants to build a tree from a subset of the attributes of the data table, he or she should first create a view on the data table that contains exactly the attributes he or she is interested in; a *Concepts* table related to this view can then be used to describe the trees.

Additionally, a view representing several characteristics of a tree learned for one specific target attribute A_i is added: *Trees_Charac* $_A_i$ (*treeId*, *accuracy*, *size*, *cost*, ...). For every tree, there is a tuple with a tree identifier *treeId* and its corresponding characteristics.

Clustering In unsupervised learning (clustering), we can see $T(A_1, \dots, A_n)$ as a projection of a relation $T'(A_1, \dots, A_n, C)$ onto $A_1 \dots A_n$ (the target attribute, indicating the class or cluster of each instance, is unobserved). P then has one more attribute than T , and can be seen as a predictive model over $Dom(T')$.

In extensional clustering, where the clusters are assumed to contain just the elements from the training set, P can be considered to have as many tuples as T . In intentional clustering, P would generalize over T in a similar way than predictive models do; i.e., it also assigns previously unseen instances to a cluster.

All possible clusterings that can be learned from T are represented in the view *Clusterings*(*clusId*, *clId*) and all clusters belonging to all clusterings are represented in the view *Clusters*(*clId*, *cid*). A unique identifier *clusId* is asso-

ciated to each clustering and each of the clusterings is described by a set of clusters. A unique identifier *clId* is associated to each cluster and each of the clusters is described by a set of concepts.

Again, a view representing the characteristics of all clusterings is added: *Clusterings_Charac*(*clusId*, *size*, ...), with *size* the number of clusters. Of course other attributes could be added to this view.

Note that since the Concepts table has a finite number of elements (depending on the data table), the number of partitions (for clustering) as well as the number of trees that can be described using these concepts is also finite.

3.2 Discussion

The framework proposed in this paper is conceptually very different from the ones given in [2, 3] since it focuses on the representation of the semantics of the models (the function they represent) rather than on its structure (although, through the use of wild cards, limited information about the model structure is still available). Advantages of this approach are that it offers a unifying framework for all models, and that certain operations, such as using the model for prediction, become easier. On the contrary, queries about the structure, such as asking which attribute is at the root of a learned tree, become cumbersome. But, we believe that having a representation that is both suited for representing the complete structure and the semantics of the models is unrealistic. Descriptive or predictive models can be represented in many different formats (for example the ones proposed in [2, 3]), and for all these formats, views describing the model structure should be designed separately from the framework proposed here, and according to the needs and the preferences of the user.

The views representing the characteristics of a model provide information that may or may not be derivable from the tables that represent the models themselves. For instance, one could extend them with a full description of a decision tree (not just its set of leaves, as described by the Trees table). Including redundant information in the characteristics tables may simplify the formulation of constraints as queries.

In our current implementation, the identifiers in the mining views are “system-generated” values that have no meaning in the real world. Only equality of these identifiers within a query is well-defined; the same concept in different queries may have different identifiers, and vice versa. To avoid this, a kind of canonical encoding for (sets of) concepts would be needed, which is easy if the set of all concepts is known in advance but not when it depends dynamically on the extension of the relation, as is the case with our definition.

4 Model Querying

In this section, we give some concrete examples of common data mining tasks that can be expressed with SQL queries over the mining views. Compared to [2, 3], the new constructs introduced in this paper simplify the expression of queries

for prediction and allow for a more declarative description of constraints on the desired models.

4.1 Prediction

In [3], to support decision trees in a relational database, the structure of decision trees was coded in a relational table. To predict the class of a new example, a query has to be written that explicitly expresses, almost in a procedural way, how the tree stored in the relation needs to be used. In our framework, a predictive model is stored as a set of concepts, capturing instead the semantics of the model. In order to classify a new example using one or more of the learned classifiers, one simply looks up the concept that covers the new example. More generally, if we have a test set S , all predictions of the examples in S are obtained by equi-joining S with the semantic representation of the classifier. As the concepts table is just a compact representation of this semantic view, we join S to *Concepts* using a variant of the equi-join that requires that either the values are equal, or there is a wild card.

Consider the classic *PlayTennis* example. The following query predicts the attribute Play for all unclassified examples in table *Test_Set*, considering all possible decision trees of size ≤ 5 in table *Trees*.

Day	Outlook	Temp	Humidity	Wind
D7	Sunny	Hot	High	Weak
D8	Rain	Hot	High	Strong
D9	Overcast	Hot	High	Weak
D10	Overcast	Mild	High	Weak
D11	Overcast	Cool	Normal	Weak
D12	Sunny	Cool	High	Strong

```

select T.treeId, S.*, C.Play
from Test_Set S,
     Trees T,
     Concepts C,
     Trees_Charac_Play D
where T.cid = C.cid
and (S.Outlook = C.Outlook or C.Outlook = '?')
and (S.Temp = C.Temp or C.Temp = '?')
and (S.Humidity = C.Humidity or C.Humidity = '?')
and (S.Wind = C.Wind or C.Wind = '?')
and T.treeId = D.treeId
and D.size <= 5

```

4.2 Constraints

In this section, we discuss how typical constraints on association rules, decision trees or clusterings can be formulated as part of an SQL query. In our framework, these constraints can be expressed elegantly and more declaratively than in previous proposals.

For association rules, we consider constraints such as minimal and maximal support, minimal and maximal confidence, plus the constraints that a certain item must be in the antecedent, in the consequent, and boolean combinations of these. For decision trees, we consider the constraints size and accuracy. In addition to these, we also consider constraints posed on the concepts that describe the trees. For clusterings, we consider their size (number of clusters) and the popular constraints must-link (two instances must be in the same cluster) and cannot-link (two instances must not be in the same cluster) [16]. Next to these well-known constraints, in our approach, the user has also the ability to come up with new, ad-hoc constraints. In contrast, other proposals in the literature

```

(A)
select R.rid,
       C1.*, C2.*,
       R.conf
from Sets S,
     Rules R,
     Concepts C1,
     Concepts C2
where R.cid = S.cid
and C1.cid = R.cida
and C2.cid = R.cidc
and S.supp >= 30
and R.conf >= 80

(B)
select T.*
from Trees_charac_Play T
where T.accuracy =
      (select max(accuracy)
       from Trees_Charac_Play T1
        and T1.size <= 5)
and T.size <= 5

(C)
select T1.treeId,
       C1.*, C2.*
from Trees T1,
     Trees T2,
     Concepts C1,
     Concepts C2,
     Trees_Charac_Play D
where T1.treeId = T2.treeId
and T1.cid = C1.cid
and C1.Outlook= 'Sunny'
and T2.cid = C2.cid
and C2.Wind = 'Weak'
and T1.treeId = D.treeId
and D.size <= 5
and D.accuracy >= 0.8

(D)
select T.treeId, C.*
from Trees T,
     Concepts C
where T.cid = C.cid
and not exists
      (select *
       from Concepts C1
        where C1.cid = C.cid
         and C1.Temp = '?')

(E)
select C.clusId
from Clustering C,
     Clusters C11,
     Clusters C12,
     I_Concepts I1,
     I_Concepts I2
where I1.Day = 'D1'
and I2.Day = 'D2'
and C.clId = C11.clId
and C11.clId= C12.clId
and C11.cid = I1.cid
and C12.cid = I2.cid

(F)
select C1.clusId
from Clustering C1,
     Clustering C2,
     Clusters C11,
     Clusters C12,
     I_Concepts I1,
     I_Concepts I2,
where I1.Day = 'D1'
and I2.Day = 'D2'
and C11.cid = I1.cid
and C12.cid = I2.cid
and C1.clusId=C2.clusId
and C1.clId = C11.clId
and C2.clId = C12.clId
and C1.clId <> C12.clId

```

Fig. 3. Example mining queries.

that extend the query language, in general do not allow this flexibility; only those constraints the language designer explicitly added to the language can be expressed.

Consider, again, the table *PlayTennis*. Figure 3 illustrates several mining queries that can be posed in our inductive database. Some constraints can be directly imposed using the tables *Sets*, *Rules*, *Trees_Charac* or *Clusterings_Charac* as shown in queries (A), (B) and (C). Query (A) asks for association rules having support of at least 30 and confidence of at least 80%. Query (B) selects decision trees having the attribute *Play* as the target attribute, having maximal accuracy among all possible decision trees of size ≤ 5 . Query (C) asks for decision trees having a test on “Outlook=Sunny” and on “Wind=Weak”, with a size of at most 5 and an accuracy of at least 80%.

Some constraints can also be imposed independently from the tables with the characteristics. For example, Query (D) asks for decision trees where the attribute *Temp* is never a wild card.

The popular must-link and cannot-link constraints, for clusterings, can also be expressed with SQL queries in our approach. Queries (E) and (F), respectively, are examples of how the user can formulate such constraints. In both queries, *I_Concepts(Day, cid)* is a view associating every instance in the data table with its covering concepts, which can be easily created by the user. Hence, query

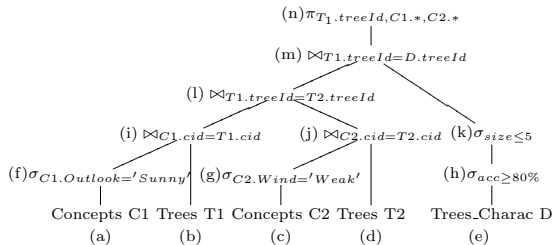


Fig. 4. An equivalent relational algebra for query (C) in Figure 3.

(E) asks for clusterings in which the instances “D1” and “D2” are in the same cluster, that is, in which both instances are covered by concepts describing the same cluster. Query (F) is formulated by using the opposite reasoning.

Hence, many well-known and common constraints can be expressed quite naturally in our model. In particular, queries that impose semantic restrictions, such as queries for prediction, or semantic constraints, such as must-link and cannot-link constraints, can be expressed more declaratively in our new framework. This more declarative nature of the queries also improves the ability to extract and exploit constraints in the queries imposed by the user for making the underlying mining operations more efficient.

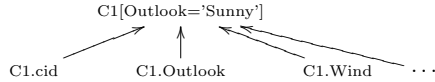
5 Constraints Extraction

In the proposed framework, the tables are virtual. This means that for answering a query involving one or more of these views, we first need to instantiate them with the information needed by the query. Obviously, adding all concepts to the *Concepts* table, all trees to the *Trees* table, etc. is infeasible. However, since we expect the user to give a reasonable amount of constraints, only a subset of all tuples will be needed to answer the query correctly. In this section, we propose an algorithm that extracts constraints on the models needed to be mined from a given SQL query over the virtual views. These constraints can then be exploited by the data mining algorithm that is going to compute the result of the query. Consider for example query (C) in Figure 3. In order to answer that query, not all decision trees need to be mined, but only those with a size of at most 5, an accuracy of at least 80%, and node tests “Outlook=Sunny” and “Wind=Weak.” In this context, the task of the constraint extraction algorithm is to extract these constraints from the query, such that they can be exploited by the tree inducer that has to be triggered to compute the result. Our constraint extraction algorithm finds constraints for all tables in the from-clause of the query, hence restricting the tuples required in the views to answer the query.

Algorithm The algorithm for extracting the constraints is an extension of the algorithm presented in [2], which only extracts constraints for itemset and

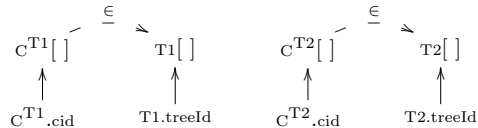
association rule queries. It starts by building an equivalent relational algebra tree. For the example query (C), the tree is given in Figure 4. Notice that the views in the from-clause of the query correspond to the leaves of the expression tree. The idea is to find a constraint for each of those nodes while traversing the expression tree bottom-up. During the traversal, annotations expressing the constraints are computed for each of the nodes, based on the relational algebra operator in that node and the annotations of its children in the tree. For a node n , the annotation expresses the set of tuples needed in order to answer the sub-query rooted at that node. Hence, the annotation for the root node is the one we are looking for.

Annotations Consider, e.g., node (f) in the example query given in Figure 4. The sub-query associated with this node asks for all tuples in the table *Concepts* $C1$ with “Outlook=Sunny”. The annotation for this node is:

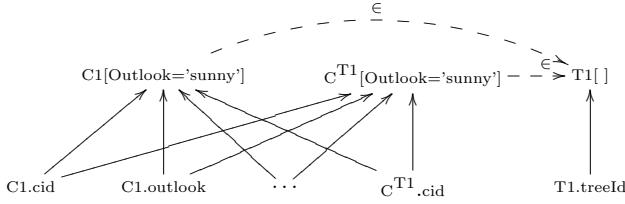


The top line in this annotation gives the views needed in order to answer the sub-query rooted at node (f). Between the square brackets a constraint on the tuples needed from this view is given. In the example node (f), we only need those tuples in the view *Concepts* $C1$ that satisfy “Outlook=’Sunny’”. The bottom line in the annotation lists all attribute names of the sub-query. The arrows represent the view they originate from. Notice that an attribute can originate from more than one view. This occurs, e.g., when two relations were joined on this attribute. If, later on, an attribute is used in the condition of a selection in the tree, the constraint(s) of the view(s) from which that attribute originates will be updated.

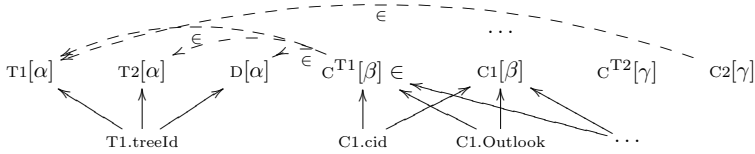
The construction procedure of the annotations needed for association rules and itemsets presented in [2] applies to our new framework as well. For decision trees and clusterings, however, the rules for constructing the annotations are more complicated. Indeed, while itemsets are described by a single concept and association rules are described by two concepts only (i.e., the antecedent and the consequent), decision trees and clusterings are both described by a priori unknown sets of concepts. In fact, a tuple $(treeId, cid)$ of the view *Trees* represents two objects: the tree identified by $treeId$ and the concept with identifier cid belonging to it. In our annotation, this is expressed by two variables, one for the tree and one for the concept. They are connected with a dashed arrow from the concept to the tree, expressing that this concept belongs to that tree. For example, the annotation of nodes (b) and (d) are respectively:



In node (i), $C1$ and $T1$ are joined on attribute cid , the annotation of node (i) will express that both concepts $C1$ and C^{T1} belong to the tree $T1$ and that they are actually the same concept. Every attribute of $C1$ also belongs to C^{T1} (and vice versa) and they inherit the constraints between the squared brackets. Hence, the annotation of (i) is as follows:



The annotations for the other nodes are built in the same way, resulting in the following:



$$\alpha = (acc \geq 80\% \wedge size \leq 5), \beta = (Outlook = 'Sunny'), \gamma = (Wind = 'Weak')$$

To ease the readability, we did not draw all \in -arrows, but, actually, from every variable representing a concept, there is an arrow to $T1$, $T2$, and D . From this final annotation, finding the final constraints on the mining views is straightforward: for example, for the view $T1$, we see that not all possible trees are needed, but only those that satisfy condition α , and, also, have concepts that satisfy conditions β and γ . These constraints can be exploited directly by a tree inducer. Also for the other views, constraints can be extracted. For the Concepts table, e.g., it can be derived that not every concept should be there, but only concepts belonging to trees in $T1$.

