Filip Železný
Nada Lavrač (Eds.)

Inductive Logic Programming

18th International Conference, ILP 2008
Prague, Czech Republic, September 2008

Late Breaking Papers

Preface

The 18th International Conference on Inductive Logic Programming was held in Prague, September 10-12, 2008. ILP returned to Prague after 11 years, and it is tempting to look at how the topics of interest have evolved during that time. The ILP community clearly continues to cherish its beloved first-order logic representation framework. This is legitimate, as the work presented at ILP 2008 demonstrated that there is still room for both extending established ILP approaches and exploring novel logic induction frameworks. However, besides the topics lending ILP research its unique focus, we were glad to see at ILP 2008 a good number of papers contributing to areas such as statistical relational learning, graph mining, or the semantic web. To help open ILP to mainstream research areas, the conference featured three excellent invited talks from the domains of the semantic web (Frank van Harmelen), bioinformatics (Mark Craven) and cognitive sciences (Josh Tenenbaum). We deliberately looked for speakers who are not directly involved in ILP research. We further invited a tutorial on statistical relational learning (Kristian Kersting) to meet the strong demand to have the topic presented from the ILP perspective. Lastly Stefano Bertolo from the European Commission was invited to give a talk on the ideal niches for ILP in the current EU-supported research on intelligent content and semantics.

Apart from the main conference track, ILP 2008 featured special sessions devoted to late breaking papers. These papers went through a separate review procedure and the 21 selected papers are contained in this book. They present mainly work in progress, challenge papers, applications reviews, and also brief summaries of relevant work presented by the authors at another conference (such as ECML/PKDD 2008). The late breaking papers were presented at ILP 2008 through posters as well as short plenary talks.

Organizing ILP 2008 was a great experience, thanks to the excellent help we received on several fronts. We are indebted to our generous sponsors who made possible the trips of the invited speakers (sponsored by the US Air Force European Office of Aerospace Research and Development, the PASCAL2 Network of Excellence, the Czech Society for Cybernetics and Informatics and the European Commission) and the best student paper prize (sponsored by the Machine Learning Journal). We are equally grateful to Springer for collaborating so flexibly and pro-actively in preparing the main track proceedings and the Machine Learning Journal special issue. Thanks go as well to the diligent program committee for their reviews of the submitted papers. Their review activity was supported by the MyReview System, which proved to be a powerful, yet easy-to-use, conference management software. ILP would be just three letters were it not for the authors of the submitted papers. To them goes our foremost gratitude. Keep up the good work and submit to ILP 2009!

August 2008 Filip Železný and Nada Lavrač

ILP 2008 Organization

Program Chairs

Filip Železný	Czech Technical University, Prague
Nada Lavrač	Jožef Stefan Institute, Ljubljana

Local Organization Chairs

Jiří Kléma	Czech Technical University, Prague
Peter Vojtáš	Charles University, Prague
Milena Zeithamlová	Action M Agency, Prague

Program Committee

Annalisa Appice, Italy
Hendrik Blockeel, Belgium
Rui Camacho, Portugal
James Cussens, UK
Luc De Raedt, Belgium
Sašo Džeroski, Slovenia
Floriana Esposito, Italy
Alan Fern, USA
Peter Flach, UK
Tamás Horváth, Germany
Katsumi Inoue, Japan
Andreas Karwath, Germany
Kristian Kersting, Germany
Stefan Kramer, Germany
John Lloyd, Australia
Francesca Lisi, Italy
Donato Malerba, Italy
Stan Matwin, Canada
Stephen Muggleton, UK
Ramon Otero, Spain
C. David Page, USA
Bernhard Pfahringer, New Zealand
Jan Ramon, Belgium
Céline Rouveirol, France
Vitor S. Costa, Portugal
Jude Shavlik, USA
Takayoshi Shoudai, Japan
Ashwin Srinivasan, India
Prasad Tadepalli, USA
Tomoyuki Uchida, Japan
Christel Vrain, France
Stefan Wrobel, Germany
Akihiro Yamamoto, Japan
Mohammed J. Zaki, USA
Gerson Zaverucha, Brazil

Local Organization Committee

Petr Buryan
Matěj Holec
Jiří Kubalík
Onřej Kuželka
Karel Moulík
Petr Pošík
Olga Štěpánková
Monika Žáková

Invited Speakers

Josh Tenenbaum	Massachusetts Institute of Technology
Kristian Kersting	Fraunhofer IAIS, Germany
Frank van Harmelen	Vrije Universiteit Amsterdam
Mark Craven	University of Wisconsin in Madison
Stefano Bertolo	European Commission

Additional Reviewers

Ana Luisa Duboc
Jiří Kléma
Oliver Ray

Sponsors

Czech Technical University in Prague
Jožef Stefan Institute, Ljubljana
Charles University in Prague
US Air Force, European Office of Aerospace Research and Development
PASCAL2 European Network of Excellence
Czech Society for Cybernetics and Informatics
Machine Learning Journal
European Commission

Table of Contents

Late Breaking Papers

The phase transition of the bounded ILP consistency problem 1
Erick Alphonse

Accelerating frequent subgraph search by detecting low support structures 7
Petr Buryan

Inductive Graph Logic Programming: work in progress 14
Christophe Costa Florêncio

Experiments with Czech Linguistic Data and ILP 20
Jan Dědek, Alan Eckhardt, Peter Vojtáš

Network Analysis of the ILPnet2 Co-authorship Network 26
Qingyi Gao, Peter Flach

Learning Comprehensible Relational Features to Distinguish Subfossil Decapod Crustacean Dactyls 32
Mark Goadrich, Jeffrey Agnew

Estimating the Parameters of Probabilistic Databases from Probabilistically Weighted Queries and Proofs [Extended Abstract] 38
Bernd Gutmann, Angelika Kimmig, Kristian Kersting, Luc De Raedt

Propositionalizing the EM algorithm by BDDs 44
Masakazu Ishihata, Yoshitaka Kameya, Taisuke Sato, Shin-ichi Minato

Using Bio-Pathways in Relational Learning 50
Matěj Holec, Filip Železný, Jiří Kléma, Jiří Svoboda, Jakub Tolar

Combining answer caching with smartcall optimization in mining frequent DL-safe queries 57
Joanna Józefowska, Agnieszka Lawrynowicz, Tomasz Lukaszewski

Probabilistic Local Pattern Mining 63
Angelika Kimmig, Luc De Raedt

HiFi: Tractable Propositionalization through Hierarchical Feature Construction 69
Ondřej Kuželka, Filip Železný

Learning Conceptual Predicates for Teleoreactive Logic Programs 75
Nan Li, David J. Stracuzzi, Pat Langley

Predicting Gene Coexpression from Pathway Relations 81
Karel Moulík, Filip Železný

TopLog: ILP using a logic program declarative bias 87
Stephen H. Muggleton, Joé C. A. Santos, Alireza Tamaddoni-Nezhad

Mutlirelatonal GUHA Method and Genetic Data 93
Martin Ralbovský, Alexander Kuzmin, Jan Rauch

On and Off-Policy Relational Reinforcement Learning 99
Christophe Rodrigues, Pierre Gérard, Céline Rouveirol

Learning Complex Ontology Alignments - A Challenge for ILP Research .. 105
Heiner Stuckenschmidt, Livia Predoiu, Christian Meilicke

A Simple Model for Sequences of Relational State Descriptions 111
Ingo Thon, Niels Landwehr, Luc De Raedt

Relational Data Mining in Crisis Management 117
Martin Večeřa, Luboš Popelínský

A Sample Complexity for PILP 123
Hiroaki Watanabe, Stephan Muggelton

Author Index 130

The phase transition of the bounded ILP consistency problem

Erick Alphonse

LIPN-CNRS UMR 7030, Université Paris 13, France
erick.alphonse@lipn.univ-paris13.fr

Abstract. A number of recent works have been focusing on analysing the phase transition of the NP-complete ILP covering test, which have been fruitful in linking this phenomenon to plateaus during heuristic search. However, it is only a facet of the ILP complexity as it is very dependent of the search strategy. Its inherent difficulty has to be studied as a whole to design efficient learners. ILP is arguably harder than attribute-value learning, which has been formalised by Gottlob et al. who showed that the simple bounded ILP consistency problem is $\Sigma_2 - complete$. Some authors have predicted that a phase transition could be exhibited further up the polynomial hierarchy and we show this is the case in this problem space, where the number of positive and negative examples are order parameters. Those order parameters are the same as for the k-term DNF consistency problem studied in the context of attribute-value learning. We show that the learning cost exhibits the easy-hard-easy pattern with a lgg-based learner.

1 Introduction

The phase transition framework, which has been strongly developped in many combinatorics domains, like in SAT or CSP domains, since [1], has changed the way search algorithms are empirically evaluated. This lead to new designs of search algorithms, from incomplete to complete solvers and from deterministic to randomised solvers [2].

Symbolic learning, which has been cast more than 20 years ago as search into a state space [3] has known few developments of the phase transition (PT) framework. As far as we know, the only work that studied the PT of learning has been done by [4] who showed that the number of positive and negative examples where order parameters of the k-term DNF consistency problem, which is NP-complete. Indeed, if one keeps one parameter constant and varies the other, one wanders from an under-constraint region, named the "yes" region, associated with a small value, where there is almost surely a solution, to an over-constraint region, named the "no" region, as the parameter value increases, where almost surely no generalisation of the positive examples is correct. A related work studied the PT of the subsumption test which is a key NP-complete sub-problem of learning. Although, this study has been fruitful in linking this phenomenon to plateaus during heuristic search [5], it is only a facet of the ILP

complexity as it is very dependent of the search strategy and does not study the complexity of learning in the PT framework.

ILP is arguably harder than attribute-value learning, like k-term DNF learning, which has been formalised by Gottlob et al. [6] who showed that the simple bounded ILP consistency problem is Σ_2-complete. This is one class higher in the polynomial hierarchy than NP-complete (or Σ_1-complete) problems. Some authors, e.g. [7], have predicted that a phase transition could be exhibited further up the polynomial hierarchy and therefore that this framework could be useful to other PSPACE-complete problems.

We show this holds for the bounded ILP consistency problem, where the number of positive and negative examples are order parameters of the phase transition. We show that the median learning cost exhibits the easy-hard-easy pattern with a simple lgg-based learner.

We present, in the next section, the necessary background on the bounded ILP consistency problem and the model RLPG which is a generator proposed to study this problem, first described in [5]. The section 3 will present the complete learner used to answer the ILP consistency problem. Section 4 will exhibit the phase transition, beyond NP, of the ILP consistency problem with respect to the two order parameters which are the number of positive and negative examples. We show that the solver used allow to exhibit the easy-hard-easy pattern of median search cost. Finally, we will conclude and draw some perspective and benefits of these results for relational learning.

2 Background

In this article, we study what has been termed the bounded ILP consistency problem for function-free Horn clauses by [6]. Given a set of positive examples E^+ and a set of negative examples E^- of function-free ground Horn clauses and an integer k polynomial in $|E^+ \cup E^-|$, does there exist a function-free Horn clause h with no more than k literals such that h θ-subsumes each element in E^+ and h does not θ-subsume any element in E^-.

[5] proposed a random generator for this problem, named model RLPG (Relational Learning Problem Generator). A learning problem instance in this model is denoted $RLPG(k, n, \alpha, N, Pos, Neg)$. The parameters k, n, α, N are related to the definition of the hypothesis and example spaces. Pos and Neg are the number of positive and negative examples respectively. The first four parameters are defined in order to ensure that a subsumption test between a hypothesis and an example during search encode a valid CSP problem, following models for random CSP. This requirement is imposed as the model RLPG was proposed to study the impact of the phase transition of the subsumption test on heuristic search. We briefly recall their meaning and focus on the last two parameters.

$k \geq 2$ denotes the arity of each predicate present in the learning language, $n \geq 2$ the number of variables in the hypothesis space, n^α the domain size for all variables, and finally N the number of literals in the examples built on a given predicate symbol. Given k and n, the size of the bottom clause of the hypothesis

space $\mathcal{L}_h$ is $\binom{n}{k}$. It encodes the largest constraint network of the underlying CSP model. Each constraint between variables is encoded by a literal built on a unique predicate symbol. $\mathcal{L}_h$ is then defined as the power set of the bottom clause, which is isomorphic to a boolean lattice. Its size is $2^{\binom{n}{k}}$.

Learning examples are randomly drawn, independently and identically distributed, given k, n, α and N. Their size is $N\binom{n}{k}$. Each example defines N literals for each predicate symbol. The N tuples of constants used to define those literals are drawn uniformly and without replacement from the possible set of $\binom{n^\alpha}{k}$ tuples.

3 Exhibiting the easy-hard-easy pattern with a complete solver

Besides the phase transition behaviour of decision problems, a strong motivation of its study is that it is conjectured that the hardest problem instances occur in the phase transition (see e.g. [1, 8, 7]). The under-constraint problems from the "yes" region appear to be easily solvable, as there are a lot of solutions. This is the same for over-constraint problems from the "no" region as it is easy to prove that they are insoluble. These findings have been corroborated on several problems, with different types of algorithms, and it is considered that the problem instances appearing in the phase transition are inherently hard, independently of the algorithms used. In the "yes" and "no" regions, the easy ones, the complexity appears to be very dependent of the algorithm. There are, in these regions, some problems exceptionally hard, whose complexity dominates the complexity of instance problems in the phase transition region for certain types of algorithm [8].

In other words, exhibiting the easy-hard-easy pattern require a "good" algorithm. We propose to use a depth-first lgg-based algorithm to solve the ILP consistency problem, DF-BDD (Depth-First Bottom-up Data-Driven), which is similar to the approach of [4] for the k-term DNF consistency problem. Its Prolog code is given below:

```
1 df_bdd(Sol,[],_,Sol).
2 df_bdd(Hypo,[Pos|L_Pos],L_Neg,Sol) :-
3         lgg(Hypo,Pos,LGG), % non-deterministic computation of a LGG
4         % consistency check
5         correctness(LGG,L_Neg),
6         df_bdd(LGG,L_Pos,L_Neg,Sol).
```

The computation of lggs (line 3) is done with depth-first search into possible subsets of the hypothesis. It outputs the largest subsets that subsume the example. The implementation is rather naive, which may blur the easy-hard-easy pattern, as we will see. Once a lgg has been computed, we test, in a depth-first way, if it is correct with respect to all negative examples (line 5).

4 Numbers of positive and negative examples as order parameters

In this section, we study the effect of the number of positive and negative examples on the solubility probability and the solving cost of the ILP bounded consistency problem. If we refer to section 2, $RLPG$ is parametrised with 6 parameters but we only study the last two, Pos and Neg, as the effect of the other parameters have been already studied in [5] for constant number of positive and negative examples. Here, we focus on few settings for these parameters, with $k = 2$, $n = 5$ and $n = 6$, to study different problem sizes, $\alpha = 1.4$ and $N = 10$. The choice of these parameters ensures that we do not generate trivially insoluble problems [7], but also various experiments, not shown here, indicated that there were representative of the phase transition behaviour of the ILP consistency problem. In all experiments below, statistics were computed from a sample of 500 learning problems.

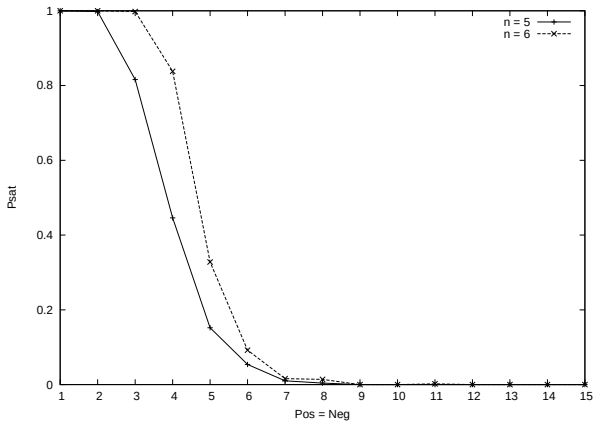


Fig. 1. Probability of statisfiability according to the number of learning examples (Pos = Neg), with $n = 5$ and $n = 6$

We start by varying both Pos and Neg. Figure 1 shows the solubility probability of the ILP consystency problem when $Pos = Neg$ are varied from 1 to 25, for $n = 5$ and $n = 6$. As we can see, when the number of examples is small, there is almost surely a consistent hypothesis, and when the number is large it is almost surely impossible to find a consistent hypothesis. The cross-over point, where the probability of solubility is about 0.5, is around 4 for $n = 5$ and 5 with $n = 6$. It is not surprising that it increases with bigger problems. For $n = 5$, the hypothesis space size is 2^{10} and 2^{15} for $n = 6$. We could not conduct experiments for larger values of n as the hypothesis space grows too fast in $RLPG$. For instance, $n = 7$ sets a hypothesis space of size 2^{21}, which cannot be handled by our complete solver. In the future, it would be interesting to modify $RLPG$

to specify the size of the bottom clause and then draw the number of variables accordingly.

Figure 2 and 3 show the associated cost (the median cost along with the 25th and 75th percentiles) to solve the problem instances, with $n = 6$. We measured the cost by recording the time in milliseconds, as well as the number of backtracks of the subsumption procedure, needed to solve a learning problem. The latter seems relevant, as the subsumption test is used to compute the lggs.
We can see that a complexity peak is associated with instances in the phase transition region, and that the search cost follows the easy-hard-easy pattern. The complexity in the "no" region slowly decreases as the number of examples increases, where we could have expected a sharper decrease, but it may be related to our implementation.

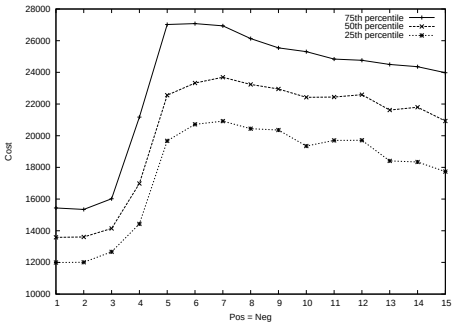


Fig. 2. Cost in resolution time (ms.) according to the number of learning examples (Pos = Neg), for $n = 6$

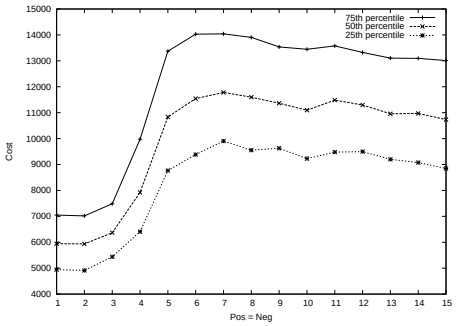


Fig. 3. Cost in number of bactracks of the subsumption test according to the number of learning examples (Pos = Neg), for $n = 6$

We study now the phase transition along the number of positive examples, for constant values of Neg, but omit cost plots. The results are almost symmetric when Pos is constant and Neg varies, and is not shown here. Figures 4 and 5 show the phase transition when Pos varies from 1 to 25, for $n = 5$ and $n = 6$ respectively. The transition becomes sharper as Pos increases, which is not surprising as the subset of complete hypotheses shrinks with Pos.

5 Conclusion

It is conjectured that the phase transition of decision problem can be exhibited further up the polynomial hierarchy and therefore that this framework could be useful to other PSPACE-complete problems. We have shown that this holds with the bounded ILP cosistency problem, a Σ_2-complete problem, which exhibits a phase transition in its solubility, with the number of positive and negative examples as order parameters. The search cost as given by a depth-first lgg-based solver exhibits the easy-hard-easy pattern. This is the first work that

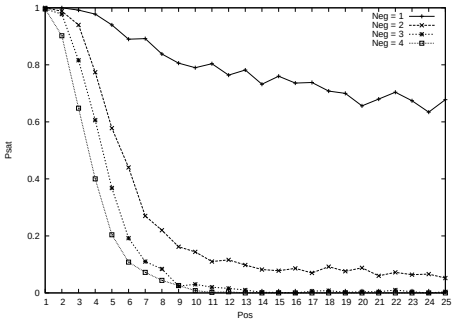


Fig. 4. Probability of statisfiability according to the number of positive examples with $n = 5$, for $Neg = 1, 2, 3, 4$

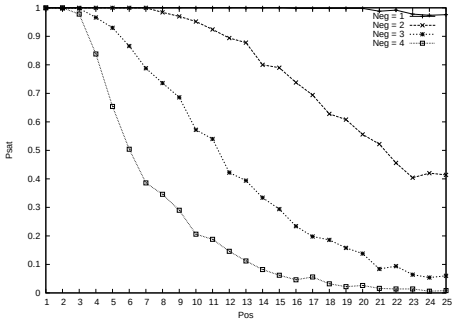


Fig. 5. Probability of statisfiability according to the number of positive examples with $n = 6$, for $Neg = 1, 2, 3, 4$

study the phase transition of learning in ILP and we hope that it will stimulate algorithmic developments, in the line of what has been done in combinatorics. It points out interesting follow-ups: the model RLPG has been used to generate random problems and we plan to study the impact of its other parameters on the generation of hard instances; we plan to generate hard problems to study the different solvers proposed in ILP.