6 Conclusion and Future Work

In this paper, we proposed a framework towards the integration of (constraint-based) data mining in a relational database, based on the so-called *mining views*. A mining view is a virtual table that contains models of the data. The main advantage over earlier proposals is that our schema elegantly covers a wider variety of models in a more uniform way, and that it makes it easier to define meaningful operations (e.g., predictions of new examples). A key component of

the proposed approach is the use of the virtual *Concepts* table, which contains all conjunctive concepts definable over the relation that is being mined. We have illustrated how association rules, decision trees and clusterings can uniformly be expressed in terms of this *Concepts table*. Furthermore, we have shown how to formulate constraints in a query, using this structure, and how to automatically extract constraints from a given query for these different models.

As a proof of concept, the ideas presented in this work have been implemented into PostgreSQL [5]. The system is currently linked to algorithms for association rule discovery and exhaustive decision tree learning [3] (an exhaustive clustering algorithm is not yet available). The prototype shows promising results, for instance: for the UCI dataset *ZOO* [13], a query for all association rules with constraints $\text{support} \geq 30$ and $\text{confidence} \geq 80\%$ is executed in 2.3 seconds; querying for all decision trees with $\text{size} \leq 5$ (without further constraints) takes 3.6 seconds.

We identify three directions for further work. First, in the current system, if the database is modified between two queries, the efficiency of the system could still be improved by investigating how to reuse the previously computed predictive models in order to compute new predictive models for the modified database. Second, it might also be interesting to reuse the results of related queries posed within the same working session. Finally, the schema described so far covers association rules, decision trees and clusterings. An obvious direction for further research is to extend it with other models.

Acknowledgements Hendrik Blockeel is a post-doctoral fellow from the Research Foundation – Flanders (FWO-Vlaanderen). This research was funded through K.U.Leuven GOA project 2003/8, “Inductive Knowledge bases”, FWO project “Foundations for inductive databases” and the EU project “Inductive Queries for Mining Patterns and Models”.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.
2. T. Calders, B. Goethals, and A. Prado. Integrating pattern mining in relational databases. In *Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, LNCS*, pages 454–461. Springer, 2006.
3. E. Fromont and H. Blockeel. Integrating decision tree learning into inductive databases. In *ECML/PKDD-2006 International Workshop on Knowledge Discovery in Inductive Databases (KDID)*, pages 59–70, 2006.
4. J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases. In *Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.
5. <http://www.postgresql.org/>
6. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Comm. Of The Acm*, 39:58–64, 1996.

7. T. Imielinski and A. Virmani. Msql: A query language for database mining. *Data Min. Knowl. Discov.*, 3(4):373–408, 1999.
8. S. Kramer, V. Aufschild, A. Hapfelmeier, A. Jarasch, K. Kessler, S. Reckow, J. Wicker, and L. Richter. Inductive Databases in the Relational Model: the Data is the Bridge. In *ECML/PKDD-2005 International Workshop on Knowledge Discovery in Inductive Databases (KDID)*, pages 124–138, 2005.
9. Y.-N. Law, H. Wang, and C. Zaniolo. Query languages and data models for database sequences and data streams. In *Proc. VLDB Int. Conf. Very Large Data Bases*, pages 492–503, San Francisco, CA, USA, 2004.
10. R. Meo, G. Psaila, and S. Ceri. An extension to sql for mining association rules. *Data Min. Knowl. Discov.*, 2(2):195–224, 1998.
11. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks, 2006.
12. T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
13. D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.
14. A. Siebes. Data Mining in Inductive Databases. In *ECML/PKDD-2005 International Workshop on Knowledge Discovery in Inductive Databases (KDID)*, pages 1–23, 2005.
15. Z. H. Tang and J. MacLennan. *Data Mining with SQL Server 2005*. John Wiley & Sons, 2005.
16. K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th Int. Conference on Machine Learning*, pages 1103–1110, 2000.
17. H. Wang and C. Zaniolo. Nonmonotonic reasoning in $ldl++$. *Logic-based artificial intelligence*, pages 523–544, 2001.
18. H. Wang and C. Zaniolo. Atlas: A native extension of sql for data mining. In *SIAM Intl. Conf. Data Mining*, pages 130–144, 2003.
19. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.
20. C. Zaniolo. Mining databases and data streams with query languages and rules. In *ECML/PKDD-2005 International Workshop on Knowledge Discovery in Inductive Databases (KDID)*, pages 24–37, 2005.

Part III

Short Presentations – Papers

***Onto4AR*: a framework for mining association rules**

Cláudia Antunes¹

¹ Instituto Superior Técnico / Technical University of Lisbon
Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal
claudia.antunes@dei.ist.utl.pt

Abstract. Despite the efforts made on last decades to center the process of knowledge discovery on the user, the balance between the discovery of unknown and interesting patterns is far from being reached. The discovery of association rules is a paradigmatic case, where this balance is quite difficult to establish. In this paper, we propose a new framework for pattern mining – the *Onto4AR*. This framework is centered on the use of ontologies, for the representation and introduction of domain knowledge into the mining process. By defining constraints based on an ontology, the framework provides a mining environment independent of the problem domain. With this simplification on the definition and use of constraints, the framework contributes to reduce the gap between discovered rules and user expectations.

Keywords: Association rules, Background knowledge, Ontologies, Constraints.

1 Introduction

In the information society, where the communication is instantaneous and the access to registered data is generalized, one of the major difficulties, faced by organizations, is related to the management of data abundance and its use on the decision support. Among the available tools to manage those large amounts of unstructured data, are *association rules*. The problem of its discovery was first introduced in 1993 [1], for finding sets of items purchased in the same transaction a significant number of times. Since then, association rules have been applied to a variety of domains, from medicine to marketing, with a fair success. Indeed, and despite the improvements on the algorithms for this task, the massive number of discovered rules turns their manual analysis impossible. On the other hand, the discovery of obvious rules diminishes the interest of users for this technique.

In this manner, a decade later, the discovery of association rules continues to present several challenges. On one hand, it is important to choose the interesting patterns to present to the user, filtering obvious rules; and on the other hand, it is mandatory that the user can control the mining process, in the sense that he should be able to incorporate his expectations and domain knowledge in the discovery process. Despite both issues have deserved considerable attention, namely in the definition of interestingness measures and languages to express constraints, their usage has not been exported to real situations, as desirable.

In this paper, we propose a new framework for discovering association rules – the *Onto4AR framework*. This framework provides an environment for specifying constraints that enables the user to control the mining process. But instead of using a specific language to introduce the constraints, it uses an ontology to represent existing background knowledge. Additionally, the user can choose the kind of constraint to be applied, both reducing the number of discovered patterns and their scope. The constraints can be chosen among a set of pre-defined ones based on the relations among concepts expressed in the ontology. In this manner, the framework provides a universal environment for mining association rules, since it is able to represent background knowledge in any domain, and supplies a set of parameterized constraints, defined independently of the problem context. Moreover, the framework allows for the definition of any new kind of constraints. In particular it is possible to introduce other content constraints, given that they are defined over the structure of ontologies.

The rest of the paper is organized as follows: next (in section 2), we overview a few interesting approaches that use background knowledge in the mining process, followed by a general description of ontologies (in section 3). In section 4, we describe the entire framework, presenting its main concepts and the adaptation of pattern mining algorithms to use proposed constraints. The section ends with a categorization of constraints that can be defined in the framework. The paper ends with the discussion of the benefits of the described approach and some guidelines for future work.

2 MINING RULES WITH CONSTRAINTS

The problem of mining association rules is defined as the discovery of “all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence respectively”. With an *association rule* corresponding to an implication of the form $A \Rightarrow B$, where A and B are propositions (sets of pairs attribute/value, most of the times named *items*), and expresses that when A occurs, B also occurs with a certain probability. This probability is known as the rule *confidence* and is given by the conditional probability $P(B|A)$. The *support* of the same rule is given by the number of transactions that include A and B , simultaneously. In basket analysis, one of the major applications of association analysis, A and B correspond to itemsets, and the rule $A \Rightarrow B$ means that if A is transacted, then B will also be transacted at the same time, with a certain probability. The problem includes two separate phases: the discovery of frequent patterns and the generation of rules from those patterns. Since the second step demands the generation of all possible combinations among the items that constitute each pattern, research has been focused on the discovery of frequent patterns.

Since its definition, several algorithms have been proposed that efficiently deal with this problem, but most of them did not give any contribution to reduce the number of discovered rules. For example, most recent algorithms that find closed patterns (see for example CLOSET+ [15]), retrieve exactly the same patterns as traditional algorithms, like Apriori [1], despite they generate considerable less

patterns than those algorithms. In fact, the most effective technique to reduce the number of discovered patterns is the use of constraints. As pointed by Bayardo, constraints play a critical role in solving the trade-offs of the generality of data mining algorithms, by focusing "the algorithm on regions of the trade-off curves (or space) known (or believed) to be most promising" [5]. By using constraints, the user assumes the responsibility of choosing which of those aspects are most important for the current task.

In the problem of mining association rules, we can distinguish between two main categories of constraints: interestingness measures and content constraints. *Interestingness measures* rank the discovered patterns or rules, by quantifying the usefulness and utility of them, discarding those with an evaluation less than a user-specified threshold. Examples of such measures are *confidence*, *lift* or the *minimum improvement* [4]. With these constraints, it is possible to both improve the performance of the algorithms, by pruning uninteresting patterns, and reduce the number of discovered patterns. The other category of constraints, *content constraints*, can be seen as filters over the content of the discovered patterns, instead of its relevance. While interestingness measures are quantitative metrics, content constraints are predicates on the powerset of the set of items [14]. In some sense, they capture application semantics and introduce it into the mining process. An example of such constraints is the *item constraint*, as proposed by Srikant *et al* [16]. Despite the advantages coupled to the use of content constraints, by allowing the incorporation of domain knowledge, they have been applied seldom.

Despite the difficulties in applying existing background knowledge in real situations, in the last decade, there were a few proposals that contribute to develop this area. The first contribution was made by Srikant *et al* [16], which besides proposing item constraints, introduced the use of taxonomies to discover generalized patterns. A taxonomy is a *is-a* hierarchy defined over the domain in analysis and is often available as the unique background knowledge. By making use of taxonomies and specifying the items of interest, it is possible to focus the process on the user, either by transferring the control to him or by introducing part of his background knowledge in the mining process. With the same goal, but in the context of sequential pattern mining, Garofalakis [8] and Antunes [2] proposed the use of regular and context-free languages to constrain the discovery. Note that in areas, like sequential pattern mining, where constraints can be specified with such restrictive power, the risk of turning the mining process in a simple hypothesis-test becomes a reality. In order to solve this antagonism, more recently, the use of constraint relaxations was proposed [3].

Another important research way in the specification of constraints is the use of logic-based languages. Usually, these languages combine inductive and deductive perspectives of data mining and provide the means to introduce constraints in the mining process. The work by Bonchi [6] gives a good overview of such languages. These languages can also be used as a filter the number of discovered patterns, in a post-processing step, like the *mine rule* operator [7]. Again with post-processing in mind, Vanzin [17] proposed the use of Ontologies to manage the discovered patterns. In this work, the user can navigate over the patterns, following the knowledge represented in a domain ontology. With a wider range, some other works propose the

use of ontologies for guiding the knowledge discovery process (KDD). The work by Češpivová et al. [8] shows that ontologies can be used in all KDD phases, from business and data understanding to modeling, evaluation and deployment phases. Another example is the MiningMart system [10], which provides the tools for pre-processing data in order to facilitate the evaluation and usage of the mining results.

Next, we describe succinctly the core concepts in ontologies, and exemplify its use in the domain of basket analysis.

3. Ontologies

One of the great problems related to the use of background knowledge is the difficulty to acquire it complete and effectively, since, business agents usually have some difficulties on expressing their knowledge. Fortunately, the recent developments in the area of knowledge management make possible the representation of background knowledge as Ontologies.

An *ontology* is an explicit specification of a conceptualization, which means that is a specification of an abstract, simplified view of the domain. Formally, and in accordance with [12], an ontology is a 5-tuple $\mathcal{O} = \{\mathcal{C}, \mathcal{R}, \mathcal{H}, rel, \mathcal{A}\}$, where \mathcal{C} is a set of concepts, who represent the entities in the ontology domain; \mathcal{R} is a set of relations defined among concepts; \mathcal{H} is a taxonomy or concept-hierarchy, which defines the is-a relations among concepts ($\mathcal{H}(C_1, C_2)$ means that C_1 is a sub-concept of C_2 , or in other words C_2 is a parent of C_1); rel is a function, $rel: \mathcal{R} \rightarrow \mathcal{C} \times \mathcal{C}$, that specifies the relations on \mathcal{R} (if $R \in \mathcal{R}$, $rel(R) = (C_1, C_2)$ is also written as $R(C_1, C_2)$, and means that C_1 is related to C_2 , but the inverse is not necessarily true); and finally, \mathcal{A} is a set of axioms, usually expressed in a logical language, that describe constraints on the ontology, expliciting implicit facts.

In the counterpart of ontologies are knowledge bases, which specify existing instantiations for a particular ontology. Formally, a *knowledge base* is a 4-tuple $\mathcal{KB} = \{\mathcal{O}, \mathcal{I}, inst, instr\}$, where \mathcal{O} is a ontology as defined above; \mathcal{I} a set of instances; $inst$ a function from \mathcal{C} to $2^{\mathcal{I}}$ called *concept instantiation* (with \mathcal{C} , the set of concepts in \mathcal{O}) and $instr$ the relation instantiation function defined from \mathcal{R} to $2^{\mathcal{I} \times \mathcal{I}}$. (It is usual to designate the concept instantiated by an instance, as its *class*). In this manner, while ontologies try “to capture the conceptual structures of a domain of interest” [12], defining its elements, and describing the relations among them; a knowledge base defines a set of elements, which can be interpreted and understand with respect to an ontology.

With the developments of the ontology research and the semantic web, several ontology specification languages have been proposed. See the work by Corcho et al [8] for an overview and comparison of such languages.

3.1 An Example

Fig. 1 illustrates a simple knowledge base \mathcal{KB} constituted by a simple ontology \mathcal{O} , which main concepts are alimentary products and meals. The knowledge base is

represented in UML, where *concepts* are represented by ellipses, *is-a relations* as non-named arrows, and other *relations* by named stronger arrows. Double-sided arrows represent two inverse relations (just for draw simplification). Instances are represented by shadowed ellipses and instantiations by dotted arrows.

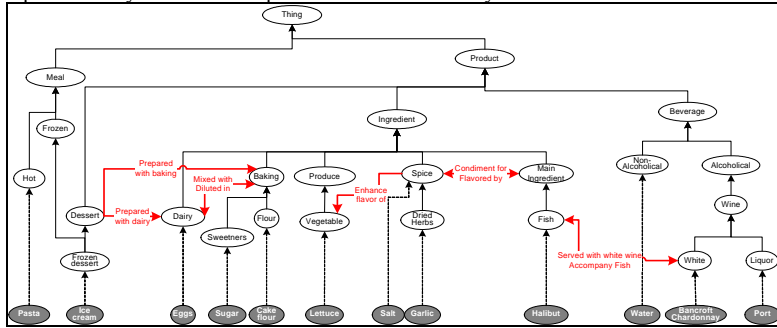


Fig. 1 Example of a knowledge base for alimentary products and meals

If we use the notation presented above, we can define the ontology as $\mathcal{O} := \{\mathcal{C}, \mathcal{R}, \mathcal{H}^{\#}, rel, \mathcal{A}^{\#}\}$. With $\mathcal{C} := \{Thing, Product, Beverage, Non-Alcoholic, Alcoholic, Wine, White wine, Liquor, Ingredients, Dairy, Baking, Sweeteners, Flour, Main ingredients, Fish, Produce, Vegetable, Spice, Dried herbs, Dessert, Frozen dessert, Meal, Frozen, Hot\}$, with *Thing* the concept, from whom all other concepts descend. The set of relations $\mathcal{R} := \{Condiment for, Flavored by, Enhance flavor of, Mixed with, Diluted in, Prepared with baking, Served with white wine, Accompany fish\}$. The taxonomy $\mathcal{A}^{\#} := \{(Meal, Thing), (Frozen, Meal), (Hot, Meal), (Product, Thing), (Beverage, Product), (Non-Alcoholic, Product), (Alcoholic, Product), (Wine, Alcoholic), (White wine, Wine), (Ingredient, Product), (Dairy, Ingredient), (Baking, Ingredient), (Sweeteners, Baking), (Flour, Baking), (Ingredient, Product), (Fish, ingredient), (Produce, Ingredient), (Vegetable, Produce), (Spice, Ingredient), (Dried herbs, Spice), (Dessert, Product), (Frozen dessert, Dessert), (Frozen dessert, Frozen)\}$. Finally, since there is no axiom, *rel* is defined by $\{Condiment for (Spice, Main Ingredient), Flavored by (Vegetables, Spice), Enhance flavor of (Main Ingredient, Spice), Prepared with baking (Prepared dessert, Baking), Served with white wine (Fish dish, White wine), Accompany fish (White wine, Fish)\}$.

The knowledge base $\mathcal{KB} := \{\mathcal{O}, \mathcal{I}, inst, instr\}$, with \mathcal{O} defined as above, $\mathcal{I} := \{Sugar, Eggs, Cake flour, Port, Bancroft Chardonnay, Water, Lettuce, Halibut, Salt, Garlic\}$ and $inst := \{(Sugar, Baking), (Eggs, Dairy), (Cake flour, Flour), (Port, Liquor), (Bancroft Chardonnay, White wine), (Water, Non-alcoholic (Lettuce, Vegetable), (Halibut, Fish), (Salt, Spice), (Garlic, Dried herbs), (Pasta, Hot), (Ice cream, Frozen dessert)\}$. Note that there is no relation instantiation.

From these, and above all, from the picture on Fig. 1, it is easy to understand that the ontology has two main concepts: *Product* and *Meal*. Moreover, among products *Ingredient*, *Dish* and *Beverage* are distinguishable. Each of these concepts has some sub-concepts, with particular emphasis to *Frozen dessert*, which derives from two distinct concepts: *Dessert* and *Frozen*.

Most of the instances in the corresponding knowledge base do not correspond to real products (with brand and model) due to the necessity of simplifying the picture.

4. The *Onto4AR* framework

Despite the developments on the creation of ontologies and knowledge representation, the large majority of mining projects does not make use of these well-formed conceptual models and despise most of the existing knowledge. On the other side, the improvements on the algorithms for mining association rules, least has contributed to incorporate background knowledge on the discovery process.

The balance between discovering unknown information and manageable quantities of patterns is far from being reached. But, as been shown, the incorporation of constraints (both in the form of interestingness measures and content constraints) is one of the few effective approaches to achieve that goal.

The *Onto4AR* framework, proposed in this paper, is a significant step in that pathway, since it defines a formal environment for incorporating background knowledge into the pattern mining process, that is independent on the problem domain, or the nature of the data. The framework is centered in a precisely definition of constraint and the problem of constrained pattern mining, based on the notions of ontology and knowledge base, presented before.

Next, the problem of mining association rules in the presence of background knowledge is defined.

4.1 Problem Statement

Consider a set of items \mathcal{I} , and an *itemset* \mathcal{A} as a subset of \mathcal{I} ($\mathcal{A} \subseteq \mathcal{I}$), and a *transaction* as an itemset that occurred in a particular instant of time. Additionally, consider that the \mathcal{I} set corresponds to the set of instances of a knowledge base ($\mathcal{KB} := \{\mathcal{O}, \mathcal{I}, inst, instr\}$), that references an ontology \mathcal{O} . Suppose that *inst* is defined, which means that the concept implemented by each instance is known. The problem of mining association rules in the context of the *Onto4AR* framework can now be stated as:

Given a set of instances I from a specific knowledge base $\mathcal{KB} = \{\mathcal{O}, \mathcal{I}, inst, instr\}$, and a set of transactions D , find all rules of the form $\mathcal{A} \Rightarrow \mathcal{B}$, where \mathcal{A} and \mathcal{B} are disjoint itemsets of \mathcal{I} that may occur in the transactions of D , and the itemsets \mathcal{A} and \mathcal{B} satisfy a set of constraints $\mathcal{C}_{\mathcal{O}}$ defined over the ontology \mathcal{O} .

Note that this definition only differs from the usual one, because both support and confidence are constraints, included in the set of constraints $\mathcal{C}_{\mathcal{O}}$. Also note that with this statement, there is no imposition on the relation between \mathcal{A} and \mathcal{B} (beside that they do not contain repeated items and all of them belong to the set of instances), neither on the number of times that they occur together or the confidence of the rule. All depends on the constraints imposed by $\mathcal{C}_{\mathcal{O}}$. As in the rest of literature (see [13]), in the context of the *Onto4AR* framework, a *constraint* is a predicate on the powerset of the set of items \mathcal{I} , which means, that it is a function $c: 2^{\mathcal{I}} \rightarrow \{true, false\}$. An itemset

\mathcal{S} is said to satisfy c , if and only if, $c(\mathcal{S})$ is *true*. In this manner, a constraint imposes some condition over the elements of the itemset, or the relations among them. These predicates can establish either qualitative or quantitative conditions. Indeed, at a first glance, two different categories of constraints can be distinguished: *interestingness constraints* (also known as *interestingness measures*) and *content constraints*.

4.2. Interestingness Measures

Interestingness measures are constraints that impose quantitative conditions over the set of items in the rule, like the number of times that the set of items are transacted together, or the novelty introduced by the discovery of the rule. Formally, an interestingness measure is a composed function $f=f_{\theta} \circ g$ [$f(x)=g(f_{\theta}(x))$], with $g: 2^{\mathcal{I}} \rightarrow \mathbb{R}$ and $f_{\theta}: \mathbb{R} \rightarrow \{true, false\}$, defined by the comparison of its argument with θ , a threshold value.

A special interestingness measure is the *existential constraint*, usually known as *support*. For the rule $A \Rightarrow B$, it is simply defined by substituting g with the function *frequency* and f_{θ} with the function \geq_{θ} . Where the function *frequency* counts the number of transactions in the dataset that contain both the items of A and B, and the function \geq_{θ} that returns *true* if and only if its argument is greater than or equal to θ , and *false* otherwise. Another usual interestingness measure is *confidence*, which in the *Onto4AR* framework can be defined by instantiating the g function by the conditional probability and the function f_{θ} again by \geq_{θ} . With the definition of these two interestingness measures, the problem of mining association rules as initially proposed [1] can be addressed in the *Onto4AR* framework, without any difficulty, since the set of items and the set of instances are the same things.

4.3. Content Constraints

Although interestingness constraints play a fundamental role on pattern mining, they only capture the knowledge about some quantity that is significant for the specific business. As such, they are not able to represent any other knowledge about the business domain. In order to go beyond this difficulty, content constraints introduce the ability of imposing that items in the rule have some specific characteristics. These characteristics can be selected among the ones represented in the domain ontology for the mining problem.

Formally, a *content constraint* can be defined as a predicate $c_c: 2^{\mathcal{I}} \rightarrow \{true, false\}$ that impose some qualitative condition on the items of its argument, expressed based in the domain knowledge represented in the ontology \mathcal{O} . For the rest of this section, consider the knowledge base $\mathcal{KB}:=\{\mathcal{O}, \mathcal{I}, inst, instr\}$, with \mathcal{O} the ontology defined by the tuple $\{\mathcal{C}, \mathcal{R}, \mathcal{N}^c, rel, \mathcal{L}\}$.

The first kind of content constraint that can be defined in this framework is the item constraint. As defined by Srikant [16], an item constraint is a ‘‘Boolean expression over the presence or absence of items in the rule’’. And in order to be able to use abstract concepts to define the scope of item constraints in the new context,

some additional notions are needed: parent, child, descendant and ancestor, like in that work [16]. In the *Onto4AR* context, *child* is a predicate defined over the concepts in the ontology \mathcal{O} , and *child*(x, y) is *true* if and only if $\mathcal{H}^c(x, y)$ is defined in the ontology. *Parent* is the inverse of *child*, and then *parent*(x, y) is *true* if and only if *child*(y, x) is *true*. The predicate *descendant* can be defined in a recursive way:

$$\text{descendant}(x,y) \Leftrightarrow \text{child}(x,y) \vee [\exists z \in \mathcal{C} : \text{child}(x,z) \wedge \text{descendant}(z,y)],$$

which means that either x is child of y or exists a third concept z that is parent of x , and is descendant of y , simultaneously. With these definitions, it is now possible to define item constraints in a generic form. In the new context, an *item constraint* is a content constraint $c_{\mathcal{S}}$, implemented by the predicate $\in_{c_{\mathcal{S}}}$ and then it only returns *true* if some items of its argument belong to the user-defined subset \mathcal{S} of \mathcal{I} . Let an itemset $X \in 2^{\mathcal{I}}$, it is said that

$$X \text{ satisfies an item constraint } c_{\mathcal{S}} \Leftrightarrow \exists x \in X : x \in \mathcal{S}$$

Note that it does not deal with the absence of some items, and the specification of \mathcal{S} has to be entered manually by the user, as it was with the specification of the Boolean expression. This specification can be done either by enumerating all accepted instances, or by describing its elements properties. For example, using the ontology described on section 3, the item constraint defined over the set $\{x \in \mathcal{I} : \text{inst}^{-1}(x) = \text{Liquor}\}$ will only accept patterns that include some liquors (with inst^{-1} the inverse function of *inst*). Additionally, consider the notion of n^{th} *cousin*. An item is a n^{th} *cousin* of another item, if they are connected by a n degree collateral relation. More specifically,

$$\forall x, y \in \mathcal{I}; n \in \mathbb{N} : n^{\text{th}} \text{cousin}(x, y, n) = \text{true} \Leftrightarrow \begin{cases} \text{descendant}(x, \text{parent}(y)), & \text{if } n = 0 \\ n^{\text{th}} \text{cousin}(x, \text{parent}(y), n - 1), & \text{otherwise} \end{cases}$$

For example, x is a 0^{th} *cousin* of y if it descends from y 's parent (x is either his “brother” or his “nephew”); x is a 1^{st} *cousin* of y , if it descends from y 's grandparent (one degree of collaterality); x is a 2^{nd} *cousin* of y , if it descends from y 's great-grandparent (two degrees of collaterality), and so on.

We can say that an item X is *at least an* n^{th} -*cousin* of another item Y if there is some $m \leq n$ such X is an m^{th} -*cousin* of Y . With these definitions, it is possible to define a new category of constraints – named *family constraints*. As expected, these constraints are implemented by the n^{th} -*cousin* and *at least* n^{th} -*cousin* predicates. From this category of constraints, it is possible to identify the *same-family* $_n$ constraint, which accepts an itemset if all their items share a common ancestor and all of them are at least n^{th} *cousins* to each other. This means that

$$X \text{ satisfies same-family}_n \Leftrightarrow \forall x, y \in X \exists m \in \mathbb{N} : x \neq y \wedge m \leq n \wedge n^{\text{th}} \text{cousin}(x, y, m)$$

Again, using the ontology described on section 3, the itemset $\{\text{Cake flour}, \text{Sugar}\}$ satisfies the *same-family* constraint with $n=0$, since they share the same parent, and $\{\text{Port}, \text{Water}\}$ satisfies the *same-family* constraint with $n=2$, since *Water* descends from the great-grandparent of *Port* (*Beverage*). But $\{\text{Port}, \text{Salt}\}$ does not satisfy the constraint *same-family* with $n=2$, only for $n=3$. Note that every itemset satisfies the *same-family* constraint for unlimited n , since all concepts descend from the concept *Thing*. Similarly, the *close-family* $_n$ constraint accepts an itemset if every item in the set satisfies the *at least an* n^{th} -*cousin* predicate with another item in the set.

$$X \text{ satisfies close-family}_n \Leftrightarrow \forall x \in X \exists y \in X \exists m \in \mathbb{N} : x \neq y \wedge m \leq n \wedge n^{\text{th}} \text{cousin}(x, y, m)$$

In order to have an itemset that satisfies the close-family constraint and does not satisfy the same-family constraint, the itemset has to contain some instantiation of a concept that derives from more than one concept, this is, it has to have multiple parents. In previous ontology, only itemsets that contain instantiations of *Frozen dessert*, can satisfy the close-family constraint. For example the itemset $\{Salt, Ice\ cream, Pasta\}$ satisfies the close-family constraint with $n=1$, because *Salt* is a 1st-cousin of *Ice cream*, and *Ice cream* is a 1st-cousin of *Pasta*. Note that *Salt* and *Pasta* are only 3rd-cousins. Note that both item constraints and family constraints can be defined using the taxonomy on the ontology, so they are examples of *taxonomical constraints*. It is important to note that item constraints are non-anti monotonic, but both the same-family and close-family constraints are.

The incorporation of an ontology in the mining process, instead of a simple taxonomy, allows for the definition of other constraints that extend the notion of item constraint. Those new constraints are based on the relations among concepts, described in the ontology. Consider some additional predicates over items and itemsets with two or more items (consider an itemset $X \in 2^{\mathcal{I}}$), needed for defining those new constraints. First, two items are *related* if there is a known non-taxonomical relation r (with $r \in \mathcal{R}$) among their classes. Formally,

$$\forall x, y \in \mathcal{I}. \text{related}(x, y) = \text{true} \Leftrightarrow \exists r \in \mathcal{R}. r(\text{inst}^1(x), \text{inst}^1(y)).$$

Based on this notion, an itemset is *weakly connected* if all its items are related with at least another item in the set. This is,

$$X \text{ is weakly connected} \Leftrightarrow \forall x \in X \exists y \in X: x \neq y \wedge \text{related}(x, y)$$

The itemset $\{Halibut, Garlic, Lettuce\}$ is weakly connected in the context of the ontology described on section 3, since there is the relation *Condiment for* from *Spice* to *Main Ingredient*, and another *Enhance flavor* from *Spice* to *Vegetable*. On the contrary, the itemset $\{Halibut, Port\}$ is not accepted since there is no relation between *Fish* and *Liquor*. An itemset can either be *softly connected*. This happens when there is a chain of relations among all its items. More specifically

$$X \text{ is softly connected} \Leftrightarrow \exists x, y \in X: x \neq y \wedge \text{related}(x, y) \wedge X \setminus \{x\} \text{ is softly connected}$$

In the context of the same ontology and knowledge base, the itemset $\{Halibut, Garlic, Lettuce\}$ is also softly connected, because there is a relation from *Halibut* to *Garlic* (*Flavored by*) and from *Garlic* to *Lettuce* (*Enhance flavor*), which means there is a chain of relations among the items.