Acknowledgment

We thank Aomar Osmani for his support and the numerous discussions about this work, as well as Henry Soldano and Dominique Bouthinon.

References

1. Cheeseman, P., Kanefsky, B., Taylor, W.: Where the really hard problems are. In Proc. of the 12th Int. Joint Conf. on Artificial Intelligence. (1991) 331–340
2. Carla Gomes, Heny Kautz, A.S., Selman, B.: Satisfiability solvers. In: Handbook of Knowledge Representation. (2007)
3. Mitchell, T.M.: Generalization as search. Artificial Intelligence **18** (1982) 203–226
4. Rückert, U., Kramer, S., Raedt, L.D.: Phase transitions and stochastic local search in k-term DNF learning. Lecture Notes in Computer Science **2430** (2002) 405–417
5. Alphonse, E., Osmani, A.: A model to study phase transition and plateaus in relational learning. In: Proc. of Inductive Logic Programming. (2008) to be published.
6. Gottlob, G., Leone, N., Scarcello, F.: On the complexity of some inductive logic programming problems. In Proc. of the 7th Int. Workshop on Inductive Logic Programming. Volume 1297 of LNAI. (1997) 17–32
7. Gent, I.P., Walsh, T.: Beyond np: the qsat phase transition. In Proc. of the 16th Nat. Conf. on Artificial intelligence. (1999) 648–653
8. Davenport, A.: A comparison of complete and incomplete algorithms in the easy and hard regions. In: Work. on Studying and Solving Really Hard Problems, CP-95. (1995) 43–51

Accelerating frequent subgraph search by detecting low support structures

Petr Buryan

Gerstner Laboratory, Department of Cybernetics, Czech Technical University, Technicka 2, Prague, Czech Republic. buryan@labe.felk.cvut.cz

Abstract. Latest research in the domain of frequent subgraph mining focuses mainly on completely searching the subgraph space, efficient candidate enumeration, duplicate identification, embeddings storing and other effective memory allocation issues leaving the search process itself aside. This paper shows two techniques that contribute to accelerating the complete search. They are based on the idea of using information discovered so far during the search to limit time spent by unnecessary testing structures for subgraph isomorphism during the support calculations and efficient utilisation of this information to drive the search process. As a result, time spent by subgraph isomorphism tests is minimised.

1 Introduction

Graph mining, particularly frequent subgraph mining became an area attractive for data-mining researchers during past decades. The domain focuses mainly on finding interesting common substructures (patterns) in a set of given graphs that could be used e.g. for classification purposes. The level of interestingness can be approached in many ways; however, it is most often the frequency of pattern appearances among the graphs that matters.

Despite the need for efficient graph mining algorithm, several approaches targeting this domain extending classical complete subgraph lattice search approach [8] appeared as late as mid 1990s. The first to start the avalanche was Subdue algorithm [9] followed by others among which Warmr [3], AGM [4], Gaston [1], gSpan [5], FSG[10], FFSM[12] or MolFea [6] can be stated here. A good overview is presented e.g. in [2]. The majority of these implementations focus on complete pattern search and are able to work with labelled graphs, i.e. graphs where vertices (and often also edges) are assigned symbols from some domain describing alphabet (e.g. molecule structures can be seen as an example of labelled graph).

Main sources of complexity in the frequent subgraph mining are:

1) The graph isomorphism problem (GI complexity – in general neither P nor NP).
2) The subgraph isomorphism problem (NP-complete).
3) The state space search (generally NP-complete).

And it is not only that solutions of problems 1) and 2) are rather time demanding but these tasks have to be solved extremely often during the search. In addition, search spaces that are determined by the most of real graph databases are sparse meaning that only a small fraction from all possible subgraphs really occurs within the databases. As a consequence, search by random sampling is very difficult and

frequent subgraph search currently bears up against randomised techniques (e.g. genetic algorithms). Another effect is that large amount of patterns generated during the search turns out to have zero coverage which is also the drawback that is targeted by this paper.
On the other hand, constraints such as maximal node degree often exist that help a lot in narrowing the search space and speeding up the algorithm by filtering out fabled structures.
Latest research [2] focuses mainly on efficient candidate enumeration, duplicate identification, embeddings storing and other effective memory allocation issues leaving the search process itself aside. This paper shows two techniques that also contribute to accelerating the search.
First one exploits the idea that instead of directly performing complicated calculations to determine support of a new discovered pattern, it is cheaper first to test the pattern for presence of smaller fragments that were discovered so far and have insufficient support. By this technique a lot of unnecessary lengthy subgraph isomorphism testing can be avoided.
Second technique that is presented in this paper is sorting the open list in ascending order according to the support of the structures listed (opposite of standard sorting for greedy search). In combination with previous technique it enables fast detection of low-support structures near the transition.

2 Graphs

A graph or undirected graph G is an ordered pair $G = (V,E)$, that is subject to the following conditions:

- V is a set of vertices or nodes;
- E is a set of pairs (unordered) of distinct vertices, called edges.

A labelled graph is a graph with labels assigned to its nodes and edges i.e. an ordered triple $G = (V,E,\lambda)$, where

$$E \subseteq V \times \Sigma_E \times V,$$
$$\lambda: V \rightarrow \Sigma_V,$$

Σ_V being the set of vertex labels (node label alphabet), Σ_E the edge label alphabet and λ is the vertex labelling function.

Verifying whether a graph G is a subgraph of H means finding set of mutually matching pairs of both vertices and edges in these structures. Let $V(G)$ be the vertex set of a graph and $E(G)$ its edge set. Graphs G and H are subisomorphic *iff* there is an injection (often called embedding)

$$f: V(G) \rightarrow V(H),$$

such that $(u,a,v) \in E(G)$ if and only if $(f(u), a, f(v)) \in E(H)$ and the label of $f(u)$ equals to that of u for all vertices u.. In other words, a graph G is sub-isomorphic to graph H *iff* graph G is isomorphic to at least one subgraph of H.
On the other hand, graphs G and H are isomorphic *iff* there is a bijection

$$f: V(G) \rightarrow V(H),$$

such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H and the label of $f(u)$ is the same as that of u for all vertices u.

Let analysed graph database D_G contain a collections of graphs G. The frequency of subgraph pattern p is calculated as a ratio of size of set $D_G(p)$, i.e. set of graphs from database that contain p to the size of set of all graphs in database D_G.

$$freq(p, D_G) = \frac{|D_G(p)|}{|D_G|},$$

$D_G(p) = \{G \in D_G \mid contains(G, p)\}$, and $contains(G, p)$ means that p is a subgraph of G.

The primary task of the subgraph mining algorithms is as follows:

Given a set of graphs (graph database) $D_G = \{G_0, G_1, \ldots, G_n\}$, find all subgraphs (patterns) appearing in at least $freq_{min} \cdot |D_G|$ graphs (isomorphic subgraphs are considered the same subgraph).

This means finding patterns the frequency of which is above some given level of minimal support $freq_{min}$

$$freq(p, S) \geq freq_{\min}.$$

3 Low-support structures detection

Sorting the structures in the list of patterns to be extended (*"open list"*) according to their coverage is quite a common approach in algorithms searching the state space, mainly in greedy or other heuristic driven approaches. The main idea is to prefer for expansion those patterns that have higher support so that interesting patterns are found fast.

Suppose a new pattern g was created by extending pattern g_0 the coverage of which ($freq_{g0} = \frac{|D_G(g_0)|}{|D_G|}$) was sufficient according to given threshold. Calculating the coverage of g means to perform approximately $|D_G| \cdot freq_{g0}$ subgraph isomorphism tests on graphs from the database D_G (not considering any case specific filtering).

The time complexity of subgraph isomorphism (SGI) testing grows fast with the size of the graphs tested. The approach presented here exploits the idea that instead of performing these SGI calculations with g on D_G directly, it is cheaper first to test g for presence on fragments smaller than g that were discovered so far and have insufficient support before advancing to testing large graphs for presence of g. By this technique a lot of unnecessary lengthy SGI testing can be avoided.

The situation is illustrated in Fig 1. Even though learning in the beginning of the search that the structure *b-a-b* is not present in the graph database, state-of-the-art algorithms spend time recalculating the fact again and again in future steps. From the image it is clear, that this is the case not only for pattern-growth approaches but for pattern combination (Apriori-like) approaches as well, as in this example the pattern *b-a-a* is frequent but extension created by its combination may be not.

Second technique that is presented in this paper is sorting the open list in ascending order as to the support of the structures which is the opposite of sorting used by standard greedy search. In combination with previously introduced keeping track of infrequent patterns during complete search for filtering purposes it enables fast detection of low-support structures near the transition.

Sorting the list in such manner (less frequent ones come first) helps to delimit fast the border between frequent and infrequent patterns. The smaller infrequent patterns are

thereby immediately at hand and can be instantly used for filtering. Sorting open list according to pattern size brings similar effect but sorting by support criterion leads to faster run of the algorithm.

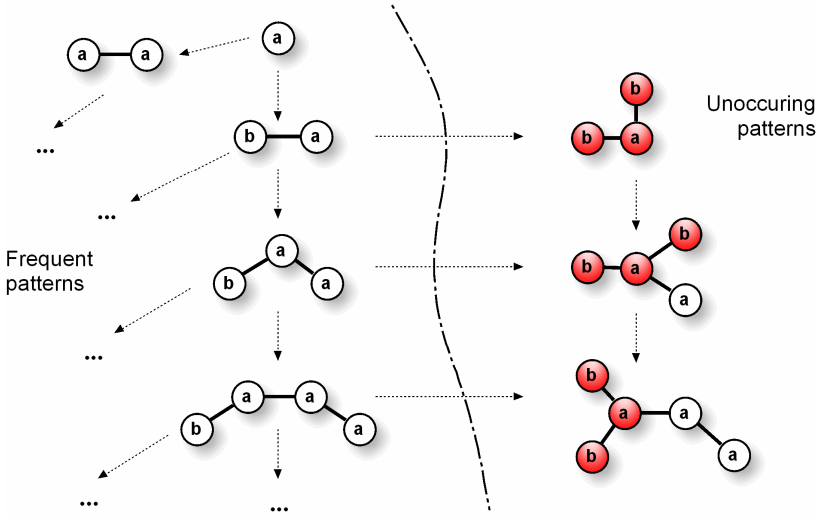


Fig. 1. Example of a part of subgraph lattice

4 Experimental results

In this section, results of basic experiments on the Mutagenesis classification problem are presented. Standard PTE dataset was used as e.g. in [5, 7] - a smaller widely used benchmark dataset containing few hundreds of chemical compounds with an average size of 30 nodes (maximum size of a graph 214 nodes) classified as carcinogens or otherwise. The dataset included hydrogen atoms in the graph encoding.

In order to evaluate usefulness of techniques presented, basic pattern growth based state space searching algorithm with embedding lists was used. No pattern is generated twice which is ensured by several simple duplicate detection techniques based on pattern statistics and graph invariants in combination with full pattern GI testing. Nevertheless, the idea presented here might be in a similar way implemented to any of the above mentioned algorithms.

The impact of presented improvement depends on the graph database determining the structure of searched space. However, keeping track of only small infrequent fragments up to size of several nodes usually speeds up the search significantly. Therefore, it is sufficient to remember only the smallest subgraphs instead of all discovered infrequent patterns. Graphs in Figures 2 and 3 show, that majority of all infrequent patterns discovered is of smaller size and majority of pruning is performed using these small infrequent patterns as well.

The process may be further optimised for pattern growth algorithms (e.g. Gaston) by indexing the database of patterns with support below threshold by graph leafs. Should the new pattern g include one of these infrequent fragments, it has to be one of fragments that include also the node/edge last added to g. This enables efficient search for infrequent candidates that may be present in tested graph.

Apart from higher memory demand, a drawback may be seen in fact, that as the patterns with high coverage come lately to expansion, interrupting the algorithm during search causes loosing potential interesting results which might be limiting for incomplete searches.

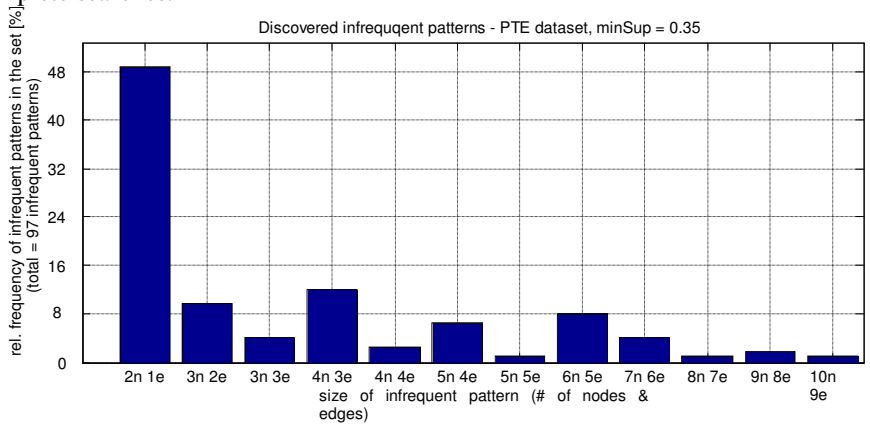


Fig. 2 Sizes of discovered infrequent patterns, PTE dataset, min. support 0.35

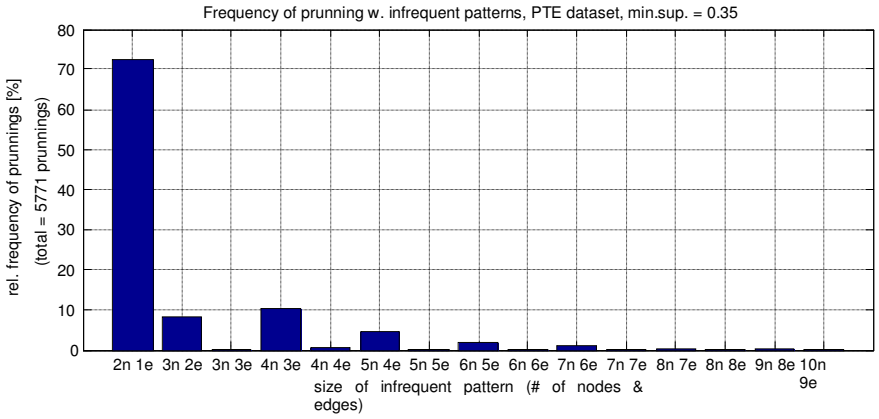


Fig. 3. Frequency of pruning actions with discovered infrequent patterns, PTE dataset, min. support 0.35

The experimental steps for determining impact on search time were as follows:

1) Sort out all node and edge labels with insufficient support in the dataset;
2) Perform classical subgraph space search, sort / do not sort patterns according to their support;
3) For each new pattern found perform following checks:
    a) check, if pattern was not generated before (pattern statistics, invariants, full GI check);
    b) check / do not check, if pattern does not contain some of discovered infrequent patterns (pruning phase);
    c) calculate pattern support (evaluation phase).

Results presented in Table 1 show, that pruning lattice by utilisation of infrequent patterns generally requires only a fraction of time needed for coverage calculations

and it helps minimising the number of "unnecessary" SGI tests performed on the dataset thereby significantly speeding up the search process. In addition, sorting the list of patterns to be expanded according to their support (low support first) brings another speed-up to the search process.

Table 1. PTE dataset experiments

Minimal support [%]	**Pattern Counts**		**Pruning Time [s]** *) **)	**Evaluation Time [s]** *) ***)		
	Frequent patterns	***Infrequent patterns***		***pruning with sorting***	***Pruning, no sorting***	***no pruning, no sortrting***
15	433	259	14.4	62.59	180	326
35	66	97	0.4	4.9	9.9	40
50	37	57	0.2	1.8	3.8	7.7

*) Average value over 10 search runs

**) Total time spent testing whether patterns contain one of infrequent patterns discovered

***) Total time spent calculating patterns support

5 Conclusions

The results presented in the paper indicate, that focusing search on border line dividing patterns with insufficient support threshold (i.e. patterns of low coverage yet above threshold) during complete subgraph search speeds up the search process significantly. Two main observations follow:

4) significant speed-up is achieved during complete subgraph search by using pruning based on infrequent patterns discovered so far before proceeding to calculating pattern support;
5) preferring patterns with lower support (above given minimal support limit) during enumeration brings another positive impact on search runtime.

Currently, extension of this idea into the Gaston algorithm [1] is in progress as it seems to be the most promising state-of-the-art algorithm.

References

[1] Nijssen S., Kok J. N.. *Frequent Graph Mining and its Application to Molecular Databases*, Proceedings of the 2004 IEEE Conference on Systems, Man & Cybernetics (SMC2004), 2004.

[2] Washio T., Motoda H., *State of the Art of Graph-based Data Mining*, SIGKDD Explorations Special Issue on Multi-Relational Data Mining, pp 59-68, Volume 5, Issue 1, 2003

[3] King R. D., Srinivasan A., Dehaspe L., *Wamr: a data mining tool for chemical data*, J. Comput.-Aid. Mol. Des., 2001, 15, pp.173-181.

[4] Inokuchi A., Washio T., Okada T., Motoda H., *Applying the Apriori-based Graph Mining Method to Mutagenesis Data Analysis*, Journal of Computer Aided Chemistry, 2001, 2, pp.87-92.

[5] Yan X., Han J., *gSpan: Graph-Based Substructure Pattern Mining*, Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), IEEE Computer Society, 2002, pp.721-724.

[6] Helma C., Cramer T., Kramer S., de Raedt L., *Data Mining and Machine Learning Techniques for the Identification of Mutagenicity Inducing Substructures and Structure Activity Relationships of Noncongeneric Compounds*, J. Chem. Inf. Comput. Sci., 2004, 44, 1402-1411.

[7] Srikant R. & Agrawal R.. Mining Generalized Association Rules. Proc. of Very Large Data Bases Conference, pp. 407–419, 1995.

[8] Ullmann, J. R.: "An Algorithm for Subgraph Isomorphism". Journal of the ACM, 23(1), pp.31–42, 1976.

[9] Cook D. J., Holder L., *Substructure discovery using minimum description length and background knowledge*, Journal of Artificial Intelligence Research, 1, pp.231-255, 1994

[10] Kuramochi M., Karypis G., "Frequent Subgraph Discovery," *ICDM*, p. 313, First IEEE International Conference on Data Mining (ICDM'01), 2001

[11] Inokuchi A., Washio T., Motoda H.,, *Complete mining of frequent patterns from graphs*, Mining graph data, Machine Learning, 50, pp.321-354, 2003

[12] Huan J., Wang W., Prins J., "Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism," *ICDM*, pp. 549, Third IEEE International Conference on Data Mining (ICDM'03), 2003

Inductive Graph Logic Programming: work in progress

Christophe Costa Florêncio

Department of Computer Science, K.U. Leuven, Leuven, Belgium
Chris.CostaFlorencio@cs.kuleuven.be

Abstract. There has recently been a lot of interest in learning from graphs. Most approaches to this problem up to this point have been pragmatic. While there definitely exists a need for such research, theoretically sound approaches that yield comprehensible theories of higher expressive power are also desirable.
These are strong points of ILP, so it seems a good starting point for such an approach. Thus, in order to express hypotheses about graphs, a graph logic has to be chosen. A good candidate is Cardelli et al's *GL*, since it has decent expressive power while retaining acceptable computational complexity. The most important operator in this logic is composition, which non-deterministically composes (splits) a graph in two parts. This makes the logic very flexible, since it allows quantification over subgraphs with specified properties, but this power comes at a price.
We argue that a restricted form of composition is much more useful for our purposes, and that little expressive power is sacrificed as long as recursive theories are allowed.