A third predicate is the strongly connected. An itemset is *strongly connected* if all its items are related to all the other items in the set, which can be stated as follows:

$$X \text{ is strongly connected} \Leftrightarrow \forall x, y \in X: x \neq y \Rightarrow \text{related}(x, y)$$

In the same context, an itemset that is strongly connected is $\{Halibut, Bancroft\ Chardonnay\}$, and the combinations among *Bakery* and *Dairy* instances such as $\{Eggs, Sugar\}$. Note that for example, the itemset $\{Garlic, Halibut, Bancroft\ Chardonnay\}$ is not strongly connected only because there is no relation from *White wine* and *Spice*.

These three predicates can be used to define new content constraints: *weak*, *soft* and *strong* constraints, respectively. All of them can be categorized as non-taxonomical constraints, since they are based on non-taxonomical relations. From these, only weakly connected is non-anti monotonic.

Beside the existence of taxonomical and non-taxonomical constraints, the *Onto4AR* framework provides another important artifact – the composition of content constraints. Again the composition can be weak or strong.

The *weak composition* of two content constraints, say f and g , results in a new content constraint, $f \circ g$, which only accepts itemsets accepted by at least one of the original constraints. Consider an itemset $X \in 2^{\mathcal{I}}$,

$$f \circ g : 2^{\mathcal{I}} \rightarrow \{true, false\} \text{ with } f \circ g(X) = true \Leftrightarrow f(X) = true \vee g(X) = true$$

On the other hand, the *strong composition* of two content constraints, say f and g , results in a new content constraint, $f \circ g$, which only accepts itemsets accepted by both original constraints. Consider an itemset $X \in 2^{\mathcal{I}}$,

$$f \circ g : 2^{\mathcal{I}} \rightarrow \{true, false\} \text{ with } f \circ g(X) = true \Leftrightarrow f(X) = true \wedge g(X) = true$$

It is important to note that the composition of any pair of the non-taxonomical constraints proposed above, always result in the most restrictive of the constraints. For example, composing weak and soft constraints will result on the last one, since all itemsets accepted by a soft constraint are also accepted by weak constraints. Indeed, the soft constraint is more restrictive than the weak constraint, and the strong constraint is more restrictive than the soft constraint. Interesting compositions can result from combining non-taxonomical with taxonomical constraints. For example, the composition of weak and item constraints would only accept itemsets that have some specific items and, simultaneously, all their items are related with at least another item in the set. In addition to the composition of content constraints, the *Onto4AR* framework allows the definition of other constraints, either as subclasses of proposed ones or by defining new kinds of constraints with different scopes. Examples of the last ones are temporal and structural constraints, used to identify relevant temporal patterns and to identify structured patterns, like frequent sequences or graphs. In order to extend the framework, it is only need to stat the new constraints in accordance to the definitions presented above. In this manner, the categorization of the constraints proposed in this paper, illustrated in Fig. 2, is just a small picture of the potentialities introduced by the *Onto4AR* framework.

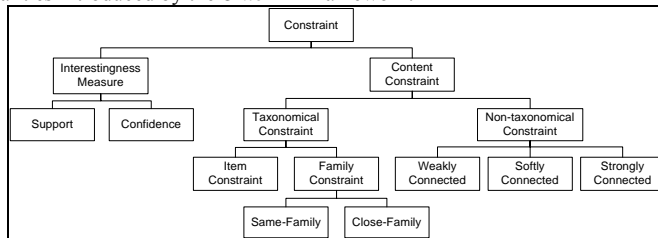


Fig. 2. Categorization of proposed constraints

4.4. Algorithms

As in the rest of pattern mining area, efficient algorithms for identifying accepted patterns is mandatory in this framework. The algorithm for this purpose should be

similar to general algorithms for pattern mining, which means that it would filter patterns that satisfy the imposed constraint in an internal pruning step. In this manner, the processing time would be significantly reduced due to the reduction on the explosion of discovered patterns.

It is important to note that this pruning step wouldn't require the navigation over the entire ontology. For each potential pattern (call it *candidate*) we need to verify if the candidate satisfies the constraint, this means, verifying if its items are linked among them, which only requires to follow the relations among their concepts in the ontology.

The last issue refers to candidate generation, since some of proposed constraints (namely item and weakly-connected constraints) are non-anti monotonic. The simplest solution is to extend frequent k-patterns with frequent items, like it is done in constrained sequential pattern mining [11] and [3]. Fig. 3 presents the pseudo-code for an apriori-based algorithm able to deal with *Onto4AR* constraints.

```

Procedure Onto4AR-apriori (Dataset D, Constraint C)
   $L_1 := \{\text{frequent items in } D\}$ 
   $k := 2$ 
  while ( $L_k \neq \emptyset$ ) do
     $C_k' := \text{join}(L_k, L_1, C)$  //candidate generation
     $C_k := \{p \in C_k' : p.\text{satisfies}(C)\}$  //candidate pruning
     $L_k := \{\text{frequent patterns in } C_k\}$  //support-based pruning
     $k := k + 1$ 
  return  $\cup_k L_k$ 

```

Fig. 3. Algorithm for mining pattern in the *Onto4AR* framework

It is important to note that recent work on sequential pattern mining using more complex algorithms and constraints (deterministic finite and push-down automata) [3] demonstrates experimentally that the reduction on the number of discovered patterns has more impact in the performance, than the verification of the constraint satisfaction for each candidate. Moreover, the *Onto4AR* just use the correspondence between instances and concepts, and the ontology structure – relations among concepts (axioms and instance relations are not used). In this manner, the ontology almost corresponds to an acyclic graph, which makes the verification process similar to the ones referred.

5. Conclusions

The recent advances in the area of knowledge representation makes possible to represent background knowledge, in an effective way, through the construction of ontologies. Since one of the main drawbacks of data mining, in general, and of pattern mining, in particular, is to ignore existing domain knowledge, with those advances, it is time to surpass that feature. This paper presents a framework that incorporates background knowledge in the core of the mining process, by using domain ontologies and by defining a set of constraints above them, which can guide the discovery process. This guidance can be conducted by the expert domain or simply by the user, who establishes the level of constraints to be applied to the process.

The great advantages of the new framework are related to its extensibility, since it is possible to define other constraints based on domain ontologies. In particular, proposed constraints are universal and can be applied to any problem domain, given that there is some corresponding domain ontology.

References

1. Agrawal, R., Imielinsky, T., Swami, A. Mining Association Rules between Sets of Items in Large Databases. In *Proc. ACM SIGMOD Conf. Management of Data*, (1993) 207-216
2. Antunes, C., Oliveira, A., Using Context-Free Grammars to Constrain Apriori-Based Algorithms for Mining Temporal Association Rules. In *Proc. Workshop on Temporal Data Mining* (2002).
3. Antunes, C., Oliveira, A.L., Constraint Relaxations for Discovering Unknown Sequential Patterns. In *Knowledge Discovery in Inductive Databases*, Springer, (2005), 11-32
4. Bayardo, R.J., Agrawal, R., Gunopulos, D., Constraint-Based Rule Mining in Large, Dense Databases. In *Proc. Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, (1999) 145-154
5. Bayardo, R.J., The Many Roles of Constraints in Data Mining. In *SIGKDD Explorations*, vol. 4, nr. 1 pp. i-ii (2002).
6. Bonchi, F. *Frequent Pattern Queries: Language and Optimizations*. PhD Thesis, Università di Pisa, 2003.
7. Boulicaut, J.-F., Klemettinen, M., Mannila, H., Querying inductive databases: a case study on the MINE RULE operator. In *Proc. Conf. Principles and Practice of Knowledge Discovery*, Springer, (1998). 194-202
8. Češpivová, H., Rauch, J., Vojtěch, S., Kejkula, M. and Toměčková, M., Roles of Medical Ontology in Association Mining CRISP-DM Cycle. In *Proc. Workshop on Knowledge Discovery and Ontologies*. (2004). 1-12
9. Corcho, O., Gómez-Pérez, A., A Roadmap to Ontology Specification Languages. In *Proc. Int'l Conf. Knowledge Engineering and Knowledge Management*, Springer. (2000). 80-96
10. Euler, T., Scholz, M., Using Ontologies in a KDD Workbench. In *Proc. Workshop Knowledge Discovery and Ontologies* (2004). 103-108
11. Garofalakis, M.N., Rastogi, R., Shim, K., SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proc. Very Large Databases Conference*. (1999), 223-234
12. Maedche, A., *Ontology Learning for the Semantic Web*, Kluwer Publishers, (2002).
13. Pei, J., Han, J., Lakshmanan, L.V.S. Mining Frequent Itemsets with Convertible Constraints. In *Proc. IEEE Int'l Conf. Data Engineering* (2001). 433-332.
14. Pei, J., Han, J., Constrained frequent pattern mining: a pattern-growth view. In *SIGKDD Explorations*, vol. 4, nr. 1. ACM Press, (2002). 31-39
15. Wang, J., Han, J., Pei, J., CLOSET+: Searching the Best Strategies for Mining Frequent Closed Itemsets. In *Proc. Int'l Conf. Knowledge Discovery and Data Mining*. ACM Press, (2003). 236-245
16. Srikant, R., Vu, Q., Agrawal, R. Mining Association Rules with Item Constraints. In *Proc. Int'l Conf. Knowledge Discovery and Data Mining*. ACM Press, (1997). 67-73
17. Vanzin, M., Becker, K., Exploiting Knowledge Representation for Pattern Interpretation. In *Proc. Workshop on Knowledge Discovery and Ontologies*. 2004. 61-72

Iterative Constraints in Support Vector Classification with Uncertain Information

Jianqiang Yang and Steve Gunn

School of Electronics and Computer Science, University of Southampton
Building 1, Highfield Campus, Southampton, SO17 1BJ, UK
{jy03r, srg}@ecs.soton.ac.uk

Abstract. This paper proposes a new iterative approach of input uncertainty classification, which incorporates input uncertainty information and exploits adaptive constraints extracted from uncertain inputs statistically and geometrically to extend the traditional support vector classification (SVC). Kernel functions can be implemented by a novel kernelized formulation to generalize this proposed technique to non-linear models and the resulting optimization problem is a second order cone program (SOCP) with a unique solution. Results demonstrate how this technique has an improved performance and is more robust than the traditional algorithms when uncertain information is available.

Key words: SVC, iterative constraints, uncertain, kernel functions

1 Introduction

Uncertain information associated with data is often ignored in traditional machine learning algorithms. Many approaches attempt to model any uncertainty in the form of additive noise on the target, which can be effective for simple models. However, for more complex non-linear models and where a richer description of anisotropic uncertainty in the input space is available, these approaches can suffer. For instance, the traditional support vector classification (SVC) can only accommodate isotropic uncertainty information in the input space.

Recent advances in machine learning methods have seen significant contribution from kernel-based approaches. These have many advantages, including strong theory and convex optimization formulation. Support vector machines (SVMs) are one approach that have been extended to incorporate uncertain data. Many other algorithms are also focusing on input uncertainty classification by implementing their own constraints. The rest of the paper explores an extension to SVC to provide a more robust algorithm, which enables uncertain information in the inputs to be incorporated iteratively into the constraints. The resulting algorithm is formulated as a second order cone programming (SOCP) optimization problem with adaptive constraints driven by the uncertainties.

The paper is organized as follows: section 2 presents the input uncertainty formulation for the classification task. In Sect.3, the dual problem is derived by introducing noise-specific covariance information as additional constraints

and the approach is extended to non-linear classification by a novel kernelized formulation. It is then shown how these geometric and statistical characteristics can be extended to generate two more efficient iterative algorithms in Sect. 4. In Sect. 5, some kernel functions are introduced along with the experimental results of these new approaches to compare with traditional algorithms.

2 Input Uncertainty Classification

Definition 1. Let $\mathcal{D} = \{z_i, y_i\}, i = 1, \dots, l$ denote the observed inputs, where $y_i \in \{-1, +1\}$, $z_i \in \mathbb{R}^n$ and $z_i \sim \mathcal{N}(x_i, \mathbf{M}_i)$, in which \mathcal{N} is a Gaussian distribution with mean $x_i \in \mathbb{R}^n$ and covariance $\mathbf{M}_i \in \mathbb{R}^{n \times n}$.

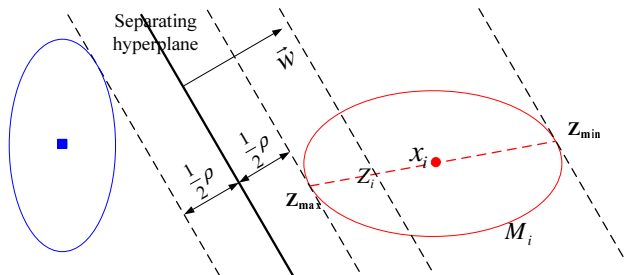


Fig. 1. The classification of Gaussian uncertainties in the input space ($n = 2$).

The input uncertainties in Definition 1 are shown in Fig. 1 [10], in which the ellipsoids represent the Gaussian distributions of the input uncertainties, ρ represents the margin between the closest edges of the ellipsoids to the optimal hyperplane, z_{\max} and z_{\min} represent those points, at which the hyperplanes parallel to the optimal hyperplane are tangent to the edges of the ellipsoids.

2.1 Geometric Interpretation

Let $\mathcal{E}(\mathbf{A}, \mathbf{a}) \subseteq \mathbb{R}^n$ denote an ellipsoid, $\mathbf{A} \in \mathbb{R}_+^{n \times n}$, $\mathbf{a} \in \mathbb{R}^n$ and $\mathcal{E}(\mathbf{A}, \mathbf{a}) := \{x \in \mathbb{R}^n \mid (x - \mathbf{a})^T \mathbf{A}^{-1} (x - \mathbf{a}) \leq 1\}$. Setting $\mathbf{Q}_i = \mathbf{M}_i^{1/2}$, according to Definition 1 and the theorem [4], which shows that every ellipsoid is the image of the unit ball around zero under a bijective affine transformation, we have

$$\begin{aligned} \max_w \{w^T z_i \mid z_i \in \mathcal{E}(\mathbf{M}_i, x_i)\} &= \max_w \{w^T \mathbf{Q}_i \mathbf{Q}_i^{-1} z_i \mid \mathbf{Q}_i^{-1} z_i \in \mathbf{Q}_i^{-1} \mathcal{E}(\mathbf{M}_i, x_i)\} \\ &= w^T \frac{1}{\sqrt{w^T \mathbf{M}_i w}} \mathbf{M}_i w + w^T x_i, \end{aligned} \tag{1}$$

where the optimal result is $\mathbf{z}_{\max} = \mathbf{x}_i + \frac{1}{\sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}} \mathbf{M}_i \mathbf{w}$, $\mathbf{z}_{\min} = \mathbf{x}_i - \frac{1}{\sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}} \mathbf{M}_i \mathbf{w}$ and consequently, $\mathbf{z}_i = \mathbf{x}_i + r \frac{1}{\sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}} \mathbf{M}_i \mathbf{w}$, $-1 \leq r \leq 1$. Figure 1 shows how \mathbf{z}_i follows its Gaussian distribution represented by an ellipsoid \mathbf{M}_i as r varies. The next section introduces a theorem on probabilistic linear inequalities, which enables a formulation of the extended uncertainty information.

2.2 Minimax Probability Machine

The minimax probability machine (MPM)[7] is a recent method introduced for pattern classification. MPM chooses a discriminative approach to minimize the misclassification probability of the future inputs without the prior knowledge of the distributions of inputs. MPM uses a theorem [1], which provides the stronger upper optimal bounds in probability than the result in Chebyshev's inequality, to derive an approach, transforming the probability inequality $\inf_{\mathbf{u} \sim (\bar{\mathbf{u}}, \Sigma_{\mathbf{u}})} \Pr\{\mathbf{a}^T \mathbf{u} \leq h\} \geq \alpha$ to the following expression without the probability in the inequality.

$$h - \mathbf{a}^T \bar{\mathbf{u}} \geq \kappa(\alpha) \sqrt{\mathbf{a}^T \Sigma_{\mathbf{u}} \mathbf{a}} \quad \text{where } \kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}. \quad (2)$$

where $\mathbf{a}^T \mathbf{u} \leq h$ represents a hyperplane, $\mathbf{a}, \mathbf{u} \in \mathbb{R}^n$, $h \in \mathbb{R}$, α is the inferior probability of the correctly classified inputs, $\bar{\mathbf{u}} \in \mathbb{R}^n$ is the mean and $\Sigma_{\mathbf{u}} \in \mathbb{R}^{n \times n}$ is the covariance of the inputs of a class.

2.3 Statistical Approach

First, we exploit theorem [1] to develop a formulation in which the probability of misclassification is minimized under this extended uncertainty description. We can use (2) to extend SVC to incorporate the uncertainty in Definition 1,

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \geq \kappa(\alpha) \sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}, \quad (3)$$

where $\kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}$. Furthermore, we can consider a Gaussian model for the uncertainties on the inputs, we can transform $\inf_{\mathbf{u} \sim (\bar{\mathbf{u}}, \Sigma_{\mathbf{u}})} \Pr\{\mathbf{a}^T \mathbf{u} \leq h\}$ to

$$\inf_{\mathbf{z}_i \sim \mathcal{N}(\mathbf{x}_i, \mathbf{M}_i)} \Pr\{-y_i \mathbf{w}^T \mathbf{z}_i \leq y_i b - 1 + \xi_i\} = \Phi \left(\frac{y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i}{\sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}} \right) \geq \alpha, \quad (4)$$

where $\Phi(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^v \exp(-s^2/2) ds$ is the cumulative distribution function for a standard normal Gaussian distribution. Since $\Phi(v)$ is monotone increasing, we can write (4) as:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \geq \Phi^{-1}(\alpha) \sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}. \quad (5)$$

We can generate SVC constraints independent of the distributions of the uncertain inputs by combining (3) and (5),

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \geq r \sqrt{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}, \quad (6)$$

where $r \in \mathbb{R}$ is the probability confidence. Although the distribution is assumed to be Gaussian in this paper, (6) provides us with a way to exploit other distributions of the uncertain inputs when this information is available.

2.4 Missing Features

In some cases, uncertainties may be partially unknown [8]. Consider the Gaussian distribution $\mathcal{N}(\mathbf{x}_i, \mathbf{M}_i)$ introduced in Definition 1 where some features of \mathbf{x}_i are missing, and as the result, only part of the covariance matrix \mathbf{M}_i is known. We then can extend the Gaussian approximation from [3] to estimate the unknown components. Let \mathbf{x}_{ik} and \mathbf{x}_{im} denote the known features and the missing features of \mathbf{x}_i , and $\mathbf{x}_i = [\mathbf{x}_{ik}^T, \mathbf{x}_{im}^T]^T$. Introducing function $f(\mathbf{z}) = \mathbf{z}$, $\mathbf{z} \in \mathbb{R}^j$, $j = 1, \dots, n$, we then obtain

$$\begin{aligned} f(\mathbf{x}_i) &\sim \mathcal{N}(\mu_{\mathbf{x}_i}, \Sigma_{\mathbf{x}_i}) = \mathcal{N}(\mathbf{x}_i, \mathbf{M}_i) \\ \text{Cov}(f(\mathbf{x}_p), f(\mathbf{x}_q)) &= \text{Cov}(\mathbf{x}_p, \mathbf{x}_q) \end{aligned} \quad (7)$$

where $\mathbf{x}_p, \mathbf{x}_q \in \mathbb{R}^m$, $m \leq n$ are parts of \mathbf{x}_i , and the covariance matrix $\text{Cov}(\mathbf{x}_p, \mathbf{x}_q)$ is part of \mathbf{M}_i . Then the distribution $p(\mathbf{x}_{ik}, \mathbf{x}_{im} | \mathbf{x}_{ik})$ is a Gaussian distribution with mean $[\mathbf{x}_{ik}^T, \mathbf{x}_{im}^T]^T$ and covariance matrix \mathbf{M}_i , which is shown as follows:

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k \end{bmatrix}, \quad (8)$$

where \mathbf{K} , \mathbf{k} and k are the covariance matrices of \mathbf{x}_{ik} , \mathbf{x}_{ik} and \mathbf{x}_{im} , and \mathbf{x}_{im} respectively. The predictive distribution of \mathbf{x}_{im} is

$$\mathbf{x}_{im} | \mathbf{x}_{ik} \sim \mathcal{N}(\mu(\mathbf{x}_{ik}), \Sigma(\mathbf{x}_{ik})) \quad (9)$$

where $\mu(\mathbf{x}_{ik})$ and $\Sigma(\mathbf{x}_{ik})$ are obtained by:

$$\begin{aligned} \mu(\mathbf{x}_{ik}) &= \mathbf{x}_{im} + \mathbf{k}^T \mathbf{K}^{-1} (\mathbf{x}_k - \mathbf{x}_{ik}) \\ \Sigma(\mathbf{x}_{ik}) &= k - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} \end{aligned} \quad (10)$$

An iterated algorithm to approximate the missing features is shown as follows:

Algorithm 1. Missing Features Approximation

When \mathbf{x}_{ik} converge to \mathbf{x}_k , $\mu(\mathbf{x}_{ik})$ and $\Sigma(\mathbf{x}_{ik})$ are what we want:

1. Initialize $\mathbf{x}_i = [\mathbf{x}_{iko}^T, \mathbf{x}_{imo}^T]^T$ and \mathbf{M}_i ;
2. Let $\mathbf{x}_k = \mathbf{x}_{ik}$, $\mathbf{x}_{ik} = \mathbf{x}_{iko}$ and $\mathbf{x}_{im} = \mathbf{x}_{imo}$, obtain \mathbf{K} , \mathbf{k} , k from \mathbf{M}_i and compute (10);
3. Recompute and collect the new value of \mathbf{x}_i and \mathbf{M}_i by using the completed data, allocate the new value to \mathbf{x}_{iko} and \mathbf{x}_{imo} , then return to step 2;

3 Uncertainty Support Vector Classification

In this section we derive the primal and dual formulations of the input uncertainty classification which is named uncertainty support vector classification (USVC).

3.1 Linear Case

The primal problem of USVC is obtained by introducing the constraints from (6) as:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^l \xi_i \\ \text{s.t.} \quad & r \|\mathbf{M}_i^{1/2} \mathbf{w}\| \leq y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \\ & r \geq 0 \quad \xi_i \geq 0 \quad i = 1, \dots, l \end{aligned} \quad (11)$$

where r needs to be set in advance in the optimization. USVC is a second order cone program (SOCP). Following the Lagrangian method, we have $\mathbf{w} = \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^l (\mathbf{M}_i^{1/2})^T \boldsymbol{\beta}_i$, in which the dual variables, $\boldsymbol{\beta}_i \in \mathbb{R}^n$, $i = 1, \dots, l$ control the influence of the covariance matrices describing the distributions of the uncertain inputs, while the dual variables, $\alpha_i \in \mathbb{R}$, $i = 1, \dots, l$, behave in a similar manner to the SVC. When $r = 0$, meaning that the probability of the examples being correctly classified is set to 0.5 in the classification, or $\mathbf{M}_i = \mathbf{0}$, $i = 1, \dots, l$, meaning that there is no uncertainty information, USVC degenerates to the SVC solution.

3.2 Extension to Non-linear Case

Generally in real life, data to be classified will require a non-linear separation. USVC needs to be extended to non-linear case. Let $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$ denote a mapping of the data of input space \mathbb{R}^n to a high dimensional space \mathbb{R}^m . Since the mapped ellipsoid of an uncertain input in the input space may lead to an irregular shape in the feature space, the Taylor Series expansion is introduced and can be expanded based on the inputs \mathbf{x}_i and \mathbf{x}_j . Set $\phi(\mathbf{x}_i) = \mathbf{z}_i = [z_{i1}, \dots, z_{im}]^T \in \mathbb{R}^m$ and $\mathbf{x}_i = [x_{i1}, \dots, x_{in}]^T \in \mathbb{R}^n$, we have $\phi(\mathbf{x}_j) = \phi(\mathbf{x}_i) + \mathbf{J}(\mathbf{x}_j - \mathbf{x}_i) + O\left(\frac{1}{2} \frac{\partial^2 \mathbf{z}}{\partial \mathbf{x}^2} (\mathbf{x}_j - \mathbf{x}_i) + \dots\right)$ where \mathbf{J} is Jacobian matrix which is made up of the first order partial derivatives and can be used to approximately map a tiny line segment $\|\mathbf{x}_i - \mathbf{x}_j\|$ in the input space to $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|$ in the feature space. The Taylor series of $\phi(\mathbf{x}_j)$ can be simplified by ignoring the higher order partial derivatives, we have $\Delta\phi(\mathbf{x}_j) \simeq \mathbf{J}\Delta\mathbf{x}_j$. Furthermore, the expression can be extended to accommodate the geometric polygonal mapping of the input space, $\left[\Delta\phi(\mathbf{x}_1)^T \quad \dots \quad \Delta\phi(\mathbf{x}_l)^T \right]^T = \left[\Delta\mathbf{x}_1^T \quad \dots \quad \Delta\mathbf{x}_l^T \right]^T \mathbf{J}^T$, where $\Delta\phi(\mathbf{x}_i) \in \mathbb{R}^m$ and $\Delta\mathbf{x}_i \in \mathbb{R}^n$. The covariance matrix \mathbf{M}_i represents the i th uncertainty distribution of the input space. The tiny line segment $\Delta\mathbf{x}_i$ is related to $O(\mathbf{M}_i^{1/2})$, which can be represented as $\mathbf{M}_i = (\mathbf{M}_i^{1/2})^T \mathbf{M}_i^{1/2} \sim \Delta\mathbf{x}_i \Delta\mathbf{x}_i^T$. Therefore, the related geometric mapping of \mathbf{M}_i in the feature space can be formed by the Jacobian matrix

$$\phi(\mathbf{M}_i^{1/2}) = \mathbf{M}_i^{1/2} \mathbf{J}^T \quad (12)$$

According to the definition $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ can be seen as independent functions during the derivatives over the kernel function,

so the first and second derivatives of the kernel function can be retrieved by the inner product of the mapping function ϕ and its derivative. Therefore, the optimization problem of USVC is given by:

$$\begin{aligned} \max_{\alpha, \beta} \quad & \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{2} \left(\right. \\ & \sum_{i=1}^l \sum_{j=1}^l \alpha_i y_i \left[\frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_j} \right]^T (\mathbf{M}_j^{1/2})^T \beta_j + \sum_{i=1}^l \sum_{j=1}^l \alpha_j y_j \beta_i^T \mathbf{M}_i^{1/2} \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} \\ & \left. + \sum_{i=1}^l \sum_{j=1}^l \beta_i^T \mathbf{M}_i^{1/2} \frac{\partial^2 K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} (\mathbf{M}_j^{1/2})^T \beta_j \right) \\ \text{s.t.} \quad & \sum_{i=1}^l \alpha_i y_i = 0 \quad \|\beta_i\| \leq r \alpha_i \quad r \geq 0 \quad 0 \leq \alpha_i \leq C \quad i = 1, \dots, l \end{aligned} \quad (13)$$

where $\frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} = \frac{\partial \phi(\mathbf{x}_i)}{\partial \mathbf{x}_i} \cdot \phi(\mathbf{x}_j)$, $\left[\frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_j} \right]^T = \phi(\mathbf{x}_i) \cdot \frac{\partial \phi(\mathbf{x}_j)}{\partial \mathbf{x}_j}$ and $\frac{\partial^2 K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} = \frac{\partial \phi(\mathbf{x}_i)}{\partial \mathbf{x}_i} \cdot \frac{\partial \phi(\mathbf{x}_j)}{\partial \mathbf{x}_j}$.

4 Minimax Probability Support Vector Classification

4.1 Total Support Vector Classification

In 2004, [2] proposed a formulation of support vector classification called total support vector classification (TSVC), which can accommodate uncertainties in the inputs. Without loss of generality, $\delta \|\mathbf{w}\|$ in [2] can be transformed to $\|\mathbf{M}_i^{1/2} \mathbf{w}\|$ in this paper under the definition of the uncertain inputs in Definition 1. As the result, the constraints of the problem of TSVC becomes to

$$-\|\mathbf{M}_i^{1/2} \mathbf{w}\| \leq y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i . \quad (14)$$

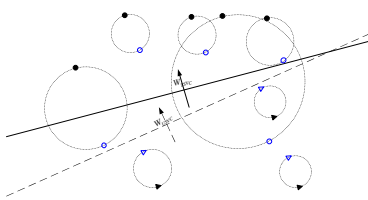


Fig. 2. Geometric Interpretation of TSVC and USVC.

Equation (14) actually can be derived from (6) with $r = -1$. Figure 2 shows the original figure from [2] with the USVC solution at the setting of $r = 1$ superimposed to illustrate the different geometric interpretation between TSVC and USVC. In Fig. 2, the ellipses with circle points on and the ellipses with triangle points on represent the examples from two different classes where $y_i = \pm 1$. In the classification, TSVC uses the farthest points (solid points) in the distributions of the uncertain inputs as a reference to obtain the optimal hyperplane (\mathbf{w}_{TSVC} , solid

line), while USVC uses the nearest points (hollow points) in the distributions of the uncertain inputs to the optimal hyperplane (\mathbf{w}_{USVC} , dashed line) to compute the classifier. According to the characteristics of support vectors and convex optimization, it can be proved that TSVC is neither a convex optimization nor a SOCP problem.

4.2 Adaptive Constraints in Uncertainty Support Vector Classification

Although TSVC is not a convex optimization, input uncertainty classification can benefit from the characteristics of its low probability confidence. An iterative algorithm is proposed here to combine TSVC and USVC to achieve a better overall performance. This new method is termed adaptive uncertainty support vector classification (AUSVC), in which, the misclassified inputs decrease their probability confidence to accommodate the misclassification in each step, while the probability confidence of correctly classified inputs remain unvaried. Therefore, individual probability confidence r_i is chosen here for each uncertainty instead of selecting a general probability confidence r for all uncertainties in USVC. In order to remain convex in AUSVC, the optimization problem selects the probability confidence $r_i \geq 0$ and $r \in \mathbb{R}$, so that the probability confidence of some misclassified inputs finally achieve 0 when AUSVC converges. Geometrically, AUSVC is a method of searching the optimal points from the nearest points to the central points of different input uncertainties respectively.