1 Introduction

There has recently been a lot of interest in learning from graphs and structured data. Most approaches to this problem up to this point have been more or less pragmatic, i.e., focused on creating algorithms that achieve decent accuracy on large datasets, using just a limited amount of computational resources (see [CH06] for a recent collection of this kind of work). Such algorithms tend to yield theories about their domain that have low expressive power, and sometimes even take the form of just a set of frequent subpatterns.

While there definitely exists a need for such research, it is also desirable to have approaches that can be shown to be mathematically sound, while yielding comprehensible theories of higher expressive power. Such approaches of course have an intrinsic theoretical interest, but certainly offer the perspective of useful applications as well.

Since soundness and comprehensible theories are strong points of ILP, techniques from this field are an obvious good starting point for such an approach. Thus, in order to express hypotheses about graphs, a graph logic has to be chosen. A good candidate is Cardelli et al's *GL* and its variants, since it has decent

expressive power (between FO and MSO) while retaining acceptable computational complexity to be usable as query language. The most interesting operator in this logic is composition, which non-deterministically composes (splits) a graph in two parts. This makes the logic very flexible, since it allows quantification over subgraphs with specified properties, but this power of course comes with a pricetag.

In this paper, we argue that in practice, *GL* as programming language requires more computational resources than *GL* used as query language. We also argue that composition yields problems when defining an lgg operator for graphs as logical objects, since such an operator could yield an exponential number of lggs. So, a restricted (linear) form of composition is much more useful for our purposes, and that little expressive power is sacrificed as long as recursive theories are allowed. In other words, the computational burden can be shifted from the operator to the more extensive use of recursive predicates. Although learning recursive theories cannot be considered 'solved', it is at least a familiar research problem. Some methods have been proposed to tame its complexity, and these could be readily be applied in our framework.

2 The graph logic *GL* and its variants

The spatial graph logic GL_μ was introduced in [CGG01,CGG02]. Its expressive power was studied in [DGG07], which introduced the notation *GL* and GL_μ for the variants without and with the least fixed-point operator, respectively, and for their linear variants *LGL* and LGL_μ.

Of course, alternative logics have been known for a long time (first order language of graphs, MSO logic for graphs) as well as alternative query languages (TQL, Strudel and Graphlog). The main reason we are interested in *GL* (and its variants) is that it allows graphs to be treated as logical objects and that it expresses properties of graphs in a very direct and concise way. This makes it an interesting candidate for ILP-like applications, since it is expected to introduce a language bias that is easily interpretable in terms of graph properties, enhancing comprehensibility. This is true both of the theories generated by a learning algorithm and the background theory that the user might specify in order to define a language bias.

3 The Query Language

The core of the graph query language consists of first order logic, with quantification restricted to edge label- (α) and node variables (ξ), an 'edge' predicate written as $\alpha(\xi_1, \xi_2)$, and the (graph) composition operator $|$. Graphs are represented as multisets of edges, and the query $Q_1|Q_2$ is true iff the current graph can be split in parts G_1 and G_2 such that Q_i is true of G_i. For the purpose of this discussion, details of transducers, abstraction and the least fix-point operator can be ignored.

Composition is sometimes called *exponential* composition, to distinguish it from the linear variant which is denoted $Q_1]Q_2$. For this variant, the left graph, for which Q_1 must be true, consists of just one edge.

We define the useful predicate here(x) with the formula in_degree(x) $\geq 1 \vee$ out_degree(x) ≥ 1.[1] The formula $\forall x.\text{here}(x) \rightarrow \phi$ says that ϕ holds for all nodes in the graph. This is abbreviated as $\forall x \in G.\phi$ ($\forall x_1, \ldots, x_n \in G.\phi$).

This notation allows the expression of some common properties of graphs. For example, the following formula defines a graph consisting of a single path: $\text{path}(x, y) \stackrel{\text{def}}{=} \text{in\_degree}(x) = 0 \wedge \text{out\_degree}(y) = 0 \wedge \forall z \in G.((z \neq x \rightarrow \text{in\_degree}(z) = 1) \wedge (z \neq y \rightarrow \text{out\_degree}(y) = 1))$. A graph consisting of just (nonoverlapping) cycles is defined by $\text{cycles} \stackrel{\text{def}}{=} \forall x \in G.\text{degree}(x) = 2$. Transducers can be used to define operations on graphs like transitive closure, for example.

This logic is not able to deal with hyperedges and isolated vertices, and does not allow for labels for vertices. The latter can easily be introduced. The logic does not really need to be extended to deal with hyperedges; there are well-known ways to rewrite a hypergraph to a 'normal' graph. However, to improve readability it is preferable to work on hypergraphs directly, all that is needed for this is some syntactic sugaring. Since hyperedges are by default ordered, ordered trees are just a specific case of such graphs, and these are obviously useful in computational linguistics.

The same is true of isolated vertices. The logic *GL* and variants do not allow for this, since they define graphs as just multisets of edges. Once all edges that have vertex v as origin or destination are removed from a graph, vertex v itself is no longer accessible either. It is however very easy to encode such graphs, simply by introducing a reserved edge label and using it to label loop edges for all vertices. This obviously does not extend the expressive power of the logics.

4 Complexity and expressive power

In [DGG07] it is demonstrated that without recursion, the linear and exponential versions of the logic are equivalent to first-order (FO) and monadic second-order (MSO) logics on graphs representing strings, respectively. Extended with the fix-point operator for recursion, both are able to express PSPACE-complete problems.

For query languages, it makes sense to distinguish the following:

1. *combined complexity* $\{(G, \phi) : G \models \phi\}$;
2. *data complexity* the complexity class that contains all sets $\mathcal{G}_\phi$.

The combined complexity takes the size of both query and database into account, the data complexity is defined strictly in terms of size of the database. Since in real-life situations, queries are generally much smaller than the databases

[1] We do not need to introduce natural numbers into the logic for this; [DGG07] shows how to define the degree predicates for any number n.

they query, the latter is considered to be more realistic. However, in the case of graph mining, the so-called transactional setting is quite common. This entails dealing with many graphs of restricted size, and thus the combined complexity, which is generally higher, may be more relevant in such cases. Whether this is really the case is hard to determine a priori, and can probably only be decided in practice.

However, in an ILP context, *GL* would be used as a description- or programming language more than a query language. In other words, it would not be used exclusively for model checking. This leaves open the possibility that the complexity goes up in such a setting, and in the following we argue that this is actually the case.

5 Learning

Casting graph learning in the ILP framework makes a whole spectrum of approaches to this problem possible: learning from positive and negative examples, learning from positive data, intensional clustering, decision trees or -lists, regression etc. All of these require the definition of a refinement operator for graphs, like least general generalization.

As the name suggests, the least general generalization operation is meant to generate, given two clauses, a clause that generalizes both, but is still as specific as possible. One aspect of this operation is the generalization of literals, which involves matching and generalizing their arguments. Normally anti-unification is the appropriate approach to this problem. Given two terms t_1 and t_2, anti-unification yields a (unique) term t such that there exist substitutions σ_1, σ_2 such that $\sigma_1[t_1] = t$ and $\sigma_2[t_2] = t$.

In the case of graph logic, where graphs are represented by multisets, this approach will not work. One reason is that there generally is no unique term. The composition operator $|$ is both associative and commutative. Unification of graphs that are represented using this operator is thus equivalent to a special case of unification under an equivalence theory (E-unification), namely unification modulo associativity and commutativity.

When used as a query language, generally one of the terms is ground (since the logic is only used for model checking) and this greatly simplifies matters. However, when used as a programming language, where both terms can contain variables representing subgraphs, nodes and edge labels, the full power of AC-unification is necessary. This operation is known to be finitary, i.e., it always yields a finite number of most general unifiers, if the terms are unifiable. This number can be very high even for simple terms ([BHK+88]), so computing the complete set of mgus is generally not a feasible approach. Even the problem of checking whether two terms containing AC-function symbols are unifiable (AC-matching) is only known to be in NP ([KN92]).[2]

Thus, pure composition seems to be an operation that is too powerful for our purposes. Note that the problems with composition remain even with a

[2] However, AC linear matching is in P ([BH96]).

restricted class of graphs for which otherwise difficult problems are known to be polynomial-time solvable, such as outerplanar graphs. It is also difficult to see how such problems could be avoided in any formalism with the same type of multi-set semantics. It may be possible to come up with optimizations which compile out composition in certain cases, but this is expected to be very hard.

Thus, linear composition is much more attractive from a computational point of view, especially when combined with recursion, and makes implementation easier as well. Whether GL has the same expressive power as LGL_μ is currently an open question, but it is clear that these systems are very close in expressive power ([KN92]). Thus, it seems we can shift the computational burden from the exponential composition operation to recursive clauses.

6 Recursion

In order to restrict composition to the linear case while preserving expressive power, recursive predicates must be allowed. Learning recursive logic programs is a notoriously difficult problem, however, positive results do exist. For example, in [Sha83] an algorithm is presented that identifies such programs in the limit, in the context of learning from interpretations, from positive and negative data. Generally, positive results are either theoretically sound but only applicable to a restricted class, or theoretically unjustified heuristics are used to prune the search space. We give some more examples:

In [Mal03] the ILP system ATRE is described. Building on work on multiple predicate learning (cf [RLD93]), ATRE is able to learn recursive theories from real world data in reasonable time. ATRE explores just a polynomially bounded part of the search space, although it may still take exponential time to do so. A short overview of work in this direction can also be found in [Mal03].

In the setting of nonmonotonic inductive logic programming, introduced in [Hel89], the focus is on finding interesting properties of the examples. In the formalization from [RD94], concepts are represented as clausal theories, and examples as interpretations that are models of the theory. It has been shown there that first order range-restricted clausal theories consisting of clauses made up of up to k literals of size at most j are PAC-learnable from positive data in polynomial time. This is an expressive class which includes recursive concepts. This framework has been implemented as the CLAUDIEN system, which does not generate all clauses in $jk-\mathrm{CT}$, but uses an optimal refinement operator and a bias specification mechanism.

A more theoretical perspective is offered by work done on the subject of elementary formal systems (EFS), basically logic programs consisting of definite clauses whose arguments are patterns instead of terms. A definite clause of an EFS is hereditary (H-EFS) if every every pattern in the body is a subword of a pattern in the head. It has been shown ([MSS00]) that $H-EFS(m,k,t,r)$ is polynomial-time (PAC)learnable, where theories consist of at most m hereditary definite clauses with predicate symbols of arity at most r, where k and t bound the number of occurences of variables in the head and the number of atoms in the

body, respectively. This is a strong theoretical result, and offers the perspective of PAC-learnability results for *GL*-based programs. For this we need a notion of pattern that has a multi-set semantics, which could easily be done.

7 Conclusions

The exponential composition operator is known to be more or less tractable in the context of a graph query language. However, for an ILP setting it seems all but unusable, and the linear variant seems a much better candidate. In order not to lose expressive power, theories expressed with this operator need to be recursive, which although problematic is not infeasible for restricted subclasses.

References

[BH96] Jochen Burghardt and Birgit Heinz. Implementing anti-unification modulo equational theory. Arbeitspapiere der GMD 1006, Fraunhofer Institute, June 1996.

[BHK+88] Hans-Jürgen Bürckert, Alexander Herold, Deepak Kapur, Jórg H. Siekmann, Mark E. Stickel, Michael Tepp, and Hantao Zhang. Opening the ac-unification race. *J. Autom. Reason.*, 4(4):465–474, 1988.

[CGG01] Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. A spatial logic for querying graphs, 2001.

[CGG02] Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. A spatial logic for querying graphs. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610, London, UK, 2002. Springer-Verlag.

[CH06] Diane J. Cook and Lawrence B. Holder. *Mining Graph Data*. John Wiley & Sons, 2006.

[DGG07] Anuj Dawar, Philippa Gardner, and Giorgio Ghelli. Expressiveness and complexity of graph logic. *Inf. Comput.*, 205(3):263–310, 2007.

[Hel89] N. Helft. Induction as nonmonotonic inference. In *Proc. First International Conference on Principles of Knowledge Representation and Reasoning*, pages 149–156, San Mateo, CA, 1989. Morgan Kaufmann.

[KN92] Deepak Kapur and Paliath Narendran. Complexity of unification problems with associative-commutative operators. *Journal of Automated Reasoning*, 9(2):261–288, 1992.

[Mal03] Donato Malerba. Learning recursive theories in the normal ILP setting. *Fundam. Inf.*, 57(1):39–77, 2003.

[MSS00] Satoru Miyano, Ayumi Shinohara, and Takeshi Shinohara. Polynomial-time learning of elementary formal systems. *New Generation Computing*, 18:217–242, 2000.

[RD94] Luc De Raedt and Sašo Džeroski. First-order jk-clausal theories are PAC-learnable. *Artif. Intell.*, 70(1-2):375–392, 1994.

[RLD93] L. De Raedt, N. Lavrač, and S. Džeroski. Multiple predicate learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1993.

[Sha83] Ehud Shapiro. *Algorithmic Program Debugging*. Cambridge, MA, MIT Press, 1983.

Experiments with Czech Linguistic Data and ILP

Jan Dědek[1], Alan Eckhardt[1,2], Peter Vojtáš[1,2]

[1] Department of Software Engineering, Charles University in Prague, Czech Republic
[2] Institute of Computer Science, Czech Academy of Science
{dedek, eckhardt, vojtas}@ksi.mff.cuni.cz

Abstract. In this paper we present basic experiments that we have made in connection with our research in the domain of the Semantic Web. These experiments should demonstrate possibilities of employing ILP technique in the task of acquisition of semantic information from text of Czech Web pages. These experiments are preceded by complex linguistic analysis of the texts and the output of linguistic tools is processed in the ILP procedure.

Keywords: ILP, web, semantics, linguistics, text processing, Czech language.

1 Introduction

Our long-term aim is to extract semantic information from natural language texts. There is a large amount of texts publicly available on internet, but it is difficult to process it using a machine or software agent as it is suggested in the idea of the Semantic Web. These texts are suitable for human reader, who has to read them. Automatic machine processing of information hidden in the texts is very problematic.

The extraction of structured information from the text is often based on linguistic methods and we also process the texts by linguistic tools and use the resulting linguistic trees of sentences for easier extraction of information with help of ILP.

In Section 2 we describe basic methodology of our extraction. Experiments are described in Section 3 and in Section 4 is a conclusion.

2 Methodology

Our extraction method exploits several linguistic tools developed mainly in the Institute of Formal and Applied Linguistics[1] in Prague, Czech Republic. These tools produce domain independent linguistic annotation. The linguistic annotation is then used in the extraction process. The extraction process is supported with extraction rules, which are learned in an ILP procedure. In this paper we present initial experiments with learning of extraction rules. After all semantic interpretation of extraction rules can provide semantics to the extracted data.

[1] http://ufal.mff.cuni.cz

Domain independent intermediate linguistic annotation.

We use a chain of linguistic analyzers ([1], [2], and [3]) that process text presented on a web page and produce linguistic (syntactic) trees corresponding to particular sentences. These trees serve as a basis of our extraction process.

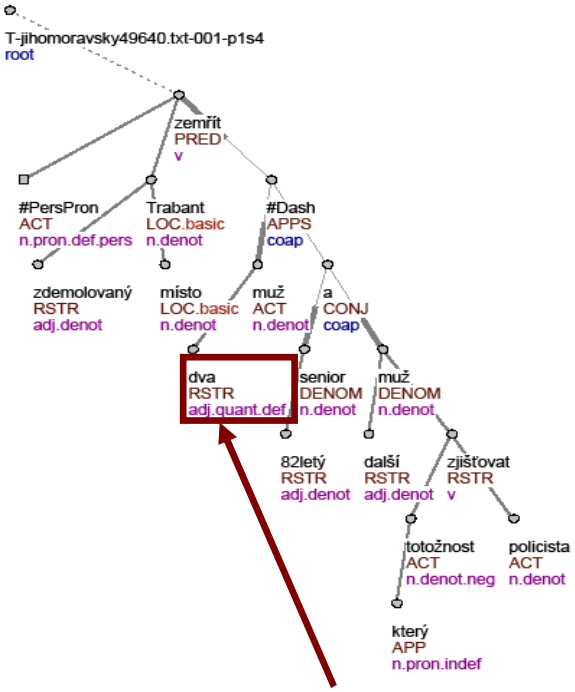


Fig. 1. A tectogrammatical tree of the sentence: "Two men died on the spot in the demolished Trabant..."

Unlike the usual approaches to the description of English syntax, the Czech syntactic descriptions are dependency-based, which means that every edge of a syntactic tree captures the relation of dependency between a governor and its dependent node. Especially the tectogrammatical (deep syntactic) level of representation [4] is closer to the meaning of the sentence. The tectogrammatical trees (see Figure 1) have a very convenient property of containing just the type of information we need for our purpose, namely the information about inner participants of verbs - actor, patient, addressee etc.

On the Figure 1 we can see the relationship between particular words of a sentence and nodes of tectogrammatical tree – the highlighted node of the tree corresponds with the word "two" in the sentence (also highlighted).

Domain and purpose dependent extraction.

Assume we have pages annotated by the linguistic annotator and we have a domain ontology. The extraction method we have used is based on extraction rules. An example of such an extraction rule is on Figure 2 (on the left side). These rules represent common structural patterns that occur in sentences (more precisely in corresponding trees) with the same or similar meaning. Mapping of the extraction

rules to the concepts of target ontology would enable the semantic extraction. Example of such mapping is demonstrated in the Figure 2.

We experimented with obtaining extraction rules in two ways.
(1) Rules and mappings were designed manually (like the rule on the Fig. 2) and
(2) Rules and mappings were learned using ILP methods.

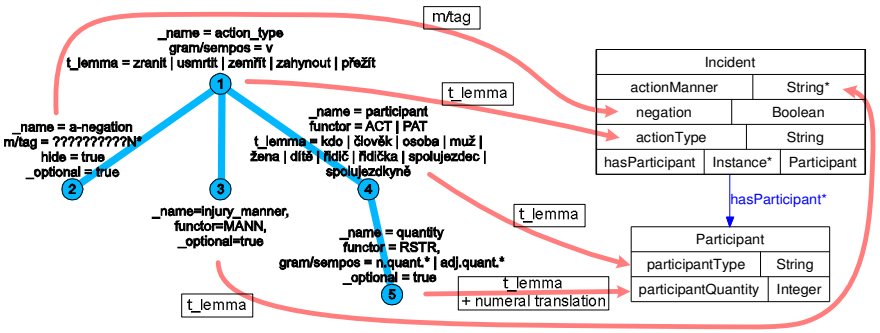


Fig. 2. An example of extraction rule and its mapping to ontology

Here we will present our experiments with learning of the rules. We discovered that we can use a straightforward transformation of linguistic trees to predicates of ILP (for example see Figure 3). Logic representation of a tree consists of three parts:

1. **nodes** (represented as atoms `nodeX_Y`)
2. **edges** (represented by predicate `edge`) and
3. **attributes** of nodes (additional predicates e.g. `t_lemma`, `functor`, `m_tagX`).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Two men died on the spot in the % demolished trabant - a senior 82
% years old and another man, who's

%%%%%%%%% Nodes %%%%%%%%
tree_root(node0_0). node(node0_0).
id(node0_0, t_jihomoravsky49640_txt_001_p1s4).

node(node0_1).
t_lemma(node0_1, zemrit).
functor(node0_1, pred).
gram_sempos(node0_1, v).

node(node0_2).
t_lemma(node0_2, x_perspron).
functor(node0_2, act).
gram_sempos(node0_2, n_pron_def_pers).
...

%%%%%%%%% Edges %%%%%%%%%
edge(node0_0, node0_1). edge(node0_1, node0_2). edge(node0_1, node0_3).
...
edge(node0_34, node0_35).

%%%%%%%%% Injury %%%%%%%%%
injured(t_jihomoravsky49640_txt_001_p1s4).
```

Fig. 3. A sample of the prolog representation of a sentence

3 Experiments

We tried to extract some semantic information from natural language texts. The aim was to find out the number of injured persons during car accidents. Firemen reports have been used; some of them were about car accidents, some were not. Each report was split into sentences and each sentence was linguistically analyzed and transformed into a tectogrammatical tree, as described in the previous section. These trees were converted to a set of the Prolog facts (see Figure 3).

Sentences, which talk about an injury during a car accident, were manually tagged by predicate `injured(X)`, where X is the id of the sentence. Those sentences that do not talk about injured persons during a car accident were tagged as `:- injured(X)`, which represents a negative example. This tagging can be done even by a user inexperienced in linguistics and ILP.

These tagged sentences formed the input for ILP; we used 22 sentences as positive examples and 13 as negative examples. We used Progol [5] as ILP software.

The rules ILP found are in Table 1.

Table 1. Rules found by ILP.

`injured(A) :- id(B,A), id(B,t_plzensky57770_txt_001_p5s2).`
`injured(A) :- id(B,A), id(B,t_plzensky60375_txt_001_p1s6).`
`injured(A) :- id(B,A), id(B,t_plzensky57870_txt_001_p8s2).`
`injured(A) :- id(B,A), id(B,t_plzensky57870_txt_001_p1s1).`
`injured(A) :- id(B,A), edge(B,C), edge(C,D), t_lemma(D,zranit).`
`injured(A) :- id(B,A), edge(B,C), edge(C,D), t_lemma(D,nehoda).`

The first four rules are over-fitted to match specific sentences. Only the last two represent generally applicable rules. But they do make sense – "zranit" means "to hurt" and "nehoda" means "an accident" in Czech. These two rules mean that the root element is connected either to a noun that means an accident or to a verb that means to hurt.

We tested these rules on a set of 15 positive and 23 negative examples. Results are in Table 2, overall accuracy was 86.84%. P is a sentence that ILP classified as positive, ~P is a negative sentence, A is a sentence that was manually tagged as positive and ~A is a sentence tagged as negative. Out of 15 sentences that were tagged as positive, 11 were also classified as positive by ILP and 4 were classified as negative. Out of 23 sentences that were tagged as negative, 22 were also classified as negative by ILP and only 1 was classified as positive.

Table 2. Results on the test set.

	A	~A
P	11	1
~P	4	22

We also tried to find the number of injured in each sentence. Each sentence was tagged by binary predicates `number_injured` and `number_severe_injury`. The first parameter is the id of a sentence; the second is the correct number of persons involved. We also added artificial negative examples with wrong numbers – we used

the numbers present in the dataset: 1.2.3.4.5.6.7.8.9. Note that this task is more difficult than the previous one, because the number is sometimes not mentioned explicitly in the sentence. For example the sentence "both men died in the accident" speaks about two men, but the word "two" is not present in the sentence.

Two following rules were mined (see Table 3). We have excluded twenty rules – each covered only one sentence. These mined rules (in the Table 3) are quite interesting because they are not meaningless. Especially the second one is nearly exemplary. This second rule contains path form the root to the analytical leaf node E:

root_id(A)→node(B) →node(C) →node(D) →analytical_node(E).

Node D is connected in the role of *patient* – affected object (`functor(D,pat)`) and in the morphological description in the node E there is restriction on *singular number* of the node D (`m_tag3(E,s)`). So there should be only **one** participant of the injury.

Table 3. Rules found by ILP.

`number_injured(A,1) :- id(B,A), edge(B,C), edge(C,D), edge(D,E), edge(E,F), m_tag2(F,m).`
`number_injured(A,1) :- id(B,A), edge(B,C), edge(C,D), edge(D,E), functor(D,pat), m_tag3(E,s).`

The results in Table 4 look promising, but the high accuracy is attended thanks to the high number of negative examples. Overall accuracy was 96,30%.

However, ILP has succeeded to classify correctly 4 out of 6 positive examples.

Table 4. Results on the test set using only 6 sentences with a number of injured present.

	A	~A
P	4	0
~P	2	48

When we look at the predicate `number_severe_injury`, the situation is about the same. Only one useful rule was mined (leaving out three rules with ids):

Table 5. A rule found by ILP.

`number_severe_injury(A,1) :- id(B,A), edge(B,C), edge(C,D), edge(D,E), m_tag3(E,s), m_tag4(E,1).`

And the results are also similar:

Table 6. Results on the test set using only 8 sentences with a number of severe injuries present.

	A	~A
P	4	1
~P	4	64

Now only 4 out of 8 positive examples were classified correctly. Overall accuracy was 93.15%.

4 Conclusion

The main contribution of this paper was to try ILP as a learning procedure of extraction rules in our information extraction process supported by linguistic preprocessing of text data. We used a complex set of linguistic tools to transform natural language to linguistic trees; these trees were afterwards transformed into a Prolog representation.

The performance of ILP is questionable – we did use rather small data set for learning and results are not astonishing. However, the preliminary results are not desperate – some rules were found and their performance was not too bad. For real-world use, many optimizations should be made.

In future, we would like to do more experiments in different domains and extend the set of extraction tasks. We also would like to test this approach on a larger data set and study the dependence of the size of the training data and the time of the inductive process. Also, some ILP systems other than Progol can be used.

Acknowledgment
This work was partially supported by Czech projects 1ET100300517, 1ET100300419 and MSM-0021620838.

References

1. Jan Hajič. Morphological Tagging: Data vs. Dictionaries. In: Proceedings of the 6th Applied Natural Language Processing and the 1st NAACL Conference, Seattle, Washington, 2000, pp. 94-101.
2. Michael Collins, Jan Hajic, Eric Brill, Lance Ramshaw, and Christoph Tillmann. A Statistical Parser of Czech, in Proceedings of 37th ACL Conference, pp. 505–512, University of Maryland, College Park, USA, 1999.
3. Vaclav Klimes. Transformation-Based Tectogrammatical Analysis of Czech, in Proceedings of the 9th International Conference, TSD 2006, number 4188 in Lecture Notes In Computer Science, pp. 135–142, Springer-Verlag Berlin Heidelberg, 2006.
4. Marie Mikulova, Alevtina Bemova, Jan Hajic, Eva Hajicova, Jirı Havelka, Veronika Kolarova, Lucie Kucova, Marketa Lopatkova, Petr Pajas, Jarmila Panevova, Magda Razımova, Petr Sgall, Jan Stepanek, Zdenka Uresova, Katerina Vesela and Zdenek Zabokrtsky. Annotation on the tectogrammatical level in the Prague Dependency Treebank. Annotation manual, Technical Report 30, UFAL MFF UK, Prague, Czech Rep. 2006.
5. S. Muggleton. Learning from positive data. In Proceedings of the Inductive Logic Programming Workshop, 1996.

Network Analysis of the ILPnet2 Co-authorship Network

Qingyi Gao and Peter Flach

Department of Computer Science, University of Bristol, Bristol, BS8 1UB, UK

Abstract. In this paper we study the ILPnet2 co-authorship network. The ILPnet2 on-line library (www.cs.bris.ac.uk/ILPnet2/Tools/Reports/) is a repository of more than 1,000 ILP-related articles by well over 500 authors, published between 1970 and 2003. Co-authorship networks constitute a specific view on bibliographic data, in which scientific publications are modeled as vertices, and two vertices are connected by an undirected edge whenever the two corresponding papers share at least one author. We investigate the largest connected component in this network, which contains 816 papers by 526 different authors. Properties of interest include degree distribution and degree centrality and PageRank. We furthermore study the community structure in this network by applying the Newman-Girvan algorithm. Our main conclusion is that, even with restricted information based purely on co-authorship, bibliographic network analysis can reveal useful and interesting patterns.

1 Introduction

Network analysis has attracted considerable attention these years. It covers a wide range of domains ranging from abstract space such as social networks of collaborations, coauthors network [2] and acquaintance network [6], to tangible networks like World Wide Web, biological networks or transportation network [3], to name but a few.

Mathematically, a network or graph consists of set of nodes or vertices together with a set of pairs of vertices called edges. Alternatively, they can be represented by an adjacency matrix. An adjacency matrix of a network with n vertices is the $n \times n$ matrix A with $A_{i,j}$ if there is an edge from vertex i to vertex j appears, and 0 otherwise.

Bibliographic data can induce various types of network, such as collaboration networks, co-citation networks and co-authorship networks. In collaboration networks authors are vertices and edges indicate scientific collaboration. Two authors have a collaboration relationship if they have published at least one paper together. Clearly, collaboration is a symmetric relation, which means that the network is undirected: an edge (u, v) exists if and only if (v, u) exists. Undirected networks have symmetric adjacency matrices. Citation networks have papers as vertices and (directed) edges indicate that one paper cites another. In a co-authorship network, which is the subject of this paper, vertices indicate papers

and undirected edges between vertices represent that papers share at least one joint author.

We investigate the largest connected component in this network, which contains 816 papers by 526 different authors. This component has 38,372 edges out of a possible 816^2, which is a relatively high 5.76%. This means that the network is comparatively dense, which is also borne out by the fact that the average distance (number of edges in a shortest path or geodesic) between any two vertices is 3.37. The diameter of the network, i.e., the longest geodesic, is 9; there are 111 paper pairs with this distance.

One question that we can ask in such a network is how important or central a vertex is. Centrality measures have been defined in social network theory as indices of prestige, prominence, importance, and power-the four Ps. [1]. In this paper we concentrate on degree centrality and PageRank centrality. Degree centrality represents the simplest definition of centrality and ranks vertices by their numbers of neighbors. PageRank centrality refines that further by trying to estimate how influential a vertex is.

2 Degree Centrality

The degree (or degree centrality) of a vertex in a network is the number of edges connected to that vertex.We can simply calculate the vector of degree centralities by multiplying the adjacency matrix with a matrix containing all ones. Figure 1 shows how many papers corresponding to a given degree. We can observe a few small plateaus in this plot, which would typically indicate groups of papers with the same authors and hence the same degrees. Figure 2 indicates cumulative degree distribution of ILPNet2, whereas Figure 3 illustrates the doubly logarithm of cumulative degree distribution of ILPNet2, Figure 4 introduces the results of using linear regression, the slope of equation is around -7. The top 10 Degree Centrality papers are as follows (The first number indicates the degree of each paper):

1. 195: L. De Raedt; S. Muggleton; - Inductive Logic Programming: Theory and Methods; - 1994
2. 176: C. Moure; M. Molina; N. Jacobs; S. Dzeroski; S. Muggleton; W. Van Laer - Detecting Traffic Problems with ILP; - 1998
3. 174: L. De Raedt; N. Lavrac; S. Dzeroski - Multiple predicate learning; - 1993
4. 174: L. De Raedt; N. Lavrac; S. Dzeroski - Multiple predicate learning; - 1993
5. 170: I. Bratko; S. Dzeroski; S. Muggleton - Applications of inductive logic programming; - 1995
6. 170: H. Blockeel; K. Driessens; L. De Raedt; S. Dzeroski - Relational Reinforcement Learning; - 1998

7. 162: S. Dzeroski; S. Muggleton; S. Russell - PAC-learnability of determinate logic programs; - 1992
8. 162: S. Dzeroski; S. Muggleton; S. Russell - PAC-learnability of constrained nonrecursive logic programs; - 1992
9. 162: S. Dzeroski; S. Muggleton; S. Russell - Learnability of constrained logic programs; - 1993
10. 156: D. Kazakov; D. Zupanic; L. Todorovski; N. Lavrac; O. Stepankova; P. Flach; S. Dzeroski; S. Weber - Internet Resources on ILP for KDD; - 2001

The top 10 authors in the top 50 degree centrality papers are (the number in brackets refers to the number of top 50 papers they are involved in): S. Muggleton (31), A. Srinivasan (21), L. De Raedt (16), S. Dzeroski (14), N. Lavrac (8), H. Blockeel (6), D. Page (5), M. Bain (5), P. Flach (3) and J. Ramon (3). In order to get a better understanding of the degree distribution, Figure 1 plots a histogram. It is clear that this distribution is very spiky. In fact, several of these spikes can be attributed to specific authors or sets of authors, as we now show. Consider the highest spike of 38 papers having degree 102. These 38 papers have the following authorships: S. Muggleton (27), S. Muggleton, M. Sternberg (2), S. Muggleton, A. Tamaddoni-Nezhad (2), S. Muggleton, K. Khan, R. Parson (2), S. Muggleton, J. Firth (1), S. Muggleton, S. Colton (1), S. Muggleton, A. Puech (1), S. Muggleton, Parson (1), and B. Swennen, C. Ghil, E. Bleken, L. De Raedt, M. Bruynooghe, V. Coget (1). In other words, 37 out of the 38 papers are either by S. Muggleton as single author, or by S. Muggleton together with other authors who have only published with S. Muggleton (otherwise their degree would be higher than 102). The only exception is the last paper, which can be seen as a coincidence. The spike can thus clearly be attributed to S. Muggleton.

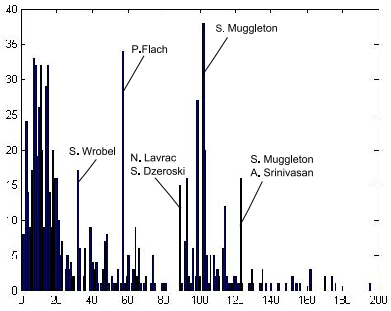


Fig. 1. General degree distribution

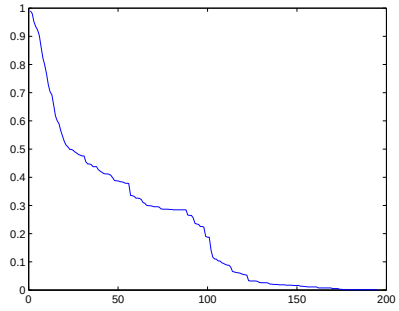


Fig. 2. Cumulative degree distribution

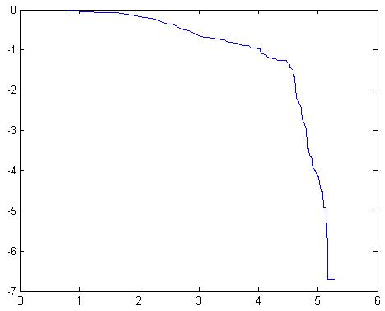


Fig. 3. Doubly logarithm plot

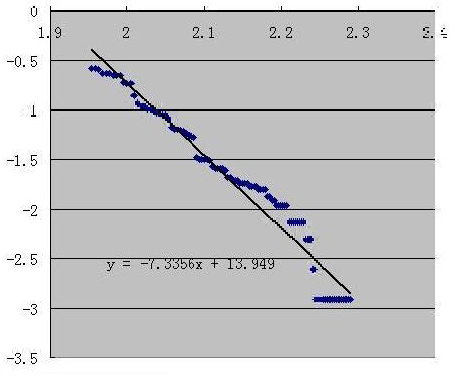


Fig. 4. Power law equation

3 PageRank Centrality

PageRank centrality differs from eigenvector centrality in that each vertex can propagate only a fixed total amount of centrality to its neighbors. In other words, if a vertex has d neighbors and centrality x, then x/d is propagated to its neighbors. Furthermore, each node is given a small amount of centrality to begin with, regulated by a parameter α. Formally, let M by a probability transition matrix obtained from A by renormalising each row to sum to 1, and let U be a transition matrix of uniform transition probabilities. The transition matrix $\alpha U + (1-\alpha)M$ establishes a Matrix chain, and the principal eigenvector of this transition matrix, which defines the stationary distribution of the Markov chain, is defined as the vector of PageRank centralities [5].

Figure 5 shows all papers sorted on decreasing PageRank centrality. It is interesting to note that we obtain a much gentler slope than with degree, and that particularly at the low end. PageRank centrality generally distinguishes papers well.

The top 10 PageRank Centrality papers are as follows (The first number indicates the PageRank centrality value of each paper):

1. 0.1013: D. Kazakov; D. Zupanic; L. Todorovski; N. Lavrac ; O. Stepankova; P. Flach; S. Dzeroski; S. Weber - Internet Resources on ILP for KDD; - 2001
2. 0.0809: F. Zelezny; M. Krogel; N. Lavrac; P. Flach; S. Rawles ;S. Wrobel - Comparative Evaluation of Approaches to Propositionalization; - 2003
3. 0.0786: B. Tausend; C. Nedellec; C. Rouveirol; F. Bergadano ; H. Adz - Declarative Bias in ILP; - 1996
4. 0.0785: D. Page; L. De Raedt; S. Wrobel - Guest Editorial; - 2001
5. 0.0767: L. De Raedt; S. Muggleton - Inductive Logic Programming: Theory and Methods; - 1994
6. 0.0747: L. De Raedt; N. Lavrac; S. Dzeroski - Multiple Predicate Learning; - 1993

7. 0.0747: L. De Raedt; N. Lavrac; S. Dzeroski- Multiple Predicate Learning; - 1993
8. 0.0742: C. Moure; M. Molina; N. Jacobs; S. Dzeroski; S. Muggleton; W. Van Laer - Detecting Traffic Problems with ILP; - 1998
9. 0.0742: H. Blockeel; K. Driessens; L. De Raedt; S. Dzeroski - Relational Reinforcement Learning; - 1998
10. 0.0735: I. Bratko; S. Dzeroski; S. Muggleton - Applications of inductive logic programming; - 1995

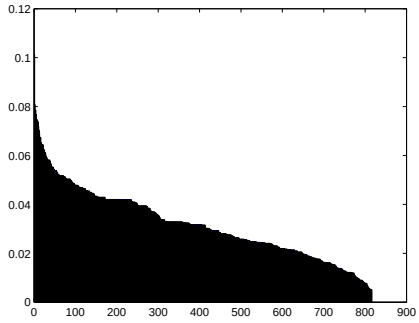


Fig. 5. PageRank Centrality

4 Community Structure

We applied Newmans community structure algorithm to the ILPnet2 co-authorship network. We observed that the algorithm repeatedly finds a smaller cluster and continues splitting the rest. That is, if we were to plot a dendrogram it would always branch in the same direction. Figure 6 gives a graphical depiction of some of the clusters. From the 57 clusters we calculated a modularity value of $Q = 0.5217$. According to Newman [4], most networks fall in the range 0.3 to 0.7. Consequently, $Q = 0.5217$ appears a reasonable split of the graph.

5 Conclusion

In this paper we analyzed the co-authorship network derived from the ILPnet2 on-line library. We have demonstrated that, even though co-authorship information is limited, it can be used to reveal interesting information. For instance, the degree distribution suggests a mixture of two distributions, a pure power law describing the fat tail of high degree papers the giant component and a log-normal distribution describing the low-degree periphery papers. We gave results

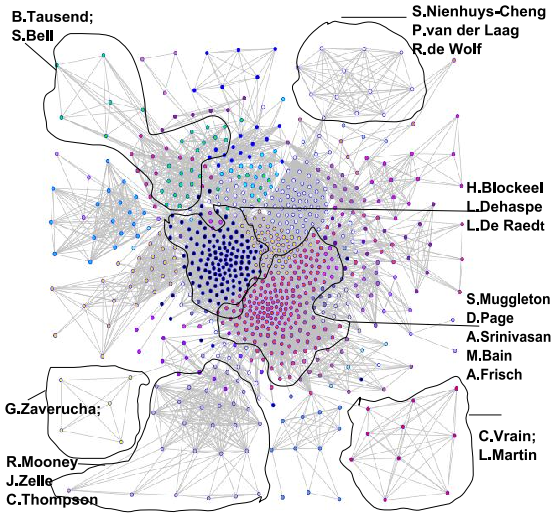


Fig. 6. Community Structure of ILPNet2

of degree centrality and PageRank centrality. Based on these results, we would suggest that the latter gives the most interesting and balanced results, as degree centrality is too simplistic. We have also given the results of a network clustering algorithm. Future work includes assessing the quality of these results with the help of domain experts, and complementing the centrality analysis of ILP papers by means of citation counts. We also plan to investigate the weighted co-authorship network, where the weights are proportional to the number or proportion of shared authors.

References

1. Stephen P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, January 2005.
2. M. E. Newman. Coauthorship networks and patterns of scientific collaboration. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5200–5205, April 2004.
3. M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Physical Review E*, 64(1):016131+, June 2001.
4. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks, August 2003.
5. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
6. S. Wuchty and E. Almaas. Evolutionary cores of domain co-occurrence networks. *BMC Evol Biol*, 5(1), 2005.

Learning Comprehensible Relational Features to Distinguish Subfossil Decapod Crustacean Dactyls

Mark Goadrich and Jeffrey Agnew

Department of Mathematics and Computer Science
Department of Geology

Centenary College of Louisiana

Abstract. Our research explores the application of Inductive Logic Programming to a new domain involving decapod crustacean claws. We find that we can distinguish dactyl shapes by automatically extracting relational features that describe their underlying spatial structure. We first use medial axis techniques to find the shock graph of each dactyl outline, which is then converted into a first-order logic representation capturing the connections, distances and angles between the nodes in this graph. We then use Aleph to find relational classification rules based on the shock graph representations. These relational rules provide a concise and human-understandable way to describe the morphological differences between closely related decapods, and can be seen as a first step to creating automatically learned quantitative taxonomic keys.

1 Introduction

Because decapod crustacean claws are potentially affected by numerous selective agents, they are excellent candidates for evolutionary studies of morphology. Despite being commonly found in shell-rich fossil assemblages, decapod dactyls (i.e., claw movable fingers) are usually ignored because of the assumption that they can be identified only to high taxonomic levels.

However, outline-based and geometric morphometric methods have successfully discriminated the dactyls of sibling species and hybrids of the stone crab *Menippe* [2], closely related species of *Panopeus* [3] and other xanthoid genera including *Cataleptodius*, *Dyspanopeus* and *Eriphia* [1]. Principal component analyses of elliptic Fourier descriptors [6] also have been used to quantify ontogenetic shape trajectories and wear in dactyls [1]. Although these techniques allow statistical tests of differences in dactyl morphologies, dactyl shapes must still be described qualitatively.

Our research introduces a new method for distinguishing dactyl shapes by automatically extracting relational features that describe their underlying spatial structure. Using Aleph [9], an Inductive Logic Programming (ILP) algorithm, we learn general rules that capture informative biological relationships, and find that we can limit overfitting by restricting ourselves to a simple representation of the data.

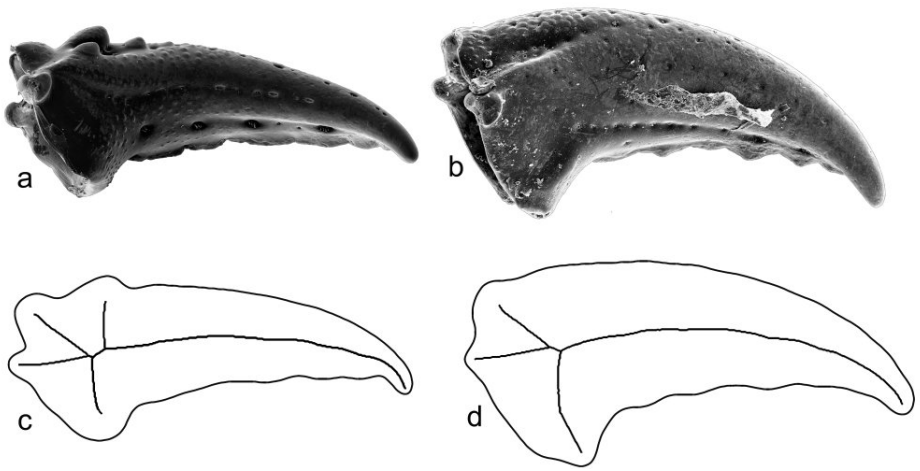


Fig. 1. Scanning Electron Microscope images of *Eriphia gonagra* (a) and *Menippe mercenaria* (b) dactyls, and their corresponding shock graphs, (c) and (d).

2 Dataset Formulation

Our dataset for this study consists of 38 dactyl images, 12 belonging to *Eriphia gonagra* and 26 belonging to *Menippe mercenaria*. Figure 1 (a) and (b) show representative left minor dactyls of these two species. We first use medial axis techniques, used for shape recognition algorithms in computer vision, to find the shock graph of each dactyl outline. Next, these shock graphs are converted into a first-order logic representation capturing the connections, distances and angles between the nodes in each graph.

2.1 Shock Graphs

We begin with the dataset from Agnew [1], where each dactyl image was scaled and aligned using the SHAPE software [5]. To create relational features for each dactyl and expose the underlying skeleton of the images, we chose to convert each image into a shock graph [4] using the flux skeleton implementation of ShapeMatcher [7]. A shock graph is created from a 2D image by first converting the image into an outline. This outline is then thinned along the normal vector according to the calculated flux at each point. Where these normal vectors meet, edges, end points and branch points can be found when looking at the image pixels.

Shock graphs have been used in computer vision as a technique for object recognition; when combined with algorithms for graph similarity, they can help identify when an object has been rotated or distorted over time and space. Sample shock graphs for each species can be found in Figure 1 (c) and (d); note that the top bump in *Eriphia gonagra* creates an edge not seen in *Menippe*

Predicate Type	Predicate Name
Head	eriphia(+example).
Basic	hasNode(+example, -node). hasEdge(+example, -edge, -node, -node). angle(+edge, +edge, -float). distance(+node, +node, -float). (+float)>(+float). (+float)<(+float). (+float)=<(+float). (+float)>=(+float).
Acute	obtuse(+float). acute(+float).
Full	interiorNode(+node). between0and20(+float). . . . between280and300(+float).

Table 1. Background knowledge and modes generated from shock graph representation.

mercenaria graphs. Also note the difference in length and angle of the bottom bump in *M. mercenaria*. We believe this shape representation can be used to qualitatively understand the phenotypic variations present between these two species.

2.2 First-order representation and Aleph

Aleph [9] is a top-down ILP covering algorithm, written completely in Prolog. As input, Aleph takes background knowledge in the form of either intensional or extensional facts, a list of modes declaring how literals can be chained together, and a designation of one literal as the head predicate to be learned. We chose our head predicate to be the smaller class of *Eriphia gonagra* dactyls, and investigate three levels of background knowledge, shown in Table 1.

First, our **basic** extentional facts are based on the shock graph, such that we create two predicates, `hasNode` and `hasEdge`, to connect the nodes and edges with each example. We also calculate the `angle` between each adjacent edge, the `distance` between any two nodes in the graph, and include the predicates of >, <, >= and =< to compare these angles and distances.

The next level of background knowledge, **acute**, includes intensional definitions for `acute`, floating-point numbers less than 90, and `obtuse`, floating-point numbers greater than 90. Finally, the **full** background knowledge level includes a predicate for designating nodes as being adjacent to two other nodes with `interiorNode`, and `between` predicates to generalize floating-point numbers to into bins of size 20, ranging from 0 to a maximum of 300 because of the maximum image size.