The optimization problem of AUSVC can be rewritten from (13) by simply replacing $\|\beta_i\| \leq r\alpha_i$ with $\|\beta_i\| \leq r_i\alpha_i$. Its iterative algorithm is shown below:

Algorithm 2. AUSVC

Initialize $r_i = 1, i = 1, \dots, l$, repeat the following three steps until $r_{inew} = r_i, i = 1, \dots, l$:

1. Fix $r_i, i = 1, \dots, l$ to the current value, solve (13) for the parameters α_j, β_j , and b ;
2. Substitute the obtained parameters α_j, β_j, b and the training inputs (\mathbf{x}_i, y_i) into

$$g(\mathbf{x}_i, y_i) = \text{sgn} \left[y_i \left(\sum_{j=1}^l \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + \sum_{j=1}^l \beta_j^T \mathbf{M}_j^{1/2} \frac{\partial K(\mathbf{x}_j, \mathbf{x}_i)}{\partial \mathbf{x}_j} + b \right) \right] \quad (15)$$

respectively to determine whether the inputs are misclassified, $g(\mathbf{x}_i, y_i) < 0$ or correctly classified, $g(\mathbf{x}_i, y_i) \geq 0$. If correctly classified, their probability confidence r_i remain unchanged, otherwise, a predefined positive scalar (normally 0.1) is deducted from its probability confidence r_i , the changed probability confidence is saved in r_{inew} ;

3. If $r_{inew} = r_i$, the optimal results of α_i, β_i , and b are achieved, otherwise, $r_i = r_{inew}$ and return to step 1;

4.3 Minimax Probability Support Vector Classification

MPM not only derives a discriminative method to classify inputs without prior knowledge of the distributions of inputs, but also as the result introduces a measure to compare the different algorithms. This measure is named minimax probability error (MPE), which adds up together the possible maximal misclassified probability of every input uncertainty, which is $\sup_{\mathbf{z}_i \sim (\mathbf{x}_i, \mathbf{M}_i)} \Pr\{y_i(\mathbf{w}^T \mathbf{z}_i + b) \leq 0\} = \frac{1}{1+d_i^2}$, and $d_i^2 = \inf_{y_i(\mathbf{w}^T \mathbf{z}_i + b) \leq 0} (\mathbf{z}_i - \mathbf{x}_i)^T \mathbf{M}_i^{-1} (\mathbf{z}_i - \mathbf{x}_i)$. To incorporate $r_i < 0$ into the optimization problem to provide lower probability confidence, we introduce MPE to generate a new optimization problem as follows:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \sum_{i=1}^l \frac{1}{1+d_i^2} \\ \text{s.t. } d_i^2 = \quad & \begin{cases} \frac{(\mathbf{w}^T \mathbf{x}_i + b)^2}{\mathbf{w}^T \mathbf{M}_i \mathbf{w}} & y_i(\mathbf{w}^T \mathbf{x}_i + b) > 0 \\ 0 & y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 0 \end{cases} \end{aligned} \quad (16)$$

Geometrically, when the uncertainty is misclassified by a hyperplane, then $d_i = 0$ and its possible maximal misclassified probability on this hyperplane is 1. Otherwise, d_i is equal to the distance between the center \mathbf{x}_i and the hyperplane, and its possible maximal misclassified probability on this hyperplane is $\frac{1}{1+d_i^2}$.

Since this approach is motivated by MPM and SVC, we call this proposed algorithm minimax probability support vector classification (MPSVC). However, the contribution of the maximal misclassified probability of each input is different in the optimization problem. Inspired by [6], in which the prior probability of each class is introduced in the optimization when the worst-case accuracies for two classes are not the same, additional parameters θ_i are introduced with different values set adaptively for the inputs misclassified and correctly classified to formulate a cost sensitive optimization problem. Introducing the previous results from USVC, $\mathbf{w} = \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^l (\mathbf{M}_i^{1/2})^T \beta_i$, and the previous results from (6) into (16), kernel functions can be exploited to extend MPSVC to non-linear input uncertainty classification. Equation (16) can be rewritten as:

$$\begin{aligned} \max_{\alpha_i, \beta_i, b, r_i} \quad & \sum_{i=1}^l \theta_i r_i \\ \text{s.t. } y_i \left(\sum_{j=1}^l \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + \sum_{j=1}^l \beta_j^T \mathbf{M}_j^{1/2} \frac{\partial K(\mathbf{x}_j, \mathbf{x}_i)}{\partial \mathbf{x}_j} + b \right) & \geq r_i \\ \left\| \sum_{j=1}^l \alpha_j y_j \mathbf{M}_i^{1/2} \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} + \sum_{j=1}^l \mathbf{M}_i^{1/2} \frac{\partial^2 K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} (\mathbf{M}_j^{1/2})^T \beta_j \right\| & \leq C \\ r_i & \geq D_i \quad i = 1, \dots, l \end{aligned} \quad (17)$$

where $\theta_i \in \mathbb{R}$ are penalty coefficients, C is a constant, $r_i \in \mathbb{R}$ is the probability confidence of the i th uncertainty, the lower bound of r_i is provided by $D_i \in \mathbb{R}$,

which can be set positive or negative depending on the uncertain inputs and the optimal hyperplane. For those which are misclassified by the hyperplane, D_i will be set negative to decrease the probability confidence of those inputs, otherwise, D_i will be positive. In general, D_i extends the geometric search area of the optimal solution from $r_i \geq 0$ in AUSVC to both $r_i \geq 0$ and $r_i < 0$ in MPSVC while this optimization problem remains a convex and SOCP problem. MPSVC provides a way of searching input uncertainty classification solutions in a specific scope by implementing the optimal solution of AUSVC as additional constraints. Its iterative algorithm is shown below:

Algorithm 3. MPSVC

When MPE converges, MPSVC achieves its optimum.

1. Run Algorithm 2 of AUSVC for the parameters α_j , β_j , and b ;
2. Substitute the obtained parameters α_j , β_j , b and the training inputs (\mathbf{x}_i, y_i) into (15) to determine whether the inputs (\mathbf{x}_i, y_i) are misclassified. If correctly classified, their D_i are set to equal to the average value of d_i of these correctly classified inputs, otherwise, $D_i = -1$, use (16) to calculate MPE_{old} ;
3. Fix $\theta_i = 1$, $i = 1, \dots, l$, solve (17) for the parameters α_j , β_j , and b ;
4. Substitute the parameters α_j , β_j , b and the training inputs (\mathbf{x}_i, y_i) into

$$h(\mathbf{x}_i, y_i) = \text{sgn} \left[y_i \left(\sum_{j=1}^l \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + \sum_{j=1}^l \beta_j^T \mathbf{M}_j^{1/2} \frac{\partial K(\mathbf{x}_j, \mathbf{x}_i)}{\partial \mathbf{x}_j} + b \right) - 1 \right] \quad (18)$$

to determine whether the inputs (\mathbf{x}_i, y_i) are misclassified ($h(\mathbf{x}_i, y_i) < 0$). If correctly classified, $\theta_i = \theta_i/\lambda$, otherwise, $\theta_i = \theta_i \times \lambda$, λ is a scalar (normally 10);

5. Use (16) to calculate MPE_{new} . If $\|\text{MPE}_{\text{new}} - \text{MPE}_{\text{old}}\| < \epsilon$, MPE converges, otherwise, $\text{MPE}_{\text{old}} = \text{MPE}_{\text{new}}$, return to step 4;

5 Experimental Comparisons

Besides the traditional measure, the number of misclassified centers of the uncertainties (NMC) and the new introduced measure MPE, a new parameter which measures the number of misclassified nearest edges of the uncertainties to the optimal hyperplane is introduced as an additional performance measure in the experiments. This new measure is named as the number of misclassified edges of the uncertainties (NME). NME provides a new viewpoint which includes the uncertainties in the performance comparison of different algorithms.

We reproduced the experiments from [2] to test the performance of these algorithms by following the exact prescription described in [2]. In the experiments with toy data sets in two dimensions, $l = 100$ training examples \mathbf{x}_i were generated from the uniform distribution on $[-5, 5]^2$ by the random number generator. Binary classification problems were created with original separating function $x_1^2 + x_2^2 = Ra^2$, where $Ra \in [3, 4]$ and $\mathbf{x} = [x_1, x_2]^T$. All the algorithms were

trained with the quadratic kernel $(\mathbf{x}_i^T \mathbf{x}_j)^2$ in the experiments. The input vectors \mathbf{x}_i were contaminated by Gaussian noise with mean $[0, 0]$ and covariance matrix $\Sigma = \delta_i \mathbf{I}$ where δ_i was randomly chosen from $[0.1, 0.8]$. \mathbf{I} is a 2×2 identity matrix. We randomly chose 0.1*l* from the first 0.2*l* examples after examples were ordered in an ascending order of their distances to the original separating hyperplane. For these 0.1*l* examples, noise was generated using a larger δ_i randomly drawn from $[0.5, 2]$. In total 16 input datasets are generated with the results of the classification of the 6th and 14th dataset shown in Fig. 3 and the 8th and 12th dataset shown in Fig. 4. The experimental code is based on the MATLAB SVM toolbox [5] and MATLAB optimization toolbox SeDuMi [9].

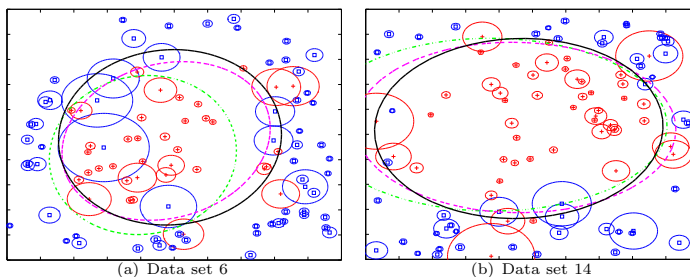


Fig. 3. Experimental comparisons. The dashed line represents USVC ($r = 1$), the dash-dot line represents AUSVC and MPSVC is represented by the solid line.

Figure 3 shows the experimental comparison of 6th and 14th dataset between USVC, AUSVC and MPSVC. Both AUSVC and MPSVC perform better than USVC under all the measures. However, AUSVC can be easily influenced at the area with low input density (see Fig. 3(a)) or the area where one class dominates the other class (see Fig. 3(b)). With the advantages of its characteristics, MPSVC can recover from the adversarial distribution introduced by uncertain inputs.

Because the probability confidence is relatively low from the strategy of the iterative algorithms in the optimization problems, AUSVC and MPSVC generally outperform in the experiments with respect to the other methods, especially when the large uncertainties from one class cross the original boundary to dominate the areas of low input density of the other class. USVC performs worse than the other methods by choosing the nearest edges of these dominant uncertainties geometrically which causes the optimal hyperplane of USVC to be stretched to these dominant uncertainties (see Fig. 4(a)). On the contrary, choosing the farthest edges of the uncertainties geometrically also causes some mistakes in the classification. In Fig. 4(d), USVC performs better than TSVC in areas of low input density of both classes. With the iterative algorithm and individually decreasing probability confidence r_i , AUSVC can generally achieve an improved performance by reducing the influence from some dominant uncertainties in the

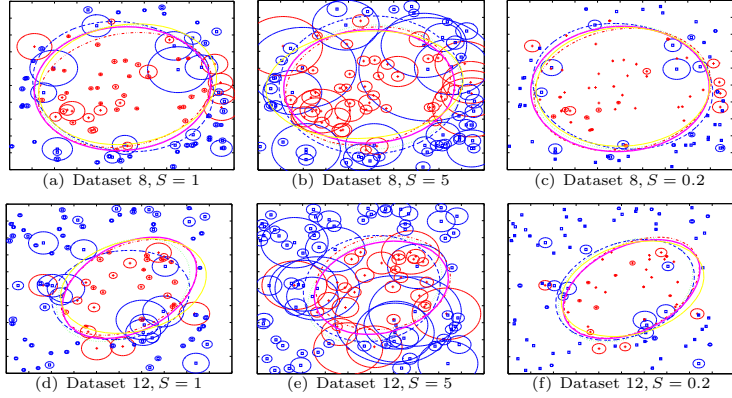


Fig. 4. Experimental Comparisons. Different algorithms are compared in the experiments. The covariance matrices \mathbf{M}_i are varied as $\mathbf{M}'_i = S\mathbf{M}_i$, and the inputs \mathbf{x}_i are kept fixed. The dotted line represents SVC, the dashed line represents TSVC, the thin solid line represents USVC ($r = 1$), the dash-dot line represents AUSVC and MPSVC is represented by thick solid line.

classification (see Fig. 4(a) 4(d)). With lower probability confidence and larger penalty coefficients for misclassified uncertainties, MPSVC can even achieve a better performance than AUSVC by recovering from the adversarial distribution introduced by uncertain inputs (see Fig. 4(d)). The experimental comparison of NMC, NME and MPE of these approaches on the 16 datasets are shown in Fig. 5,

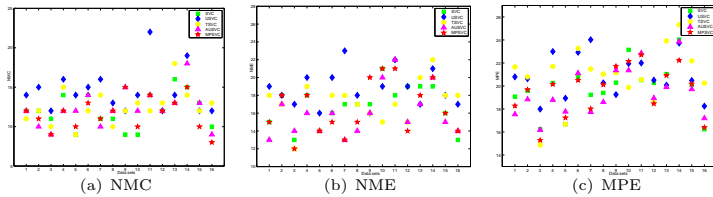


Fig. 5. Experimental comparisons of the different measures of SVC, USVC ($r = 1$), TSVC, AUSVC and MPSVC.

The influence from the uncertainties with different sizes is compared in the experiments as well. Let $S \in \mathbb{R}$ denote the size factor, the varied uncertainties \mathbf{M}'_i come from $\mathbf{M}'_i = S\mathbf{M}_i$. When the uncertainties are amplified by S (see

Fig. 4(b) 4(e), $S = 5$), the performance of USVC deteriorates in Fig. 4(b), and even more in Fig. 4(e), in which USVC can not accommodate large uncertainties during the optimization. AUSVC and MPSVC can produce superior results with its varied probability confidence r_i and penalty coefficients θ_i in iterative algorithms. In the case when the uncertainties decrease (see Fig. 4(c) 4(f), $S = 0.2$), AUSVC achieves the best performance around $r = 1$, which is very close to the performance of USVC. As S decreases, AUSVC and USVC converge to SVC, and in the limit ($S = 0$), the information of the uncertainties is unavailable, AUSVC and USVC degenerate to SVC. In this case, $\beta_i = \mathbf{0}$ and $\|\beta_i\| \leq \alpha_i$ can be rewritten as $\alpha_i \geq 0$ in (13).

6 Conclusions

A new approach, USVC, has been proposed here for classifying data with uncertain information which has been implemented in this paper as additional constraints in the optimization. Along with USVC, two novel iterative algorithms AUSVC and MPSVC have been designed by using adaptive constraints come from the noise-specific covariance information. These methods have been extended to non-linear models by a novel formulation to accommodate kernel functions. Experimental comparisons show that these iterative approaches based around adaptive constraints have greatly improved the performance of input uncertainty classification.