```
eriphia(A) :-
    hasEdge(A,B,C,D), hasEdge(A,E,D,F), hasEdge(A,G,F,H), distance(D,H,I),
    distance(F,C,J), J<I, hasEdge(A,K,F,L), distance(L,H,M),
    J<M, distance(L,C,N), J<N, hasEdge(A,O,P,Q),
    distance(Q,H,R), M<R, distance(L,Q,S), J<S.
```

Fig. 2. Sample rule learned from fold 0, which covered 9 positive and 0 negative training examples, and 3 positive and 0 negative testing examples.

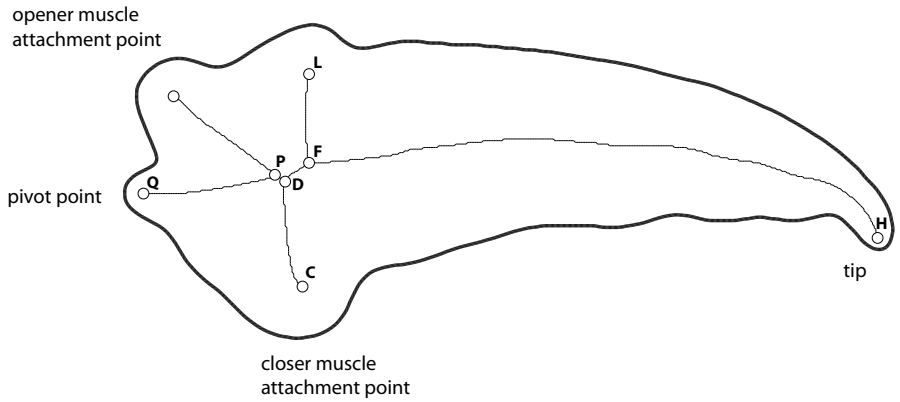


Fig. 3. One possible match of the the nodes C, D, F, H, L, P and Q from the rule in Figure 2 when applied to the first *Eriphia gonagra* example.

3 Experimental Results

We divided the data of 38 examples into five folds of roughly equal size, distributing the positive and negative examples separately to ensure a distribution in each subfold comparable to the complete dataset. In Aleph, we used the **induce** method of exploring and removing seed examples, with the heuristic search method and m-estimate evaluation function, setting m to 20. Other parameter settings changed were to have a minimum accuracy of 0.2, a search depth of 10, a variable-chaining length of 20, a maximum clause length of 20, and a maximum search nodes explored of 20,000.

Figure 2 shows a sample rule learned from fold 0 using only the basic background knowledge. This rule captures all of the positive *Eriphia gonagra* examples and none of the negative examples, in both the training set and testing set. It includes a sequence of connected nodes, C to D to F to H, where the distance between nodes C and F, called J, is less than other calculated distances in this rule. A corresponding distance J is learned in almost all folds, and when this rule is applied to the positive examples, as seen in Figure 3, node C frequently corresponds to the closer muscle insertion point and H to the tip point.

Algorithm	Accuracy	Precision	Recall	F1 Score
Basic	89.4	90.0	75.0	81.8
Acute	86.8	88.8	66.6	76.2
Full	81.6	72.7	66.6	69.6
All Pos	31.6	31.6	100.0	48.0
All Neg	68.4	-	0.0	-

Table 2. Pooled results from five-fold cross-validation experiments, comparing Basic, Acute and Full background knowledge.

We believe natural selection could be acting on the distances in this learned relationship between these areas of the shock graph. Because *Menippe* feeds almost exclusively on hard-shelled prey and *Eriphia* is more of an opportunistic generalist, *Menippe* should have claws with stronger biting forces than *Eriphia* [11]. Our learned rule discusses the length and angle of the closer muscle insertion point in relation to the tip. This relationship is directly related to the mechanical advantage of the claw, such that a shorter length in *E. gonagra* will result in weaker closing strength.

We compare the results of using Aleph and each of the three levels of background knowledge (basic, acute and full) with two baseline algorithms, one which classifies all examples as positive, and another which classifies all examples as negative. The true positive, false positive, true negative and false negative results across the five testsets are pooled to find the overall accuracy, precision, recall and F1 score for each algorithm. These results are reported in Table 2, and we can see Aleph clearly outperforms the baseline algorithms.

When comparing the different levels of background knowledge, we find that simpler is better. The heuristic search employed by Aleph incorporates the additional background knowledge predicates into our learned rules, however, these rules have a lower testset performance and tend to overfit, scoring lower than the basic background knowledge across all evaluation metrics.

4 Conclusions and Future Work

This research demonstrates the feasibility of learning relational features to distinguish between decapod dactyl shapes. By combining techniques from computer vision and ILP, we can learn general rules that are informative to both biologists and paleobiologists, and find that we can limit overfitting by restricting ourselves to a simple representation of the data.

Recent work by Macrini *et al.* [8] extends shock graphs to bone graphs to decrease their brittle dependency on noise variations of the initial shape. We plan to replace shock graphs with bone graphs as the basis for learning, and expect to see increases in our performance as well as more general features.

Suard *et al.* [10] have investigated kernel methods applied to shock graphs. They propositionalize many features of the graphs to create their kernels for the purpose of shape retrieval and image clustering, as opposed to our research

of learning explanatory and discriminatory patterns using the relational graph descriptions. Although our findings point to less background knowledge instead of more, we plan to investigate some of their features and hopefully increase the understandability of our rules without sacrificing their generalization.

Our current dataset is quite small, with test folds having only 2 or 3 positive examples. We plan to further investigate this approach with a larger dataset consisting of 970 major and minor dactyls from nine xanthoid crab species. This dataset will allow us to evaluate whether this method can be used to distinguish dactyls of several closely related species. Also, because many of the dactyls of these species change shape with growth, we can quantify those allometric transformations and identify dactyl sizes where species level differences emerge.

5 Acknowledgements

We would like to thank the authors of the software packages Aleph, SHAPE and ShapeMatcher for the availability of their code.

References

1. J. G. Agnew. *Dactyls Reveal Evolutionary Patterns in Decapod Crustaceans*. PhD thesis, Louisiana State University - Baton Rouge, 2008.
2. J. G. Agnew and L. C. Anderson. Phenotypic differences among sibling species, hybrids and fossils of the stone crab *menippe*. In *Geological Society of America Annual Meeting*, volume 34, page 399, 2002.
3. J. G. Agnew and L. C. Anderson. Inferring diet of crabs using wear patterns on claws. In *Geological Society of America Annual Meeting*, volume 38, page 442, 2006.
4. P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. In *Computer Vision and Pattern Recognition*, 2000.
5. H. Iwata and Y. Ukai. SHAPE: A computer program package for quantitative evaluation of biological shapes based on elliptic fourier descriptors. *The Journal of Heredity*, 93(5):384–5, 2002.
6. F. Kuhl and C. R. Giardina. Elliptic fourier features of a closed contour. *Computer Graphics Image Processing*, 18:236–258, 1982.
7. D. Macrini. Indexing and matching for view-based 3-d object recognition using shock graphs. Master's thesis, University of Toronto, July 2003.
8. D. Macrini, K. Siddiqi, and S. Dickinson. From skeletons to bone graphs: Medial abstraction for object recognition. In *Computer Vision and Pattern Recognition*, 2008.
9. A. Srinivasan. The Aleph Manual Version 5. *http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/*, 2003.
10. F. Suard, A. Rakatomamonjy, and A. Bensrhair. Mining shock graphs with kernels. Technical Report AR-06-01, University of Rouen, Dec 2006.
11. A. B. Williams. *Shrimps, lobsters, and crabs of the Atlantic coast of the eatern United States, Maine to Florida*. Smithsonian Institution Press, Washington, D.C., 1984.

Estimating the Parameters of Probabilistic Databases from Probabilistically Weighted Queries and Proofs [Extended Abstract]

Bernd Gutmann[1], Angelika Kimmig[1], Kristian Kersting[2], and Luc De Raedt[1]

[1] Dept. of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, POBox 2402, BE-3001 Heverlee, Belgium {firstname.lastname}@cs.kuleuven.be
[2] Fraunhofer IAIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany
kristian.kersting@iais.fraunhofer.de

Abstract. We introduce the problem of learning the parameters of the probabilistic database ProbLog. Given the observed success probabilities of a set of queries, we compute the probabilities attached to facts that have a low approximation error on the training examples as well as on unseen examples. Assuming Gaussian error terms on the observed success probabilities, this naturally leads to a least squares optimization problem. Our approach, called LeProbLog, is able to learn both from queries and from proofs and even from both simultaneously. This makes it flexible and allows faster training in domains where the proofs are available. Experiments on real world data show the usefulness and effectiveness of this least squares calibration of probabilistic databases.[1]

1 Introduction

Many real-world application today depend on managing enormous volumes of uncertain data. Such "dirty" databases arise for example when integrating data from various sources, when analyzing social, biological, and chemical networks, within privacy-preserving data mining where only aggregated data is available, and within pervasive computing. These are only some of the many real-world applications showing the abundance of uncertain data and the need for probabilistic databases, i.e., generalizations of traditional relational databases that can deal with uncertainty.

Over the last years, the statistical relational learning community has devoted a lot of attention to learning both the structure and parameters of probabilistic logics, cf. [1, 2], but so far seems to have devoted little attention to the learning of probabilistic database formalisms. Probabilistic databases [3, 4] associate probabilities to facts, indicating the probabilities with which these facts hold. This information is then used to define and compute the success probability of queries

[1] A longer version of this paper has been accepted at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2008) under the title "Parameter Learning in Probabilistic Databases: A Least Squares Approach"

or derived facts or tuples, which are defined using background knowledge (in the form of predicate definitions). As one example, imagine a life scientist mining a large network of biological entities in an interactive querying session. The biological network is a probabilistic graph, in which the edges are represented by probabilistic facts about the biological entities [4]. Interesting questions can then be asked about the probability of the existence of *a* connection between two nodes, or the most reliable path between them.

The key contribution of the present paper is the introduction of a novel setting for learning the parameters of a probabilistic database from examples together with their target probability. The task then is to find those parameters that minimize the least squared error w.r.t. these examples. The examples themselves can either be queries or proofs, where a proof is a conjunction of all facts in the database needed to proof a query by SLD-resolution. This learning setting is then incorporated in the probabilistic database ProbLog [4], though it can easily be integrated in other probabilistic databases as well. This yields the second key contribution of the paper, namely an effective learning algorithm called *LeProbLog* (Least square estimation for ProbLog).

Within the probabilistic database community, parameter estimation has received surprisingly little attention. Nottelmann and Fuhr [5] consider learning probabilistic Datalog rules in a similar setting where the underlying distribution semantics is similar to ProbLog. However, their setting and approach also significantly differ from ours. A single probabilistic target predicate only is estimated whereas we consider estimating the probabilities attached to definitions of multiple predicates. Their approach employs the training probabilities only. Specifically, they generate training examples labeled with 0/1 randomly whereas we use the observed probabilities directly. Finally, the probabilistic database setting differs from the usual statistical relational learning approach in that there is no underlying generative model as for instance at PRISM programs [6].

We proceed as follows. After reviewing ProbLog in Section 2, we will define the parameter estimation problem for probabilistic databases and introduce our least-squares approach LeProbLog for solving it. Before concluding, we will present the results of an extensive set of experiments on a real-world data set.

2 ProbLog

In this work, the probabilistic database employed is ProbLog, a simple probabilistic extension of Prolog introduced in [4]; alternative formalisms that could be used include those of [3] or [5]. We repeat the main ideas of ProbLog here, see [4] for details. A ProbLog program consists of a set of labeled facts $p_i :: c_i$ together with a set of definite clauses encoding *background knowledge* (BK). Each ground instance (that is, each instance not containing variables) of such a fact c_i is true with probability p_i, where all probabilities are assumed mutually independent. The corresponding ProbLog program $T = \{p_1 :: c_1, \cdots, p_n :: c_n\} \cup BK$ defines a probability distribution over sets of facts $L \subseteq L_T = \{c_1, \cdots, c_n\}$ as follows:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i).$$

Based on this distribution, the *success probability* $P_s(q|T)$ of a query q in a ProbLog program T is defined as

$$P_s(q|T) = \sum\nolimits_{L \subseteq L_T} P(q|L) \cdot P(L|T) \,, \tag{1}$$

where $P(q|L) = 1$ if there exists a θ such that $L \models q\theta$, and $P(q|L) = 0$ otherwise. In other words, the success probability of query q corresponds to the probability that the query q is *provable* in a randomly sampled logic program. Alternatively, it can be calculated based on the set of all proofs of query q, see [4] for details. The probability of a *specific* proof, also called explanation, corresponds to that of sampling a logic program L that contains all the facts needed in that proof. The *explanation probability* $P_x(q|T)$ is then defined as the probability of the most likely explanation or proof of the query q:

$$P_x(q|T) = \max\nolimits_{e \in E(q)} P(e|T) = \max\nolimits_{e \in E(q)} \prod\nolimits_{c_i \in e} p_i \tag{2}$$

where $E(q)$ is the set of all explanations for query q [7]. Finally, the k-probability $P_k(q|T)$ approximates the success probability by restricting the set of proofs used in the calculation to the k most likely proofs. Note that $P_1(q|T) = P_x(q|T)$.

3 Parameter Learning in Probabilistic Databases

Data added to a probabilistic database is often uncertain, e.g. information extraction algorithms yield results with probabilities. Furthermore, there is uncertainty about how to explain the data, e.g. there is a connection between genes and diseases but the exact dependency is unknown. This results in a learning task which can be formulated as following:

Definition 1 (Parameter Learning in Probabilistic Databases). ***Given*** *a set of training examples* $\{q_i, \tilde{p}_i\}_{i=1}^{M}$, $M > 0$, *where each* $q_i \in \mathcal{H}$ *is a query or proof and* $\tilde{p}_i$ *is the k-probability of* q_i, ***find*** *a function* $h : \mathcal{H} \to [0, 1]$ *with low approximation error on the training examples as well as on unseen examples.* $\mathcal{H}$ *comprises all parameter assignments for a given database* T.

This framework allows to naturally combine *learning from entailment* and *learning from proofs*, two learning settings that so far have been considered separately. In ProbLog, proofs correspond to conjunctions of probabilistic facts, and can be seen as a conjunction of queries. Therefore, a learning algorithm can use examples of both forms, (atomic) queries and proofs, at the same time. The *error function* that we want to minimize is the mean squared error:

$$MSE(T) = \frac{1}{M} \sum\nolimits_{1 \leq i \leq M} \left(P_s(q_i|T) - \tilde{p}_i \right)^2 . \tag{3}$$

To do so, we use a standard gradient descent approach. To obtain the gradient of the MSE, one has to apply the sum and chain rule to Eq. (3)

$$\frac{\partial MSE(T)}{\partial p_j} = \frac{2}{M} \sum_{1 \leq i \leq M} \underbrace{\left(P_s(q_i|T) - \tilde{p}_i \right)}_{\text{Part 1}} \cdot \underbrace{\frac{\partial \; P_s(q_i|T)}{\partial p_j}}_{\text{Part 2}} . \tag{4}$$

Algorithm 1 Evaluating the gradient of a query efficiently by traversing the corresponding BDD, calculating partial sums, and adding only relevant ones.

function GRADIENT(BDD b, fact to derive for n_j)
  $(val, seen) =$ GRADIENTEVAL$(root(b), n_j)$
  If $seen = 1$ return $val \cdot \sigma(a_j) \cdot (1 - \sigma(a_j))$
  Else return 0
function GRADIENTEVAL(node n, target node n_j)
  If n is the 1-terminal return $(1, 0)$
  If n is the 0-terminal return $(0, 0)$
  Let h and l be the high and low children of n
  $(val(h), seen(h)) =$ GRADIENTEVAL(h, n_j)
  $(val(l), seen(l)) =$ GRADIENTEVAL(l, n_j)
  If $n = n_j$ return $(val(h) - val(l), 1)$
  ElseIf $seen(h) = seen(l)$ return $(\sigma(a_n) \cdot val(h) + (1 - \sigma(a_n)) \cdot val(l), seen(h)))$
  ElseIf $seen(h) = 1$ return $(\sigma(a_n) \cdot val(h), 1)$
  ElseIf $seen(l) = 1$ return $((1 - \sigma(a_n)) \cdot val(l), 1)$

To ensure that all p_j stay between 0 and 1 during gradient descent, we reparameterize the search space and express each $p_j \in]0, 1[$ in terms of the sigmoid function $p_j = \sigma(a_j) := 1/(1 + \exp(-a_j))$ applied to $a_j \in \mathbb{R}$. Part 1 can be calculated by a ProbLog inference call computing (1). Part 2 can be calculated as following

$$\frac{\partial P_s(q_i|T)}{\partial a_j} = \sigma(a_j) \cdot (1 - \sigma(a_j)) \cdot \sum_{\substack{S \subseteq L_T \\ L \models q_i}} \delta_{jS} \prod_{\substack{c_x \in S \\ x \neq j}} \sigma(a_x) \prod_{\substack{c_x \in L_T \setminus S \\ x \neq j}} (1 - \sigma(a_x)).$$

where $\delta_{jS} := 1$ if $c_j \in S$ and $\delta_{jS} := -1$ if $c_j \in L_T \setminus S$. It is derived by first deriving the gradient $\partial P(S|T)/\partial p_j$ for a fixed subset $S \subseteq L_T$ of facts, which is straight-forward, and then summing over all subsets S where q_i can be proven. Going over all subprograms S in the last equation is infeasible. But there is an efficient algorithm to compute $P_s(q_i|T)$ relying on BDDs [4]. We updated this towards the gradient as shown in algorithm 1. LeProbLog combines the BDD-based gradient calculation with a standard gradient descent search. Starting from parameters $\mathbf{a} = a_1, \ldots, a_n$ initialized randomly, the gradient $\Delta\mathbf{a} = \Delta a_1, \ldots, \Delta a_n$ is calculated, parameters are updated by subtracting the gradient, and updating is repeated until convergence. When using the k-probability with finite k, the set of k best proofs may change due to parameter updates. After each update, we therefore recompute the set of proofs and the corresponding BDD.

4 Experiments

We set up experiments to investigate the following questions:

Q1 Does our approach reduce the mean squared error on training and test data?
Q2 Is LeProbLog able to recover the original parameters?

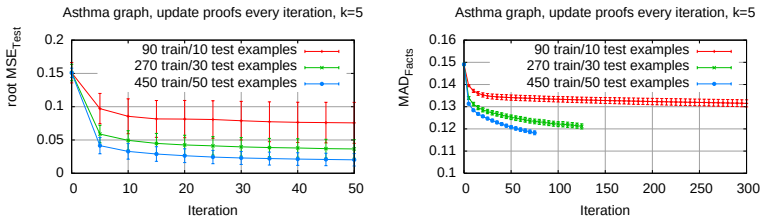


Fig. 1. $\sqrt{MSE_{\text{Test}}}$ (left) and MAD_{facts} (right) for asthma using the 5 best proofs ($k = 5$) (**Q1** and **Q2**)

Q3 Does the algorithm perform better when parts of the training data are given as proof? (For instance the parse tree for a sentences when learning probabilistic grammars, or the most likely path in a probabilistic graph)

We extracted a graph around asthma disease from a collection of databases. We obtained a set of related genes by searching Entrez for human genes annotated with asthma; the asthma phenotype is from OMIM. Weights were assigned to edges as described in [8]. We used 7 randomly chosen asthma genes for graph extraction. The resulting graph contains 127 nodes and 241 edges. From this graph we sampled 500 random node pairs (a,b) and estimated the query probability for path(a,b) using P_5, the probability of the 5 best proofs (for **Q1**, **Q2**). We also sampled 300 node pairs and calculated P_1 for path(a,b), the probability of the best path between a and b, and used it to answer **Q3**. We use both the root mean squared error on the test data and the mean absolute difference MAD_{facts} between learned and original fact probabilities to assess the results.

Q1, Q2: Sanity Check We attach probabilities to queries in the training set based on the best $k = 5$ proofs. The same approximation is used in the gradient descent algorithm. The left graph of Figure 1 shows the evolvement of the root mean squared error. LeProbLog reduces the MSE on both training and test data, with significant differences in all cases (two-tailed t-test, $\alpha = 0.05$). These results affirmatively answer **Q1**. The MAD_{facts} error is reduced as can be seen in the right plot of Figure 1. Again, all differences are significant (two-tailed t-test, $\alpha = 0.05$). Using more training examples results in faster error reduction. This answers **Q2** affirmatively.

Q3: Learning from Proofs and Queries To investigate the effect of using both proofs and queries as examples, we compute the best proof and its probability for 300 examples. For each example, we either use the query or the best proof, both with the probability of the best proof. Learning uses $k = 1$. We use proofs for $0, 50, \ldots, 300$ examples and queries for the remaining ones. Figure 2 shows the results of this experiment. The curve on the left side indicates that the error per fact (MAD_{facts}) goes down faster in terms of iterations when increasing the fraction of proofs. The curve on the right side shows that the root MSE on the test set decreases. These results answer **Q3** affirmatively.

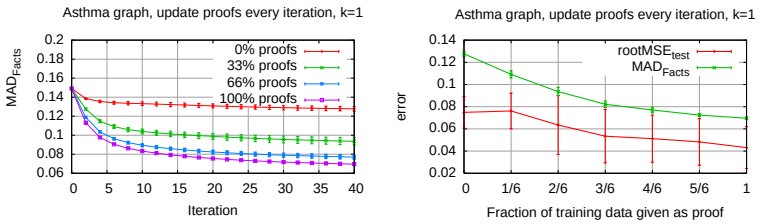


Fig. 2. MAD_{facts} and $\sqrt{MSE_{\text{Test}}}$ after 40 iterations on the asthma graph when different fractions of the data are given as proof (**Q3**)

5 Conclusions

We have introduced an approach to learning the parameters of the probabilistic database ProbLog and successfully shown it at work on a real biological application. Interesting directions for future research include conjugate gradient techniques and regularization-based cost functions. Those enable domain experts to successively refine probabilities of a database by stating training examples.

Acknowledgments

Angelika Kimmig and Bernd Gutmann are supported by the Research Foundation-Flanders (FWO-Vlaanderen), Kristian Kersting by a Fraunhofer ATTRACT fellowship. This work is supported by the GOA project 2008/08 Probabilistic Logic Learning and uses HPC resources `http://ludit.kuleuven.be/hpc`.

References

1. Getoor, L., Taskar, B., eds.: Statistical Relational Learning. The MIT press (2007)
2. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S., eds.: Probabilistic Inductive Logic Programming - Theory and Applications. Volume 4911 of LNAI. Springer (2008)
3. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. In: Proceedings of VLDB. (2004) 864–875
4. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In Veloso, M., ed.: IJCAI. (2007) 2462–2467
5. Nottelmann, H., Fuhr, N.: Learning probabilistic datalog rules for information classification and transformation. In: CIKM, ACM (2001) 387–394
6. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. Artif. Intell. Res. (JAIR) **15** (2001) 391–454
7. Kimmig, A., De Raedt, L., Toivonen, H.: Probabilistic explanation based learning. In: ECML. (2007) 176–187
8. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., Toivonen, H.: Link discovery in graphs derived from biological databases. In: DILS. (2006) 35–49

Propositionalizing the EM algorithm by BDDs

Masakazu Ishihata[1], Yoshitaka Kameya[1], Taisuke Sato[1], and Shin-ichi Minato[2]

[1] Graduate School of Information Science and Engineering,
Tokyo institute of Technology
{ishihata,kameya,sato}@mi.cs.titech.ac.jp
[2] Graduate School of Information Science and Technology,
Hokkaido University
minato@ist.hokudai.ac.jp

Abstract. We propose an EM algorithm working on binary decision diagrams (BDDs). It opens a way to applying BDDs to statistical inference in general and machine learning in particular. We also present the complexity analysis of noisy-OR models.

1 Introduction

Binary decision diagrams (BDDs) have been popular as a basic tool for compactly representing boolean functions [1, 2]. In this paper[3] we present yet another application of BDDs. We propose a new EM algorithm that works on BDDs. Since the EM algorithm is a fundamental parameter learning algorithm for maximum likelihood estimation in statistics [4], our proposal opens a way to apply BDDs to statistical learning in general and to machine learning in particular.

2 The EM algorithm

We here describe our unsupervised learning setting and review the expectation-maximization (EM) algorithm [4]. First of all, we assume our problem domain is modeled with k boolean *random* variables $X_1, X_2, \ldots, X_k$, each taking 1 (true) and 0 (false) independently of each other. Let F be a boolean function composed of these k variables, and assume only the value of F is observable whereas those of the X_i's are not. Hereafter, to make notations simple, F is treated as a boolean random variable as well that takes the value of (the function) F. We then call F an *observable variable*, and the X_i's *basic variables*. The EM algorithm proposed in this paper aims to estimate the probabilities of basic variables being true from the observed values of F.

Let ϕ be an assignment of the set of basic variables $\mathbf{X} = \{X_1, X_2, \ldots, X_k\}$. ϕ maps each variable $X \in \mathbf{X}$ to its value $x \in \{0, 1\}$. We assume $\mathbf{X}$ is partitioned

[3] This is a shortened version of [3], which deals with zero-suppressed BDDs (ZBDDs) as well as BDDs.

into $\mathcal{S}$, sets of i.i.d. variables, and each partition $s \in \mathcal{S}$ has a *parameter* $\theta_{s,x}$, a common probability of $X \in s$ taking x. We use Φ to stand for the set of all assignments. Since the value $F = f \in \{0, 1\}$ is uniquely determined by ϕ, F is a function $F(\phi) = f$ of assignments. Hence the set of assignments which make $F = f$ is written as $F^{-1}(f) = \{\phi \in \Phi \mid F(\phi) = f\}$. We introduce $\sigma_{s,x}(\phi) = |\{X \in s \mid \phi(X) = x\}|$ to denote the total number of i.i.d. variables in the partition s that takes a value x by ϕ. The EM algorithm we develop for the setting described above consists of two steps, the *expectation step* (E-step) and the *maximization step* (M-step), defined as follows:

- **E-step**: Compute the *conditional expectation* $E_{\boldsymbol{\theta}}[\sigma_{s,x}(\cdot) \mid F = f]$ by $\eta^x_{\boldsymbol{\theta}}[s]/P_{\boldsymbol{\theta}}(F = f)$, where:

$$\eta^x_{\boldsymbol{\theta}}[s] = \sum_{\phi \in F^{-1}(f)} \sigma_{s,x}(\phi) \prod_{s' \in \mathcal{S}} \prod_{x' \in \{1,0\}} (\theta_{s',x'})^{\sigma_{s',x'}(\phi)} \tag{1}$$

$$P_{\boldsymbol{\theta}}(F = f) = \sum_{\phi \in F^{-1}(f)} \prod_{s \in \mathcal{S}} \prod_{x \in \{1,0\}} (\theta_{s,x})^{\sigma_{s,x}(\phi)}. \tag{2}$$

- **M-step**: Update $\boldsymbol{\theta}$ to $\hat{\boldsymbol{\theta}}$ by $\hat{\theta}_{s,x} \propto E_{\boldsymbol{\theta}}[\sigma_{s,x}(\cdot) \mid F = f]$.

3 BDDs and the EM algorithm

A BDD is a rooted directed acyclic graph representing a boolean function as a disjunction of exclusive conjunctions. It has two terminal nodes, $\boxed{1}$(true) and $\boxed{0}$(false). Each nonterminal node n is labeled with a binary random variable denoted by $\mathit{Label}(n)$, and has two outgoing edges called 1-edge and 0-edge, indicating that $\mathit{Label}(n)$ takes 1 and 0, respectively. $\mathrm{Ch}^x(n)$ stands for a child node of n, connected by the x-edge ($x \in \{0, 1\}$).

A *reduced ordered BDD (ROBDD)* is a BDD which is a unique representation of the target boolean function. Fig. 1 illustrates some different representations of a boolean function $F = (A \vee B) \wedge \bar{C}$. Fig. 1 (a) is a truth table, in which a row corresponds to an assignment ϕ for $\mathbf{X} = \{A, B, C\}$. One way to obtain the ROBDD for F (Fig. 1 (d)) is to consider a binary decision tree (BDT) (b) and apply two reduction rules, the deletion rule and the merging rule, as many times as possible to reach (d).

Consider a BDT like the one in Fig. 1 (b). In a BDT, there is a unique path π_ϕ from the root to a terminal for an assignment ϕ, in which every basic variable appears once as a node label. We rewrite Eq. 1 to Eq. 3 so that $\eta^x_{\boldsymbol{\theta}}[s]$ is computed on a BDD:

$$\eta^x_{\boldsymbol{\theta}}[s] = \sum_{\pi_\phi : \phi \in F^{-1}(f)} \sum_{n' \in \pi_\phi : L_{n'} \in s} \mathbf{1}_{\phi(L_{n'}) = x} \prod_{n \in \pi_\phi} (\theta_{[L_n]})^{\phi(L_n)} (\theta_{\overline{[L_n]}})^{1 - \phi(L_n)} \tag{3}$$

Here $n \in \pi_\phi$ says that the node n is on the path π_ϕ. L_n is a shorthand for $\mathit{Label}(n)$ and $[L_n]$ is the partition to which L_n belongs. $\mathbf{1}_{\phi(L_{n'}) = x} = 1$ if $\phi(L_{n'}) = x$ is true, and 0 otherwise.

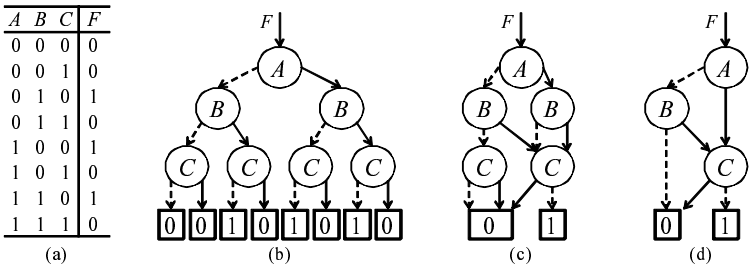


Fig. 1. Examples of (a) a truth table, (b) a binary decision tree (BDT), (c) a BDD which is ordered but is not reduced, (d) the ROBDD, for $F = (A \vee B) \wedge \bar{C}$.

4 The BDD-EM algorithm

We here present the BDD-EM algorithm which is an EM algorithm working on BDDs. There are four auxiliary procedures for the procedure BDD-EM(), i.e. IterateEM(), GetBackward(), GetForward() and GetExpectation().

Procedure: BDD-EM()
  Initialize all parameters $\boldsymbol{\theta}$;
  repeat
    IterateEM();
  until the parameters $\boldsymbol{\theta}$ converge;
end

Procedure: IterateEM()
  // E-step
  GetBackward();
  GetForward();
  GetExpectation();
  // M-step
  for each $s \in \mathcal{S}$ **do**
    $\theta_{s,1} \propto \eta^1_{\boldsymbol{\theta}}[s]/\mathcal{P}^f_{\boldsymbol{\theta}}[F]$;
    $\theta_{s,0} \propto \eta^0_{\boldsymbol{\theta}}[s]/\mathcal{P}^f_{\boldsymbol{\theta}}[F]$;
  end for
end

Procedure: GetBackward()
  $\mathcal{B}^1_{\boldsymbol{\theta}}[\boxed{1}] = 1$, $\mathcal{B}^1_{\boldsymbol{\theta}}[\boxed{0}] = 0$;
  $\mathcal{B}^0_{\boldsymbol{\theta}}[\boxed{1}] = 0$, $\mathcal{B}^0_{\boldsymbol{\theta}}[\boxed{0}] = 1$;
  $\mathcal{N} = \mathrm{Par}(\boxed{1}) \cup \mathrm{Par}(\boxed{0})$;
  // $\mathrm{Par}(n)$: the set of parents of n.
  while $\mathcal{N} \neq \phi$ **do**
    $n = \mathrm{argmax}_{n' \in \mathcal{N}}\, Ord(n')$
    // $Ord(n)$ is the index of $Label(n)$
    // in the variable order.
    $X = Label(n)$;
    $\mathcal{B}^1_{\boldsymbol{\theta}}[n] = \theta_{[X]}\mathcal{B}^1_{\boldsymbol{\theta}}[\mathrm{Ch}^1(n)]$
      $+\theta_{[\bar{X}]}\mathcal{B}^1_{\boldsymbol{\theta}}[\mathrm{Ch}^0(n)]$;
    $\mathcal{B}^0_{\boldsymbol{\theta}}[n] = \theta_{[X]}\mathcal{B}^0_{\boldsymbol{\theta}}[\mathrm{Ch}^1(n)]$
      $+\theta_{[\bar{X}]}\mathcal{B}^0_{\boldsymbol{\theta}}[\mathrm{Ch}^0(n)]$;
    $\mathcal{N} = \mathcal{N} \backslash \{n\} \cup \mathrm{Par}(n)$;
  end while
end

Backward and forward probabilities: We compute backward and forward probabilities like those in hidden Markov models. The procedure GetBackward() calculates *backward probabilities* for each node in the BDD representing F. A *backward probability* $\mathcal{B}^1_{\boldsymbol{\theta}}[n]$ (resp. $\mathcal{B}^0_{\boldsymbol{\theta}}[n]$) is the sum of the probabilities of all paths from node n to $\boxed{1}$(resp. $\boxed{0}$). We set $\mathcal{B}^1_{\boldsymbol{\theta}}[\boxed{1}] = 1$ and $\mathcal{B}^0_{\boldsymbol{\theta}}[\boxed{0}] = 1$ respectively. They are calculated from terminals to the root. Contrastingly the procedure GetForward() calculates *forward probabilities* for each node from the root to terminals. A *forward probability* $\mathcal{F}_{\boldsymbol{\theta}}[n]$ is the sum of the probabilities