References

1. Bertsimas, D., Popescu, I., Sethuraman, J.: Moment problems and semidefinite optimization. *Handbook of Semidefinite Optimization*. (2000) 469–509
2. Bi, J., Zhang, T.: Support vector classification with input data uncertainty. *Advances in Neural Information Processing Systems*. **16** (2004)
3. Girard, A., Rasmussen, C.E., Quiñero-Candela, J., Murray-Smith, R.: Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. *Advances in Neural Information Processing Systems*. **15** (2003)
4. Grötschel, M., Lovász, L., Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization*. 2nd corr. ed.. Springer-Verlag. ISBN: 0-38-756740-2 (1993) 66–73
5. Gunn, S.R.: Support vector machines for classification and regression. Technical Report. University of Southampton. (1998)
6. Huang, K., Yang, H., King, I., Lyu, M.R., Chan, L.: The minimum error minimax probability machine. *Journal of Machine Learning Research*. **5** (2004) 1253–1286
7. Lanckriet, G.R.G., El Ghaoui, L., Bhattacharyya, C., Jordan, M.I.: A robust minimax approach to classification. *Journal of Machine Learning Research*. **3** (2002) 555–582
8. Shivaswamy, P.K., Bhattacharyya, C., Smola, A.J.: Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*. **7** (2006) 1283–1314
9. Sturm, J.F.: Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*. **11-12** (1999) 625–653
10. Yang, J., Gunn, S.R.: Input uncertainty in support vector machines. *Machine Learning Workshop*, Sheffield, UK, (2004)

Multitarget Polynomial Regression with Constraints

Aleksandar Pečkov, Sašo Džeroski, and Ljupčo Todorovski

Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract. The paper addresses the task of multi-target polynomial regression, i.e., the task of inducing polynomials that can predict the value of more than one numeric variable. As in other learning tasks, we face the problem of finding an optimal trade-off between the complexity of the induced model and its predictive error. We propose a minimal description length scheme for multi-target polynomial regression, which includes coding schemes for polynomials and their predictive errors on training data. The proposed MDL scheme is implemented in an algorithm for polynomial induction that can also take into account language constraints, i.e., constraints on terms to be included in the induced polynomials. We empirically compare the multi-target model with the multiple single target models. The results of the experiments show that there is no loss in predictive performance when using multi-target models as compared to multiple target models and that fewer equation structures are considered in the former case.

1 Introduction

Regression models are used to predict the value of a dependent numeric variable from the values of independent (predictor) variables. Commonly used regression models include linear regression and regression trees [1]. While the linear regression method tries to find a global model of the data (a linear equation), regression trees are piecewise models that partition the data space into a number of sub-spaces and induce a simple constant or linear model in each of them. While linear models tend to be oversimplistic, regression trees can sometimes overfit the training data. In this paper we address the task of polynomial regression, i.e., the task of inducing polynomial equations from numeric data that can be used to predict the value of a numeric variable. Polynomials can also overfit the data. Namely, it is well known that a data set of n points can be perfectly interpolated (and often overfitted) with a polynomial of $(n - 1)$ -th degree.

In order to address the problem of overfitting, different approaches to model selection have been proposed in the literature [3] (pages 193-222). Each approach tries to find an optimal trade-off between the complexity of the induced model and its predictive error and thus avoid overfitting. The minimal description length (MDL) principle is one of them. Following the MDL principle, the quality of a model is estimated by combining the estimate of the model complexity and the predictive error the model makes on the training data. The

complexity of the model and the error are measured in terms of the number of bits necessary for encoding them.

In this paper we first address the task of polynomial regression and present a MDL encoding scheme for single target polynomial model. We compare this encoding scheme with an Akaike like ad-hoc encoding scheme. Also we compare the better to liner regression, regression trees, and model trees. Then we extend our approach for multi-target regression by extending our encoding scheme for multi-target models. Finally, we empirically compare the multi-target approach with the single target approach.

2 Polynomial Regression

The task of polynomial regression, is to induce a polynomial equation from numeric data that can predict the value of a numeric variable.

Every polynomial over variables x_1, x_2, \dots, x_n can be written in the form:

$$P = c_0 + \sum_{i=1}^m c_i \cdot T_i$$

where $T_i = \prod_{j=1}^n x_j^{a_{i,j}}$, c_i , $i = 1..m$ and c_0 are constants and $c_i \neq 0$. We say T_i is a *term* or *monomial* in P . The length of P is defined as $Len(P) = \sum_{i=1}^m \sum_{j=1}^n a_{i,j}$, while the size of P is $Size(P) = m$; and the degree of P is $Deg(P) = Max_{i=1}^m \sum_{j=1}^n a_{i,j}$.

An example polynomial equation is $P = 1.2x^2y + 3.5xy^3 + 5xz + 2$. This equation has size 3 (it has three terms), degree 4 (the maximal term degree is 4) and length 9.

Ciper [7] (Constrained Induction of Polynomial Equations for Regression) is a beam search algorithm that heuristically searches through the space of candidate polynomial equations for the ones that fit the data best.

The top-level outline of Ciper algorithm is shown in Table 1. First, the beam is initialized either with the simplest polynomial equation $P = c$, or with a minimal polynomial that follows the given constraints (the constraints will be described below). In every search iteration, a set of polynomials is generated from the beam using a refinement operator. The coefficients before the terms are fitted using linear regression. For each of the polynomials, the value of the minimal description length (MDL) heuristics is calculated. At the end of the iteration, the equations with smallest MDL values are retained in the beam. The evaluation stops when the refinement operator can not generate new equations or when the content of the beam was unchanged in the last iteration. Such situation occurs when every polynomial that is generated in the last iteration has worse MDL estimate than the polynomials already in the beam.

The refinement operator increases the length of an equation by one, either by adding a first degree term or by multiplying an existing term by a variable (Figure 1). Starting with the simplest equation, and iteratively applying the refinement operator, all polynomial equations can be generated.

Table 1. A top-level outline of the CIPER algorithm. Q and Q_r are sets of equations (the beam).

```

procedure CIPER(Data, InitialPol, Constraints)
  InitialPol = FITPARAMETERS(InitialPol, Data)
   $Q = \{InitialPol\}$ 
  repeat
     $Q_r =$  refinements of (some) equation structures in  $Q$ 
    foreach equation structure  $E \in Q_r$  do
       $E =$  FITPARAMETERS( $E$ , Data)
    endfor
     $Q = \{\text{best } b \text{ equations from } Q \cup Q_r\}$ 
  until  $Q$  unchanged during the last iteration
  print  $Q$ 

```

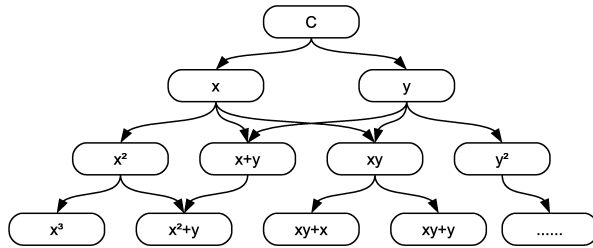


Fig. 1. The Ciper refinement operator.

Ciper can take into account two types of constraints on the form of the induced polynomial equations:

- *Language constraints* specify structural bounds on the polynomial structures considered during search: P_L and/or P_U . These constraints specify that every candidate equation P should be a super-polynomial of P_L , while P_U should be super-polynomial of every P . A polynomial P is a subpolynomial from a polynomial P' if for every term T in P exists a term T' in P' such that the degree of every variable in T is larger than or equal to the degree of the same variable in T' .
- *Complexity constraints* constrain the complexity of a polynomial with specifying the upper bound for the polynomial length, degree, or size.

Our approach is based on the single target Ciper algorithm can take into account some constraints during induction, most noticeably, language constraints. This capability is preserved in the multi-target version of Ciper. In the next section, we describe the heuristic function used to evaluate every candidate equation.

3 Minimal Description Length

Following the minimal description length (MDL) principle, among the number of candidate models, we select the one that represents a good trade-off between model's predictive error its complexity. The MDL principle combines two ideas (or assumptions) about relation between learning and data compression:

- regularities in the data can be used to compress the data, i.e., the more regularities there are, the more the data can be compressed;
- the more we are able to compress the data, the more we have learned about the data.

Thus, complexity of the model can be estimated as its ability to compress data: larger the compression, smaller the complexity of the obtained model. More specifically, MDL estimate of the model quality is composed of two components:

$$MDL(H) = L(H) + L(D|H),$$

where the first component $L(H)$ corresponds to the length of the encoding of model (hypothesis) H , while the second one $L(D|H)$ is the length of the description of the data when encoded using the model H .

3.1 Encoding polynomial structure

In order to encode the structure of polynomials, we follow the refined MDL approach [10]. We first partition the space of candidate models into subgroups \mathcal{H}_c of models with equal complexity c . A particular model $H \in \mathcal{H}_c$ can be then encoded using $N = \log|\mathcal{H}_c|$ (note that \log stands for binary logarithm) bits, where $|\mathcal{H}_c|$ denotes the number of models in the class \mathcal{H}_c .

In case of polynomials, we are going to partition the space of candidate polynomial structures in classes at several levels. At the highest level, we'll group together the candidate polynomials with same length l and same number of terms (size) m . We'll refer to these classes as $G(m, l)$; for example $G(1, 1)$ contains polynomial structures with one linear term, while $G(1, 2)$ contains polynomial structures with only one term of second degree. Note that $m \leq l$. On the second level, we partition each $G(m, l)$ in subgroups with fixed term degrees $G'(a_1, a_2, \dots, a_m)$. Polynomials in this subgroup have m terms with degrees $a_1 \geq a_2 \geq \dots \geq a_m$. Note that $\sum_{i=1}^m a_i = l$. Now we have to calculate how many sub-groups G' there are in a single $G(m, l)$ group and also calculate how many polynomial structures there are in each $G(a_1, a_2, \dots, a_m)$ group.

The number $|G'(a_1, a_2, \dots, a_m)|$ can be easily calculated using a procedure roughly depicted in Figure 2. Given the degree of the first term a_1 we have to choose a_1 variables from the set $\{x_1, x_2, \dots, x_n\}$ where variables can appear in the selection more then once. Thus, the number of possibilities for the first term equals the number of combinations with repetition where we select a_1 elements from a set of given n . This number equals $\binom{n+a_1-1}{a_1}$. Continuing the same reasoning for all m terms, gives us the number of possible structures in

$G'(a_1, a_2, \dots, a_m)$ to be $\prod_{i=1}^m \binom{n+a_i-1}{a_i}$. However, if there are several a_i values that are equal, we will encounter the same term many times, which means that the above formula over-estimates the number of possible structures. The remedy is to divide the number with the factorial of repetitions observed in the tuple. For example, when dealing with case $G'(5, 5, 3, 2, 2, 2)$, we have to divide with $2!3!$, since value 5 is repeated twice and value 2 is repeated three times. Note also that each multiplicative term decreases by 1 for each degree value repetition (see Figure 2).

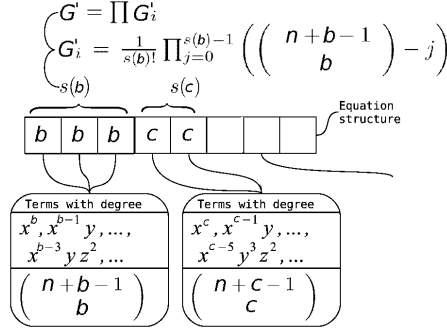


Fig. 2. Calculating the number of polynomial structures in $G'(a_1, a_2, \dots, a_m)$. At the bottom, we have the sets of terms (one with terms of degree b and one with terms of degree c). In the middle layer, they are combined in equation structures, where $s(b)$ and $s(c)$ denote the number of repetitions of b and c values respectively.

Having the number of equation structures in each G' group, we now turn to a problem of calculating the number of G' groups within each $G(m, l)$. The size of G grows according to the recursive formula $|G(m, l)| = |G(m-1, l-1)| + |G(m, l-m)|$. The first additive term corresponds to the cases when the G' groups contain linear terms (there is a_i with value 1), while the second corresponds to the cases when the G' groups where all $a_i > 1$. In the first case, when removing the linear term, we obtain polynomials with $m-1$ terms and length $l-1$. In the second case, we can remove one variable from all the terms, which leads to polynomials with same number of terms (m) and length $m-l$. Figure 3 depicts the relationship between G and G' classes of polynomial structures.

Now, having this partitioning and number of polynomials in each partition, we can decompose the code for each candidate polynomial in four components. First, we have to encode its length l and for this we need $\log(l) + 2\log(\log(l))$ bits (the second double logarithm term is necessary, since we do not know the magnitude of l in advance). Second, we encode the number of terms m , for which we need $\log(l)$ bits (remember that $m \leq l$). Third, we can identify a particular G' class within the class $G(m, l)$ using $\log(|G(m, l)|)$ bits. Finally, we identify the specific polynomial structure within G' using $\log(|G'(a_1, a_2, \dots, a_m)|)$ bits.

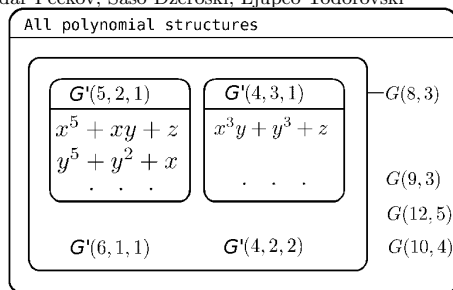


Fig. 3. General overview of the partitioning of polynomial structures. The small sets correspond to G' classes (e.g., the set $G'(5, 2, 1)$). In turn, we group them into a larger classes of structures G that have same length and size.

Putting these four components together gives us the final formula: $L(H) = 2\log(l) + 2\log(\log(l)) + \log(|G(n, l)|) + \log(G'(a_1, a_2, \dots, a_n))$, for number of bits necessary to encode the polynomial structure.

3.2 Encoding Data

Rissanen provides a formula for calculating the stochastic complexity of a model obtained using linear regression [5]:

$$W = \min_{\gamma} \left\{ (N - k) \log(\hat{\tau}) + k \log(\hat{R}) + (N - k - 1) \log\left(\frac{1}{N - k}\right) - (k - 1) \log(k) \right\}$$

where γ index goes through the all possible subsets of variables involved in the linear regression, k is the number of elements in γ , N is the size of the dataset, $\hat{\tau}$ is the maximum likelihood estimation of the model error, and $\hat{R} = \frac{1}{n} \hat{c}^T (X^T X) \hat{c}$ (where $\hat{c} = (X^T X)^{-1} X^T y$). The stochastic complexity of the model then is $2W$. Intuitively, this corresponds to the length of the code necessary to encode the errors of the linear regression model ($L(D|H)$) together with the constant parameters of the linear model. The later is closely related to the model error and thus can not be encoded separately, that is together with the model structure, which is what is mostly done when using (ad-hoc) MDL principle in machine learning algorithms. For further details, see [5].

4 Single-target Polynomial Regression

The task of single-target polynomial regression is to induce a polynomial equation from numeric data that can predict the value of a single numeric variable. Ciper original implementation used an ad-hoc MDL based heuristics function. In this section we will present the ad-hoc heuristics and compare it with the improved MDL heuristics (Section 3) on single target datasets.

4.1 Ad-hoc MDL

The ad-hoc MDL heuristic function is given by

$$MDL(P) = \text{len}(P) \cdot \log(m) + m \cdot \log(MSE(P))$$

where P is the polynomial equation being evaluated, $len(P)$ is its length, $MSE(P)$ is its mean squared error, and m is the number of training examples.

This evaluation function is based on the Akaike and Bayesian information criteria for regression model selection [3]. The second term of the ad-hoc MDL heuristic function measures the degree of fit of a given equation and the first term introduces a penalty for the complexity of the equation. With this penalty the MDL heuristic function introduces preference toward simpler equations.

4.2 Empirical evaluation for the heuristic functions

The main goal of the performed experiments is to evaluate the predictive performance of Ciper using the different heuristics described above. We compared them with the standard regression methods, implemented in the data mining suite Weka [9]. The performance of the methods is evaluated on fifteen data sets from the UCI Repository [4] and another publicly available collection of regression data sets [8]. These data sets have been widely used in other comparative studies.

In all the experiments presented here, we estimate the predictive performance on unseen examples using 10-fold cross validation. The predictive performance of a model M is measured in terms of relative root mean squared error ($RRMSE$). The Ciper algorithm that uses ad-hoc MDL heuristic will be referred to as ad-hoc Ciper, and the Ciper algorithm that uses the improved MDL heuristic will be referred with MDL Ciper.