```
1: Procedure: GetForward()
2:    InitializeF();
3:    F_θ[root] = 1;
4:    N = {root};
5:    while N ≠ φ do
6:       n = argmin_{n'∈N} Ord(n');
7:       X = Label(n);
8:       F_θ[Ch^1(n)] += F_θ[n]θ_[X];
9:       F_θ[Ch^0(n)] += F_θ[n]θ_[X̄];
10:      N = N\{n} ∪ {Ch^1(n), Ch^0(n)};
11:   end while
12: end
```

```
1: Procedure: GetExpectation()
2:    InitializeEta();
3:    for each n ∈ N do
4:       X = Label(n);
5:       e^1_n = F_θ[n]B^f_θ[Ch^1(n)]θ_[X];
6:       e^0_n = F_θ[n]B^f_θ[Ch^0(n)]θ_[X̄];
7:       η^1_θ[[X]] += e^1_n, kη^0_θ[[X]] += e^0_n;
8:       for each Z ∈ Del^1_Y(n) do
9:          η^1_θ[[Z]] += e^1_n θ_[Z];
10:         η^0_θ[[Z]] += e^1_n θ_[Z̄];
11:      end for
12:      for each Z ∈ Del^0_Y(n) do
13:         η^1_θ[[Z]] += e^0_n θ_[Z];
14:         η^0_θ[[Z]] += e^0_n θ_[Z̄];
15:      end for
16:   end for
17: end
```