The last two columns in Table 2 give the performance comparison between ad-hoc and improved MDL. It is noticeable that MDL Ciper performs better than ad-hoc Ciper. The statistical significance is tested using a paired t-test. If the p-value is smaller than 0.05 then we reject the null hypothesis, and conclude that the difference is statistically significant. The + sign in the table is used when the improvements we introduce perform significantly better and the - sign is used when they perform worse.

We found that ad-hoc Ciper never performs better and MDL Ciper performs better on six datasets. We can conclude that Ciper with MDL heuristics performs better than Ciper using ad-hoc heuristics. Ciper clearly outperforms linear regression, and performs much better than regression trees on more datasets. Also Ciper is comparable to model trees (Ciper was better on three and model trees on three datasets).

5 Multitarget Polynomial Regression

The task of multi-target polynomial regression is to induce a polynomial equation from numeric data that can predict the value of several numeric variables.

A multi-target polynomial model can be defined as:

$$P = C_0 + \sum_{i=1}^m C_i \cdot T_i$$

Table 2. Predictive performance in terms of relative root mean square error of commonly used regression methods implemented in Weka: linear regression (LR), model trees (MT), and regression trees (RT). Also comparison of ad-hoc Ciper and MDL Ciper.

Data set	LR	RT	MT	ad-hoc	mdl
2dplanes	0.5427 +	0.2272	0.2270	0.2270	0.2270
autoprice	0.4715 +	0.5426 +	0.3659	0.4128	0.3815
bank32nh	0.6858	0.7512 +	0.6739	0.8119 +	0.6751
basketball	0.7737 +	0.8850 +	0.7737	0.7784	0.7738
bodyfat	0.1643	0.3293 +	0.1557 -	0.2834	0.1663
cal-housing	0.6037	0.5177 -	0.4778 -	0.5903 +	0.5767
cpu-small	0.5371 +	0.2247 -	0.1738 +	0.4161 +	0.1628
elusage	0.4781 +	0.6560 +	0.4372	0.4009	0.4009
fried-delve	0.5265 +	0.3550 +	0.2780 +	0.2903 +	0.1996
house-8l	0.7869 +	0.6250	0.5929	0.6097	0.6123
housing	0.5281 +	0.5095 +	0.4286	0.4264	0.4184
kin-8nm	0.7661 +	0.6837 +	0.6093 +	0.8463 +	0.5570
mv	0.4309 +	0.0475 +	0.0131 -	0.0440 +	0.0214
pw-linear	0.4954 +	0.5640 +	0.3258	0.3301	0.3310
vineyard	0.7133	0.8617	0.7458	0.5254	0.6749

where $T_i = \prod_{j=1}^n x_j^{a_{i,j}}$ are the terms. Here $C_i, i = 1..m$ and C_0 are constant vectors (not constants) and $C_i \neq \mathbf{0}$. The number of coordinates of these vectors is the number of targets we want to predict.

An example of a multi-target polynomial equation is $P = (1, 2, 5) \cdot x^2y + (3, 5, 7) \cdot xy^3 + (2, 3, 10)$. It is equivalent to three single target polynomial equations $P_1 = 1 \cdot x^2y + 3 \cdot xy^3 + 2$; $P_2 = 2 \cdot x^2y + 5 \cdot xy^3 + 3$; $P_3 = 5 \cdot x^2y + 7 \cdot xy^3 + 10$, i.e $P = (P_1, P_2, P_3)$. This equation model predicts three targets.

In this way a single equation can be used for predicting more targets. The idea is that the complexity of this equation will be smaller than the complexity of a set of equations (one equation per each target). If the single target equations depend on the same term then we will need less to encode the multi-target model than the single target models. Also, in this case we may obtain a model that have better predictive capabilities because the risk of overfitting will be smaller.

The Ciper algorithm for multi-target polynomial regression goes just the same as in the single target case (Table 1). The data can be represented as a matrix M , where the number of rows is the number of instances, and the number of columns is the number of terms plus one (the first column is filled with ones). We calculate the coefficients C_i of the equation as

$$C = (M^T \cdot M)^{-1} \cdot (M^T \cdot Y)$$

where Y is the matrix of values we are trying to predict. We have introduced some optimizations for obtaining the coefficients. In this equation the multiplication is computationally expensive because of the large number of rows. If we have

terms T_1 , T_2 , T_3 and T_4 , such that $T_1 \cdot T_2 = T_3 \cdot T_4$, then the appropriate elements in matrices $M_{T_1, T_2}^T \cdot M_{T_1, T_2}$ and $M_{T_3, T_4}^T \cdot M_{T_3, T_4}$ are equal. We store all generated elements from the matrices $M^T \cdot M$. We use it later to calculate the matrices of the subsequently generated polynomials. Even more this matrices are the same for every target. This optimization considerably lowers the amount of calculations.

Notice that the language and the complexity constraints mentioned in Section 2 can equally be used in the multi-target case.

Because the structure of the polynomial hasn't changed, the complexity is the same for the multi-target case like in the single target case. Summing the stochastic complexities of the linear model for each target with the complexity of the structure we have the total complexity of the multi-target model.

6 Empirical Evaluation

6.1 Real datasets

The main goal of the performed experiments is to evaluate the predictive performance of Ciper on multi-targets datasets. The performance of the methods is evaluated on five data sets EDM, SIGMEA-REAL, and SIGMEA-SIM.

EDM dataset describes 154 actions taken by a human operator controlling two variables (target variables). It contains eight numeric attributes from which two are the target attributes. SIGMEA-REAL dataset collects 817 measurements of the rate of herbicide resistance of two lines of plants (target variables). It has eight attributes. SIGMEA-SIM dataset describes the effects of the individual field characteristics and cropping systems on the rate of pollen and the seed dispersal rate, (target variables). Dataset includes 10368 cases and it has 13 attributes.

A multi-target model is build for each dataset and a single target model for each of the targets. The predictive performance of a single target model is measured in terms of relative root mean squared error (*RRMSE*). The predictive performance of a multi-target model is presented as an array of the predictive performance of the appropriate single target models.

In all the experiments presented here, we estimate the predictive performance on unseen examples using 10-fold cross validation. The statistical significance is tested using a paired t-test. If the p-value is smaller than 0.05 then we reject the null hypothesis, and conclude that the difference is statistically significant. The + sign in the table is used when the improvements we introduce perform significantly better and the - sign is used when they perform worse.

The results suggest that there is no significant difference in the predictive capabilities whether we make single target models or multi target models.

We counted the number of equations generated with Ciper. It seems that the number of equations generated for multi-target Ciper is usually smaller than the total number of equations needed for building the single target models. The number of equations in the table 3 is averaged from the number of generated equations from the 10 folds.

Table 3. Comparison of predictive performance of multi-target and single-target Ciper in terms of relative root mean square error, and number of tested equations.

dataset	target	multi-target	single-target	num. eqs.
EDM	1	0.90703	0.86435	31152
	2	0.79134	0.75032	415
num. tested eqs.		458.0		3156.7
SIGMEA-SIM	1	0.07441	0.07509	139045.1
	2	0.06141	0.06158	157931.2
num. tested eqs.		202345.9		296976.3
SIGMEA-REAL	1	0.71846	0.71190	1497.2
	2	0.64948	0.63467	1812.6
num. tested eqs.		2265.6		3309.8

6.2 Using constraints in modeling chemical reactions

To illustrate the use of constraints in $\overline{\{x_5, \mathbf{x}_7\} \rightarrow \{x_1\}; \{x_1\} \rightarrow \{x_2, x_3\}}$ discovering dynamics, we address the task of reconstructing a partially specified $\overline{\{x_1, x_2, \mathbf{x}_7\} \rightarrow \{x_3\}; \{x_3\} \rightarrow \{x_4\}}$ network of chemical reactions. The part of the network given in bold is assumed to be unknown, except for the fact that x_6 and x_7 are involved in the network. This is a task of revising an equation-based model. A network of chemical reactions can be modeled with a set of polynomial differential equations. The reaction rate of a reaction is proportional to the concentrations of inputs involved (product, e.g. $x_5 \cdot x_7$). It influences the rate of change of all inputs (negatively) and all outputs (positively). The equation structures (left) / full equations (right), corresponding to the partial/full network, are given below.

Partial structure/Full equations	
	$x_1 = 0.8 \cdot x_5 \cdot x_7 - 0.5 \cdot x_1 - 0.7 \cdot x_1 \cdot x_2 \cdot x_7$
	$x_2 = 0.7 \cdot x_1 + 0.2 \cdot x_4 + 0.1 \cdot x_4 \cdot x_6 - 0.3 \cdot x_1 \cdot x_2 \cdot x_7$
$x_1 = -c \cdot x_1 + c \cdot x_5 - c \cdot x_1 \cdot x_2$	$x_3 = 0.4 \cdot x_1 + 0.3 \cdot x_1 \cdot x_2 \cdot x_7 - 0.2 \cdot x_3$
$x_2 = c \cdot x_1 + c \cdot x_4 - c \cdot x_1 \cdot x_2$	$x_4 = 0.5 \cdot x_3 - 0.7 \cdot x_4 \cdot x_6$
$x_3 = c \cdot x_1 + c \cdot x_1 \cdot x_2 - c \cdot x_3$	$x_5 = -0.6 \cdot x_5 \cdot x_7$
$x_4 = c \cdot x_3 - c \cdot x_4$	$x_6 = 0.2 \cdot x_4 - 0.8 \cdot x_4 \cdot x_6$
$x_5 = -c \cdot x_5$	$x_7 = -0.1 \cdot x_1 \cdot x_2 \cdot x_7 - 0.1 \cdot x_5 \cdot x_7$

The full equations were simulated for 1000 time steps of 0.01 from a randomly generated initial state (each variable randomly initialized in the interval (0,1)), thus providing a trace of the behavior of the 7 system variables over time.

Subsumption constraints can be used in a natural way. A partially specified reaction network gives rise to equations that involve subpolynomials of the polynomials modeling the entire network.

The knowledge of the partial network can be used to constrain the search through the space of possible equations. The polynomial structures in the equations for $x_1 \dots x_5$ in the partial network should be subpolynomials of the corresponding equations in the complete network. These subpolynomial constraints

were given to Ciper together with the behavior trace for all 7 variables. The subsumption constraint used in multi-target Ciper is $x_1 + x_3 + x_1 \cdot x_2 + x_4 + x_5$.

The generated multi-target model is

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} -0.50 & 0.00 & 0.00 & 0.00 & -0.70 & 0.81 \\ 0.69 & -0.01 & 0.20 & 0.10 & -0.29 & 0.08 \\ 0.40 & -0.20 & 0.00 & 0.00 & 0.30 & 0.00 \\ -0.01 & 0.50 & 0.00 & -0.70 & 0.00 & 0.03 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.61 \\ 0.00 & 0.00 & 0.20 & -0.80 & 0.00 & 0.00 \\ 0.01 & 0.00 & 0.00 & 0.00 & -0.10 & -0.12 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_3 \\ x_4 \\ x_4 \cdot x_6 \\ x_1 \cdot x_2 \cdot x_7 \\ x_5 \cdot x_7 \end{pmatrix}$$

Ciper successfully reconstructs the equations for the entire network, i.e., for each of the 7 system variables. Discovery without constraints, however, fails for 3 equations. However the predictive error of the generated models without constraints is less than 1 percent as shown in table 4.

Table 4. Comparison of predictive performance of multi-target and single-target Ciper in terms of relative root mean square error, and number of tested equations.

dataset	target	multi-target	single-target	num. eqs.
DYN1	1	+ 0.00019	0.00340	71.0
	2	0.00539	0.00546	957.0
	3	0.00038	+ 0.00035	942.5
	4	+ 0.00202	0.02143	71.0
	5	0.00067	0.00067	71.0
	6	0.00121	0.00119	71.0
	7	+ 0.00219	0.01706	71.0
num. tested eqs.		2087.4		2254.5
DYN2	1	0.00020	0.00019	508.2
	2	0.00270	0.00269	514.1
	3	0.00046	0.00046	485.9
	4	0.00065	+ 0.00053	465.0
	5	0.00021	0.00021	36.0
num. tested eqs.		1177.6		2009.2

7 Discussion

We have proposed an approach for multi-target polynomial regression, i.e., the task of inducing polynomials that can simultaneously predict the values of several numeric target variables. To this end, we extend the Ciper system for polynomial regression that has been recently modified to employ a principled MDL heuristic in its search for polynomial equations: The latter is empirically shown to perform better on a number of datasets. We adapt this MDL heuristic to the multi-target case as well. The multi-target approach can also take into account language constraints, i.e., constraints on terms to be included in the induced polynomials.

We have empirically compared the multi-target approach to the application of several single-target models. The results of the experiments show that there is no loss in predictive performance when using multi-target models as compared to using multiple single-target models. In addition, the appearance of common terms in the equations for the different targets in the multi-target model makes these models more stable. Because the same equation structure must be good for all targets, and hence the risk of over-fitting is reduced.

Fewer equation structures are considered by multi-target Ciper. We have also included optimizations that use the fact that there are common calculations for each target. This makes the multi-target approach faster than the single-target approach where we build a model separately for each target.

In addition to applying the multi-target approach to several real-world problems, we also apply it to the task of system identification, i.e., discovering dynamics. Here the task is to induce a system of simultaneous differential equations that describe the behaviour of a system whose state changes over time: The state of the system typically consists of a vector of system variables. The time derivatives of the system variables are typically interdependent and expressed as functions (polynomials) of the system variables.

We have used this approach to discover dynamics of chemical reaction networks. In this domain, subsumption constraints have a natural interpretation. They can be used to specify, e.g., a partially known network as prior knowledge of chemical reactions. Constraints proved crucial for the successful reconstruction of an example network.

References

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International, Belmont, Ca.
2. Grünwald, P., Myung, I., & Pitt, M. (Eds.). (2005). *Advances in minimum description length: Theory and applications*. Cambridge, Massachusetts: MIT Press.
3. Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.
4. Newman, D., S. Hettich, C. B., & Merz, C. (1998). UCI repository of machine learning databases.
5. Rissanen, J. (1999). Mdl denoising. *IEEE Transactions on Information Theory*, 46, 2537–2543.
6. Robnik, M. (1998). Pruning regression trees with mdl. *Proceedings of the European Conference on Artificial Intelligence* (pp. 455–459). Brighton, UK: John Wiley and Sons.
7. Todorovski, L., Ljubič, P., & Džeroski, S. (2004). Inducing polynomial equations for regression. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 441–452).
8. Torgo, L. (1998). Regression datasets.
9. Witten, I. H., & Frank, E. (Eds.). (2005). *Data mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann.
10. Grünwald, P. D., Myung I. J., Pitt M. A. (Eds.) (2004) *Advances in Minimum Description Length: Theory and Applications*. MIT Press

Author Index

Antunes, Cláudia , 37

Bade, Korinna, 3

Blockeel, Hendrik , 21

Bringmann, Björn, 4

Calders, Toon, 21

Džeroski, Sašo , 61

Esmeir, Saher, 3

Fromont, Elisa, 21

Goethals, Bart, 21

Gunn, Steve, 49

Kononenko, Igor, 9

Lavrač, Nada, 9

Markovitch, Shaul, 3

Pečkov, Aleksandar, 61

Podpečan, Vid, 9

Prado, Adriana, 21

Todorovski, Ljupčo, 61

Yang, Jianqiang, 49

Zimmermann, Albrecht, 4