```
1: Procedure: GetExpectation*()
2:    InitializeEta();
3:    for each n ∈ N do
4:       X = Label(n);
5:       e^1_n = F_θ[n]B^f_θ[Ch^1(n)]θ_[X]
6:       e^0_n = F_θ[n]B^f_θ[Ch^0(n)]θ_[X̄]
7:       η^1_θ[[X]] += e^1_n;
8:       η^0_θ[[X]] += e^0_n;
9:       X' : Ord(X') = Ord(X) + 1;
10:      ζ[X'] += e^1_n + e^0_n;
11:      ζ[Label(Ch^1(n))] −= e^1_n;
12:      ζ[Label(Ch^0(n))] −= e^0_n;
13:   end for
14:   X = X;
15:   X = argmin_{X'∈X} Ord(X');
16:   z = ζ[X];
17:   X = X\{X};
18:   while X ≠ φ do
19:      X = argmin_{X'∈X} Ord(X');
20:      η^1_θ[[X]] += zθ_[X];
21:      η^0_θ[[X]] += zθ_[X̄];
22:      z += ζ[X];
23:      X = X\{X};
24:   end while
25: end
```

Fig. 2. Improved GetExpectation()

of all paths from the *root* to node n. The procedure InitializeF() initializes $\mathcal{F}_{\boldsymbol{\theta}}[n] = 0$ for all n.

Conditional expectations: The procedure GetExpectation() updates $\eta^x_{\boldsymbol{\theta}}[[X]]$ which is defined in Section 2 for each $X \in \mathbf{X}$. The procedure InitializeEta() sets each $\eta^x_{\boldsymbol{\theta}}[[X]] = 0$. In GetExpectation(), $f \in \{1, 0\}$ is the observed value of F, and $\mathbf{N}$ is the set of all nodes in the BDD.

Note that in order to compute probabilities properly, we need to *recover* deleted nodes. So, to denote the nodes deleted by the deletion rule, $Del^1_Y(n)$ and $Del^0_Y(n)$ are introduced in GetExpectation(). $Del^x_Y(n)$ ($x \in \{1, 0\}$) stands for the set of labels (i.e. variables) of deleted nodes between n and $\mathrm{Ch}^x(n)$. So we have $Del^x_Y(n) = \{X \in \mathbf{V}(\delta_Y) \mid Label(n) \prec X \prec Label(\mathrm{Ch}^x(n))\}$. What we actually use for the computation of conditional expectations is not GetExpectation() however, as it incurs some inefficiency, but GetExpectation*() shown in Fig. 2 which processes computation of the deleted nodes much more efficiently (details omitted).

5 Time complexities for noisy-OR models

The time complexity of building BDDs is NP-hard in general [5]. However, there are efficient techniques to build BDDs using the *Apply operation* [2] and those to find good variable orderings, be they *dynamic* or *static* [5, 6]. So building BDDs can be done efficiently in practice. In this section, we evaluate the time complexity of both building BDDs and running the BDD-EM algorithm for noisy-OR models.[4]

A noisy-OR model represents a relation between multiple causes and an effect. Let F be an observable variable representing an effect, and C_1, C_2 and C_3 basic variables representing possible causes which make F true. While the logical OR relation is represented as $F \Leftrightarrow C_1 \vee C_2 \vee C_3$, the noisy-OR relation allows for a situation where C_1 is true but F is false. For this noisy-OR model, we introduce *inhibition variables*, I_1,I_2 and I_3, which inhibit F to be true with probabilities $\theta_{[I_1]} = P(F=0 \mid C_1=1, C_2=0, C_3=0)$, $\theta_{[I_2]} = P(F=0 \mid C_1=0, C_2=1, C_3=0)$ and $\theta_{[I_3]} = P(F=0 \mid C_1=0, C_2=0, C_3=1)$, respectively. An N-input noisy-OR model between F and $C_1, C_2, \ldots, C_N$ is described by:

$$F = (C_1 \wedge \bar{I}_1) \vee (C_2 \wedge \bar{I}_2) \vee \cdots \vee (C_N \wedge \bar{I}_N).$$

Fig. 3 shows a BDD representing F under the variable ordering *Ord* such that $C_i \prec C_j$, $I_i \prec I_j$ $(i < j)$ and $C_i \prec I_k$ $(i \leq k)$. We construct a BDD from F using the Apply operation, denoted by $\mathrm{Apply}(\delta_X, \delta_Y, \langle \mathrm{op} \rangle)$, that builds a BDD representing $X \langle op \rangle Y$ where δ_X and δ_Y represent the boolean functions X and Y, respectively. Although the time complexity of $\mathrm{Apply}(\delta_X, \delta_Y, \langle \mathrm{op} \rangle)$ is $O(N_X N_Y)$ in general, where N_X (resp. N_Y) is the number of nodes in the BDD representing X (resp. Y), we can see an application of $\mathrm{Apply}(\cdot)$ for an N-input noisy-OR model takes just $O(1)$. So the BDD is obtained by applying the Apply operation N times, and the time complexity becomes $O(N)$ under *Ord*. Also the time complexity of the E-step is $O(N)$ because $|\mathbf{N}| = 2N$ and $|\mathbf{X}| = 2N$.

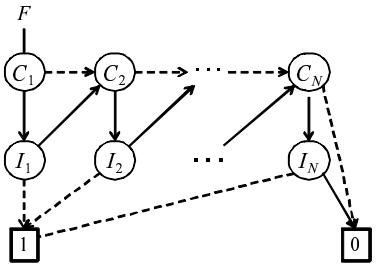


Fig. 3. A BDD representing the noisy-OR model.

[4] We confirmed the BDD-EM algorithm properly converges by numerical experiments.

6 Related work and concluding remarks

We have presented an EM algorithm that works on BDDs. Our work is considered as a succession to the previous work done by Minato et al. [7]. It shows how to compile BNs into ZBDDs to compute probabilities but probability learning is left untouched. In [3], we supplemented a necessary algorithm to apply ZBDDs to EM learning.

The introduction of BDDs solves a long-standing problem of PRISM [8], a logic-based language for generative modeling. It employs a propositionalized data structure called *explanation graph*s similar to decomposed BDDs to represent boolean formulas in disjunctive normal form. The current PRISM however assumes the *exclusiveness condition* that the disjuncts are exclusive to make sum-product probability computation possible. Since the proposed algorithms are applicable to explanation graphs as well, it allows PRISM to abolish the exclusiveness condition.

ProbLog is a recent logic-based formalism that computes probabilities via BDDs [9]. A ProbLog program computes the probability of a query atom from a disjunction of conjunctions made up of independent probabilistic atoms by converting the disjunction to a BDD and applying the sum-product computation to it. [5] Since our BDD-EM algorithm works on BDDs, integrating it with ProbLog for probability learning seems an interesting future research topic.

References

1. Akers, S.B.: Binary decision diagrams. IEEE Trans. Computers **27**(6) (1978) 509–516
2. Bryant, R.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers **35**(8) (1986) 677–691
3. Ishihata, M., Kameya, Y., Sato, T., Minato, S.: Propositionalizing the EM algorithm by BDDs. Technical Report TR08-0004, Dept. of Computer Science, Tokyo Institute of Technology (2008)
4. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. of the Royal Statistical Society B **39** (1977) 1–38
5. Drechsler, R., Sieling, D.: Binary decision diagrams in theory and practice. Int'l J. on Software Tools for Technology Transfer **3** (2001) 112–136
6. Minato, S., Ishiura, N., Yajima, S.: Shared binary decision diagram with attributed edges. Proc. ACM/IEEE Design Automation Conf. (1990) 52–57
7. Minato, S., Satoh, K., Sato, T.: Compiling Bayesian networks by symbolic probability calculation based on Zero-suppressed BDDs. In: Proc. of IJCAI'07. (2007) 2550–2555
8. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. of Artificial Intelligence Research **15** (2001) 391–454
9. De Raedt, L., Angelika, K., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discoverry. In: Proc. of IJCAI'07. (2007) 2468–2473

[5] It should be noted that a special treatment is required for the computation of conditional expectations (see [3] for details).

Using Bio-Pathways in Relational Learning

Matěj Holec[1], Filip Železný[1], Jiří Kléma[1], Jiří Svoboda[1] and Jakub Tolar[2]

[1]Czech Technical University, Prague
e-mail: {holecm1, zelezny, klema, svoboj1}@fel.cvut.cz
[2]University of Minnesota, Minneapolis
e-mail: tolar003@umn.edu

Abstract. In the present work we compile expression and pathway data related to a specific biological classification problem, into a relational database in Prolog, providing benchmarking material for ILP experimentation. We also review the principal pitfalls arising in the attempts to connect the two sources of knowledge through the relational formalism.

1 Introduction

In the last ten years, gene expression data measured by high-throughput technologies such as DNA microarrays [2] have become an important challenge for machine learning. The typical approaches include the construction of classifiers for tissue or disease class from the expression data (the paper [3] representing the first significant success story), or using clustering algorithms for discovering previously unknown classes characterized by distinguished expression profiles.

To boost the explanatory power of gene expression based classifiers, relevant biological background knowledge must be integrated into the learning process. Inductive logic programming (ILP) here thus offers itself as a suitable tool for data analysis. The review paper [5] indicated several ways in which relevant genomic background knowledge, such as primary gene structure (the order of DNA bases in the gene), could be exploited. The paper [7] presented an application of ILP for inducing relational descriptions of groups of co-expressed genes. The descriptions were first-order conjunctions addressing genes' joint relationships to functions, processes or cellular locations formalized by the gene ontology [1]. The descriptions also pertained to mutual gene relationships derived from another piece of background knowledge – the database of known gene interactions.

In extensive discussions of the results of the study [7] with biologists, we perpetually received the feedback that Gene Ontology terms, acting as the primary vocabulary for expressing classifiers, are overly general to allow for any serious biological interpretation, let alone experimental validation. The similar held for rules conditioned on gene-interaction assumptions. Offering instead to use a language addressing the properties of gene primary structure, we seemed to have jumped to the opposite extreme, in that this level of information would be too specific to allow interpretation in terms of systems biology.

[1] http://www.geneontology.org/

In this work we try to exploit as background knowledge the descriptions of known biochemical pathways (metabolic, gene regulatory, signalling) which appear to possess just the right level of generality for a biologist. In their simplest form abstracting from reaction stochiometry and kinetics, pathways may be seen as directed graphs with labeled nodes representing compounds (enzymes, metabolites, etc.) and labeled edges standing for various relationships among them (inhibition, activation, reaction product, etc.). Previous work in ILP [6] exists, where metabolic networks were constructed or completed from empirical data. We are however not aware of previous ILP research exploiting pathway descriptions as background knowledge incorporated for sakes of gene-expression based classification.

In the general area of bioinformatics, pathways are already recognized as units exploitable for prediction tasks in gene expression mining. However, the technical means actually provided for their exploitation in current gene expression analysis systems[2] are surprisingly stone-age. In particular, given a sample of expression values and a pathway, the systems calculate the "pathway expression". This is the average of the expressions of genes which code for enzymes present in the pathway. The researcher then looks for the pathways most over- or under-expressed for a given set of samples, compared to a control sample group.

This approach offends biological intuition in several ways, out of which the most important is that a pathway rarely activates as a whole. Usually, it is proper subgraphs of the pathway graphs whose expression may be specific for a given biological condition. In fact, the very division of the cell processes into separate units called pathways is rather arbitrary and has been guided by human convenience rather than any systematic method of network clustering. The recent paper [4] demonstrated a systematic way to extract pathway subgraphs called *fully coupled fluxes* (FCF). It was shown that FCF's comply better than pathways to the intuitive notion of "working units" as the correlation of gene expression in FCF's is larger than in pathways.

ILP systems have in principle all the necessary means to identify the pathway fragments relevant for a given classification task, without much human intervention. With this paper we thus wish to stimulate ILP researchers to explore ways in which this can be done, i.e. how to best exploit pathway information as background knowledge for gene-expression based classification. Note that this task is more ambitious than ordinary search for over(under)-expressed pathway subgraphs. For example, recursion may prove as a suitable syntactic instrument to express that *paths* of certain properties in pathway graphs are typically activated in a given biological condition.

2 Synthesis of the ILP input data

The biological problem motivating the ILP task is to distinguish between two types of cell tissues produced in bone marrow important in blood forming, so

[2] the Expression Profiler available from the European Bioinformatics Institute, or the DAVID system provided by the US National Center for Biotechnology Information

called *stromal* cells and *hematopoietic* cells. The full biological background of this problem is out of the scope of this short paper. In the machine learning perspective, the problem can be formalized as classification or knowledge extraction. The former seeks to classify the tissue samples into two distinct classes/cell types. The latter aims to identify and interpret emerging molecular patterns, i.e. gene sets whose expression and/or coregulation differentiate between the cell types. In this paper we confine ourselves to classification.

The usual way of classifying gene expression data falls in attribute-value learning as tissue samples can be characterized by an invariable probe/gene set. However, in this experiment we use samples from four different species. In particular, from human, macaque, mouse and rat. The instant reason is that any single organism does not provide representative samples in both of the classes. More importantly, learning and generalizing over genomic properties of different species is of fundamental importance in the study of biological and evolutionary principles [1]. In any case, the tissue expression vectors cannot be directly matched as they are measured by different arrays using diverse probe (and thus gene/attribute) sets.

In this paper we propose alternative "working units" whose expression can be figured out and matched in different species – fully coupled fluxes. The various gene vector spaces are transformed into a uniform FCF space in which the classification is carried out. This approach not only allows to generalize beyond species, but also introduces a vocabulary of terms more robust than the original probes, respectively genes. The following subsections give a detailed description of the original data and their Prolog counterparts as well as construction of the abstract FCF attributes.

2.1 Gene Expression Data

We searched the Gene Expression Omnibus (GEO)[3], a public gene expression data library, for expression samples from various experiments involving different organisms. Irrespectively of the objectives of these different experiments, we selected those which included one of the mentioned tissue types among their measured samples. We were using only the measurements acquired by different Affymetrix DNA microarrays (chips).

We obtained 268 biological samples measured by 8 different arrays. 150 samples represent stromal cells while 118 samples correspond to hematopoietic cells. 163 samples were human, 11 of macaque, 8 of rat and 97 murine. Each GEO sample has a XML annotation that gives basic information about it. Among others, the annotation gives CellType which in our case can be hematopoietic or stromal. Further, several allied samples can be acquired within the same experiment, GDSno carries its identification. SampleID characterizes the sample, TissueState distinguishes normal and treated tissues, Organism determines the species and MArrayID gives the identification of the used chip. The annotations

[3] http://www.ncbi.nlm.nih.gov/geo/

were parsed and stored into the Prolog predicate *array*. The predicate as well as a particular fact are shown below:

array(CellType,GDSno,SampleID,TissueState,Organism,MArrayID,Comment).
```
array(hematopoietic,'GDS2718','GSM169465','normal','Mus musculus',
'GPL1261','GCOS 1.4 software (Affymetrix)').
```

Measurements were available as plain text files where rows contain microarray probe identifiers and corresponding expression values. They can easily be transformed into a list of Prolog facts, where SampleID characterizes the sample, AffyID identifies the probe and ExpressionValue gives the expression measured on the given probe in the given sample:

e(SampleID, AffyID, ExpressionValue).
```
e('GSM101111', 'AFFX-BioB-M_at', 1016.3).
```

2.2 Pathway Data

Kyoto Encyclopedia of Genes and Genomes (KEGG)[4] is a collection of manually drawn pathway maps representing common knowledge on the molecular interaction and reaction networks. KEGG stores pathways as XML files with a strictly defined structure. We transformed the four species specific KEGG XML files to Prolog facts, the transformations preserved as many graph features as possible (in fact, only visual representation and position of the elements were neglected). The predicate *entry* represents vertices, the predicate *relation* corresponds to edges. The argument Organism gives the species ('hsa' stands for homo sapiens). PathwayID identifies the reference pathway – a unique pathway of the given function shared by various organisms, KeggNodeID determines its vertex. ListOfEntrezIDs provides a list of genes that map on the given vertex within the specific organism. The genes are given by identifiers that are used by Entrez[5] – the integrated, search and retrieval system developed and maintained by National Center for Biotechnology Information (NCBI)[5]. NodeType specifies the type of vertex (e.g. gene product, group of gene products, compound or map). Interaction or relation is basically an oriented edge among nodes given by BeginNodeID and EndNodeID. A more detailed description can be found in KEGG Markup Language[4].

entry(Organism, PathwayID, KeggNodeID, ListOfEntrezIDs, NodeType).
```
entry('hsa', 04520,1, [hsa:4089], 'gene').
```

relation(Organism, PathwayID, BeginNodeID, EndNodeID, TypeOfRelation, SubTypeName, SubTypeValue).
```
relation('hsa', 04520, 14, 16, 'pprel', ['activation','phosphorylation'],
['-->','+p']).
```

[4] http://www.genome.jp/kegg/, http://www.genome.jp/kegg/docs/xml/
[5] http://www.ncbi.nlm.nih.gov/, http://www.ncbi.nlm.nih.gov/sites/gquery

2.3 Data Processing – Fully Coupled Fluxes

The above-mentioned representation enables to merge the species dedicated pathway data along the enzymes exhibiting the same behavior. In other words, the orthologous genes involved in the same vertex and having a similar function in the pathway can be mapped across all of the species under consideration. However, to improve robustness we use linear pathway subgraphs instead of vertices – FCF's. FCF is the longest possible chain of vertices in which non-zero vertex activation implies a certain (non-zero) activation in its successors. FCF's make the abstract attributes which substitute the original probes/genes.

The transformation proceeds in the following steps. First, the probes are assigned EntrezIDs, i.e. the probes are matched with genes. Prevailingly, there are multiple probes mapped to single EntrezID. The conversion predicate *affy2entrez* extracted from corresponding BioConductor libraries[6] maps AffyIDs and EntrezIDs introduced earlier:

affy2entrez(MArrayID,AffyID,EntrezID).
`affy2entrez('GPL1261','1452692_a_at',72900).`

Second, the activity of enzymes can be inferred from the predicates *affy2entrez*, *e* and *entry*. The activity is considered in terms of expression of the underlying genes, respectively probes. Obviously, this step involves one of the major difficulties. There is many-to-many relationship between array probes and pathway vertices, respectively enzymes. Moreover, the relationship varies across chips. It is also advisable to consider dependencies among contextual enzymes – the same enzyme can be produced by different genes in different contexts. That is why, this enumeration is delayed until FCF's are constructed. Third, the FCF's are found on basis of *relation* – the predicate *flux* is introduced:

flux(FCFID,PathwayID,ListOfKeggNodeIDs).
`flux(flux9,04520,[6,7,8,9,10,11,12,13,14,15,16,17,18,19]).`

Last, the activity of FCF's can be enumerated in every sample and used to classify them. This step is decomposed into two substeps. Initially, flux interpretations have to be found. These interpretations aim to find the most plausible definition of FCF's in all chips. For each flux, enzyme and chip, the corresponding interpretation picks the "optimal" probe. Correlation of expression values underlies the process of selection – the selected probes show the highest mean correlation in the given FCF within all samples measured by the given chip. The predicate *fi* gives the Prolog definition of flux interpretation in the given chip:

fi(MArrayID,FCFID,ListOfAffyIDs,CorrelationOfAffyIDs)
`fi('GPL1261',flux0,['1451002_at','1450048_a_at'],0.86).`

Having the flux interpretations, it is straightforward to calculate FCF activity in every sample. It is given by the mean expression of the probes taken from the appropriate interpretation. The overview of entities, predicates and their relations is given in Figure 1. For the sake of lucidity, we do not explicitly

[6] http://bioconductor.org/packages/1.9/data/annotation/

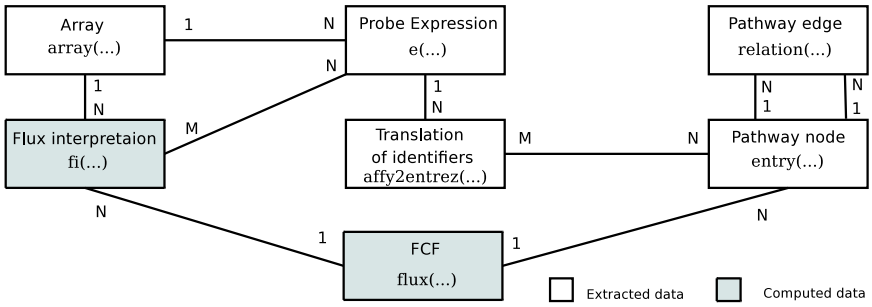


Fig. 1. Overview of entities, predicates and their relations.

mention other necessary and implemented features – missing value treatment or normalization of raw expression data.

3 Conclusions

The proposed ILP task introduces a new way to mine heterogeneous genomic data. It allows to generalize beyond genes as well as species. There are manifold direct implications to gene expression mining. Considering biological viewpoint, no targeted assay has yet been conducted to measure the expression profiles of the two types of tissues in a single experimental setup. The general methodology provides means to increase robustness and explanatory power of molecular classifiers. We have already obtained basic mining results enabled by the developed representation, although not yet with an ILP system. There were found FCF's with plausible biological interpretation that exhibit a statistically significant fold-change (ratio between the mean activity in both classes). Principal component analysis done in FCF's feature space confirmed that fluxes capture the difference between stromal and hematopoietic cells. The Prolog knowledge base is available on request.

Acknowledgements

This work was supported by the grant 1ET101210513 "Relational Machine Learning for Analysis of Biomedical Data" funded by the Czech Academy of Sciences and by the Czech Technical University grant CTU0814413. The Czech-USA travels were covered by Czech Ministry of Education through the project ME910.

References

1. S. Bergmann, J. Ihmels, and N. Barkai. Similarities and Differences in Genome-Wide Expression Data of Six Organisms. *PLoS Comput Biol*, 2(1):e9, 2003.

2. W. Dubitzky, M. Granzow, and Berrar D.P. *Fundamentals of Data Mining in Genomics and Proteomics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
3. T. Golub, D. Slonim, P. Tamayo, and et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.
4. R.A. Notebaart, B. Teusink, R.J. Siezen, and B. Papp. Co-Regulation of Metabolic Genes Is Better Explained by Flux Than Network Distance. *PLoS Comput Biol*, 4(1):e26, 2008.
5. D. Page and M. Craven. Biological applications of multi-relational data mining. *SIGKDD Explor. Newsl.*, 5(1):69–79, 2003.
6. A. Tamaddoni-Nezhad, R. Chaleil, A. Kakas, and S. Muggleton. Abduction and induction for learning models of inhibition in metabolic networks. In *ICMLA '05: Proc of the Fourth Int Conf on Machine Learning and Applications*, pages 233–239, Washington, DC, USA, 2005.
7. I. Trajkovski, F. Železný, N. Lavrač, and J. Tolar. Learning relational descriptions of differentially expressed gene groups. *IEEE Trans. Sys Man Cyb C*, 38(1):16–25, 2008.

Combining answer caching with smartcall optimization in mining frequent DL-safe queries

Joanna Józefowska[1], Agnieszka Ławrynowicz[1], and Tomasz Łukaszewski[1]

Institute of Computing Science, Poznan University of Technology,
ul. Piotrowo 2, 60-965 Poznan, Poland
{jjozefowska, alawrynowicz, tlukaszewski}@cs.put.poznan.pl

Abstract. Several query transformations have been proposed to speed up the execution of data mining algorithms in ILP systems. Many of them exploit the property that clauses are refined in a systematic way, which allows to save computation already done in the previous step. In this paper we empirically investigate query transformation technique that combines answer caching with smartcall optimization. The proposed technique is implemented in the setting of frequent pattern discovery in a hybrid knowledge base, combining description logics with disjunctive rules, where patterns have the form of conjunctive, DL-safe queries.

1 Introduction

Many approaches to inductive logic programming perform the search on hypothesis space, starting with a general hypothesis and systematically refining it. Hence, more specific hypothesis are very similar to their "parents" and what follows, the computations involved in testing the validity of the consecutive hypotheses may be very similar. In this context, we propose and emipirically investigate query transformation technique, based on the idea of caching and reusing intermediate results of previous computations. We implemented this technique in the frequent pattern mining setting, where patterns are represented as atom sets in the form of conjunctive, $\mathcal{DL}$-safe queries over combined knowledge base represented in description logics (DL) with disjunctive, positive rules.

2 Problem setting

Representation of data and patterns We assume mining patterns in a combined knowledge base (KB, P), where KB is description logics component and P is a program containing a set of (disjunctive) rules. Description logics knowledge base KB is divided, as usual, into *intensional* part (*terminological* one, a $TBox$) that contains axioms describing general domain knowledge and *extensional* part (*assertional* one, an $ABox$) that contains instance assertions. We assume that all the rules in P are $\mathcal{DL}$-safe [3]. Hence, the formalism used in our approach is that of $\mathcal{DL}$-safe rules, introduced in [3], however with some restrictions imposed

which are as described next. The subset of description logics assumed in our approach is $\mathcal{SHIF}$. In [3], there is the subset $\mathcal{SHOIN}(\mathbf{D})$ assumed. $\mathcal{SHOIN}(\mathbf{D})$ corresponds to OWL DL variant of OWL[1], a standard ontology language for the Web, while $\mathcal{SHIF}(\mathbf{D})$ to lighter version named OWL-Lite. $\mathcal{SHIF}$ is $\mathcal{SHIF}(\mathbf{D})$ without datatypes.

Example 1 (Example knowledge base (KB, P)). Given is a knowledge base (KB, P) describing bank services, presented below. $familyAccount$ is the only non-DL-predicate in (KB, P).

Terminology in KB	
$Client \equiv \exists\ isOwnerOf$	A client is defined as an owner of something.
$\top \sqsubseteq \forall\ isOwnerOf.(Account \sqcup CreditCard)$	The range of *isOwnerOf* is a disjunction of *Account* and *CreditCard*.
$\top \sqsubseteq \forall\ isOwnerOf^{-}.Person$	The domain of *isOwnerOf* is *Person*.
$relative \equiv relative^{-}$	The role *relative* is symmetric.
$\top \sqsubseteq \forall\ relative.Person$	The range of *relative* is *Person*.
$Account \sqsubseteq \exists\ isOwnerOf^{-}$	All accounts have an owner.
$\top \sqsubseteq \forall\ hasMortgage.Mortgage$	The range of *hasMortgage* is *Mortgage*.
$\top \sqsubseteq \forall\ hasMortgage^{-}.Account$	The domain of *hasMortgage* is *Account*.
$\top \sqsubseteq \leq\ 1\ hasMortgage^{-}$	A mortgage can be associated up to one account.
$Account \equiv \neg\ Person$	*Account* is disjoint with *Person*.
$Account \equiv \neg\ CreditCard$	*Account* is disjoint with *CreditCard*.
$Account \equiv \neg\ Mortgage$	*Account* is disjoint with *Mortgage*.
$Person \equiv \neg\ CreditCard$	*Person* is disjoint with *CreditCard*.
$Person \equiv \neg\ Mortgage$	*Person* is disjoint with *Mortgage*.
$Mortgage \equiv \neg\ CreditCard$	*Mortgage* is disjoint with *CreditCard*.
Assertions in KB	
Person(Anna).	Anna is a person.
isOwnerOf(Anna,a1).	Anna is an owner of a1.
hasMortgage(a1,m1).	M1 is associated to a1.
relative(Anna,Marek).	Anna is a relative of Marek.
Person(Jan).	Jan is a person.
isOwnerOf(Jan,cc1).	Jan is an owner of cc1.
CreditCard(cc1).	Cc1 is a credit card.
Person(Marek).	Marek is a person.
isOwnerOf(Marek,a1).	Marek is an owner of a1.
Rules in P	
familyAccount(x)← Account(x), isOwner(y,x), isOwner(z,x), relative(y,z)	*familyAccount* is an account that is co-owned by at least two relatives.

The patterns being found in our approach have the form of *conjunctive queries* over combined knowledge base (KB, P). The *answer set* of the query contains individuals of a reference concept $\hat{C}$. We assume that the queries are *positive*, that is do not contain negative literals. Moreover, we assume that the queries are $\mathcal{DL}$*-safe*, that is all variables in such a query are bound to individuals explicitly occurring in the knowledge base, even if they are not returned as part of the query answer.

[1] http://www.w3.org/TR/owl-features/

Definition 1 (Pattern). *Given is a combined knowledge base (KB, P). A pattern Q is a conjunctive, positive $\mathcal{DL}$-safe query over (KB, P) of the following form:*

$$Q(key) =? - \hat{C}(key), B_1, ..., B_n$$

B_1,...,B_n represent atoms of the query (where predicates are either atomic concepts, simple roles or non-DL-predicates). $q(key)$ denotes that variable key is the only one variable whose bindings are returned in the answer (distinguished one). x_1,...,x_m represent the variables of the query which are not a part of the answer (existential ones). A trivial pattern is the query of the form: $Q(key) =? - \hat{C}(key)$.

We assume that the queries posses *linkedness* property and are *range-restricted* both with regards to variable *key*. Note, that conjunctive queries, with regards to the work presented in [1], have been extended here from the ones over KB to the ones over (KB, P). What follows, the conjunctive queries, as presented in this paper, can contain intensional predicates from Disjunctive Datalog program P, (like $familyAccount$ defined in Example 1 and employed in Example 2). It means also that the patterns can contain n-ary predicates.

Example 2 (Example patterns). Consider the knowledge base (KB, P) from Example 1. Assuming that $Client$ is the reference concept $\hat{C}$, the following patterns, queries over (KB, P), may be built:
$Q_{ref}(key) =? - Client(key)$
$Q_1(key) =? - Client(key), isOwnerOf(key, x)$
$Q_2(key) =? - Client(key), isOwnerOf(key, x), familyAccount(x)$
where Q_{ref} is a trivial query, *reference query*, that counts the number of instances of the reference concept.

Frequent pattern mining Our formulation of frequent pattern mining is closest to the one defined in *SPADA* [2]. First we are going to define the support of the pattern.

Definition 2 (Support). *A* support *of query Q with respect to the knowledge base (KB, P) is defined as the ratio between the number of instances of reference concept $\hat{C}$ that satisfy query Q and the total number of instances of reference concept $\hat{C}$.*

Consider the queries from Example 2. The reference query has 3 items in its answer set that is 3 individuals from (KB, P) that are deduced to be $Client$. Query Q_2, for example, has 2 items in its answer set that is the clients that are co-owners of at least one account with their relatives ($Anna$, $Marek$). The support of query Q_2 is then calculated as: $support(\hat{C}, Q, KB) = \frac{2}{3} \approx 0.66$

Definition 3 (Frequent pattern discovery). *Given a knowledge base (KB, P), a set of patterns in a language $\mathcal{L}$ in a form of queries Q that all contain a reference concept $\hat{C}$ as a predicate in one of the atoms in the body, a minimum support threshold* minsup *specified by the user and assuming that queries with support s are frequent in (KB, P) given $\hat{C}$ if $s \geq$minsup, the task of frequent pattern discovery is to find the set $\mathcal{F}$ of frequent queries.*

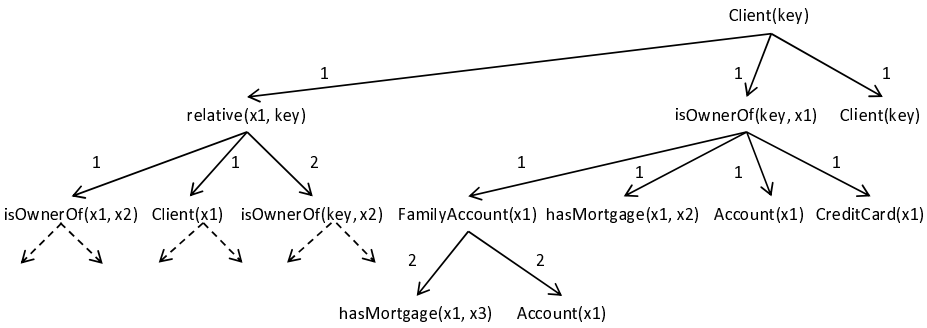


Fig. 1. A part of the trie constructed for the knowledge base from Example 1, *Client* as a reference concept and $minsup$=0.2.

Similarly to our work presented in [1], a *trie* data structure, introduced in [4], is used to systematically generate patterns. The trie data structure is a tree where each path from the root to a leaf contains candidate queries. Nodes in a trie correspond to the atoms of the query. An example of a trie is given in Figure 1. The labels with numbering on archs in Figure 1 correspond to two ways in which atoms can be added to the end of the query: (1) as *dependent atoms*, that is atoms that could not have been added at an earlier position, as they use at least one new variable of the last atom of the query, and (2) as copies of *right brothers* of a given atom in a trie. The trie is searched depth first.

3 Query transformation based on smartcall optimization and answer caching

Our proposed method introduces special predicates to an assertional part of a knowledge base to store the answers of previous, frequent queries. Each frequent query $Q1$ is transformed to a query $Q2$ with all of the relevant variables of $Q1$ as distinguished ones. "Relevant" means that not all the variable bindings have to be materialized at each step, only those that may have been affected by adding new literal to a query. Relevancy is computed based on the idea of so-called *smartcall* transformation [5]. In smartcall transformation, a query Q is partitioned according to the equivalence relation which is the transitive closure of the "shares undistinguished variables with" relation. Then, while testing refinements Q, R_i of frequent query Q, parts of Q for which the success is independent of the success of R_i (that is parts of Q that are not connected to R_i via undistinguished variable chains) need not be tested again. We use this idea to identify the subsets of variables that have to be materialized, due to their connection with the variables in an atom newly introduced to a query. Last depth at which given variable bindings were materialized is stored in a hash list.

Each tuple in the result of $Q2$ is materialized in the following way. For each variable v_i in $Q2$, except the *key* one, an assertion is created, of the form: $\$sup\_d\_v_i(a, b)$, where d denotes depth of a query in a trie, a-value of

the *key* variable, *b*-value of the given variable v_i in a tuple. For the *key* variable the assertion has the form: $\$key\_d(a)$ (see: Algorithm 2). A trie is generated recursively depth first. The assertions, introduced by some node, stay in a knowledge base until all of it's children are recursively expanded. The assertional part of knowledge base from Example 1, after materializing the results of frequent patterns from the trie in Figure 1, up to the pattern $Q(key) = ?-Client(key), isOwnerOf(key, x1), familyAccount(x1)$, looks as follows:

Updated assertions in KB			
Person(Anna). *isOwnerOf(Anna,a1).* *hasMortgage(a1,m1).* *relative(Anna,Marek).*	*$key_1(Anna).*	*$key_2(Anna).* *$sup_2_x1(Anna,a1).*	*$key_3(Anna).* *$sup_3_x1(Anna,a1).*
Person(Jan). *isOwnerOf(Jan,cc1).* *CreditCard(cc1).*	*$key_1(Jan).*	*$key_2(Jan).* *$sup_2_x1(Jan,cc1).*	
Person(Marek). *isOwnerOf(Marek,a1).*	*$key_1(Marek).*	*$key_2(Marek).* *$sup_2_x1(Marek,a1).*	*$key_3(Marek).* *$sup_3_x1(Marek,a1).*

Then, evaluating queries with regards to the new assertional part of a knowledge base is processed like is presented in Algorithm 1.

Algorithm 1 ***evaluateCandidate(*** $Q1(\mathbf{x}, \mathbf{y})$***, d, lastMaterializationDepthsList)***

1. ***foreach*** *variable z_i, except key one, appearing in both: $last(Q1)$ (last atom of $Q1$) and in the preceding atoms of $Q1$* ***do***
2. *retrieve last depth $d(z_i)$, where bindings of z_i were materialized, from the hash list lastMaterializationDepthsList*
3. *construct query $Q2(\mathbf{x}, \mathbf{z})$, of the form: $Q(x) = ?-[\$key\_(d-1)(x)], [\$sup\_(d(z_1))\_z_1(x, z_1), ..., \$sup\_(d(z_k))\_z_k(x, z_k)], last(Q1)$*
4. *evaluate query $Q2$*

The example, candidate queries: $Q_1(key) = ?-Client(key), isOwnerOf(key, x1), familyAccount(x1), hasMortgage(x1, x3)$ and $Q_2(key) = ? - Client(key), isOwnerOf(key, x1), familyAccount(x1), Account(x1)$ look as follows after transformation: $Q'_1(key) = ?-\$sup\_3\_x1(key, x1), hasMortgage(x1, x3)$ and $Q'_2(key) = ? - \$sup\_3\_x1(key, x1), Account(x1)$ respectively. For creating subsequent materialized results previous materialized results are used as in Algorithm 2.

Algorithm 2 ***materializeResults(*** $Q1(\mathbf{x}, \mathbf{y})$***, d, lastMaterializationDepthsList)***

1. *determine variables z_i connected to $last(Q1)$ via body variable chain*
2. ***foreach*** *variable z_i* ***do***
3. *retrieve last depth $d(z_i)$, where bindings of z_i were materialized, from the hash list lastMaterializationDepthsList*
4. *construct query $Q2(\mathbf{x} \cup \mathbf{z}, \emptyset)$ of the form: $Q(x, z_1, ..., z_k) = ? - \$key\_(d-1)(x), \$sup\_(d(z_1))\_z_1(x, z_1), ..., \$sup\_(d(z_k))\_z_k(x, z_k), last(Q1)$*
5. *evaluate query $Q2(\mathbf{x} \cup \mathbf{z}, \emptyset)$*
6. *add relation $\$key\_d(x)$, a set of key variable bindings in the query result, to KB*
7. ***foreach*** *z_i*
8. *compute relation $\$sup\_d\_i(x, z_i)$, a set of 2-tuples selected from the query result, and add it to KB*

For example, the pattern $Q(key) =? - Client(key), isOwnerOf(key, x1), familyAccount(x1)$ that added the assertions *$key_3(Anna)*, *$sup_3_x1(Anna,a1)* *$key_3(Marek)* and *$sup_3_x1(Marek,a1)* to KB, would be transformed to the query $Q'(key, x1) =? - \$key\_2(key), \$sup\_2\_x1(key, x1), familyAccount(x1)$.

4 Empirical evaluation

Our implementation is written in Java. As the reasoner on a combined knowledge base (KB, P) we use KAON2[2]. In Figure 4 some of our experimental results on three benchmarks are presented[3].

LUBM, minsup=0.3, reference Concept=Person

Depth	C	P	C	P	Time(s)		Speedup(%)
	Up to depth		At depth		OA	MR	MR
1	1	1	1	1	1,0	1,0	100,0%
2	66	6	66	6	14,9	**8,7**	**171,1%**
3	265	31	199	25	133,8	**104,7**	**127,8%**
4	1416	193	1151	162	3308,0	**3104,0**	**106,6%**

SWRC-AFB, minsup=0.2, reference Concept=Person

Depth	C	P	C	P	Time(s)		Speedup(%)
	Up to depth		At depth		OA	MR	MR
1	1	1	1	1	1,0	1,0	100,0%
2	88	3	88	3	19,5	**11,6**	**167,4%**
3	259	14	171	11	146,5	**115,8**	**126,5%**
4	873	98	614	84	2291,3	**1642,7**	**139,5%**

FINANCIAL-GOLD, minsup=0.2, reference Concept=Client

Depth	C	P	C	P	Time(s)		Speedup(%)
	Up to depth		At depth		OA	MR	MR
1	1	1	1	1	1,0	1,0	100,0%
2	8	6	8	6	3,3	3,9	84,8%
3	78	24	70	18	54,7	56,9	96,2%
4	182	63	104	39	238,1	**198,4**	**120,0%**
5	448	125	266	62	1023,5	**701,3**	**145,9%**
6	612	182	164	57	2201,4	**1385,0**	**158,9%**
7	801	208	189	26	3123,5	**1987,5**	**157,2%**
8	853	215	52	7	2426,3	**2220,1**	**109,3%**
9	872	216	19	1	3582,2	**2254,4**	**158,9%**

C-candidates, P-patterns

Fig. 2. Experimental results. OA - original algorithm, MR - materialized results used.

A trie was generated up to the specified maximum depth values. The runtimes are the times of a whole trie generation for each maximum depth (maximum length of patterns). Using materialized results of previous queries (MR) allowed to achieve speedups for all knowledge bases.

References

1. Józefowska J., Ławrynowicz A., Łukaszewski T. (2006) Frequent pattern discovery in OWL DLP knowledge bases, In Proc. of EKAW 2006, 287-302
2. Lisi F.A., Malerba D. (2004) Inducing Multi-Level Association Rules from Multiple Relation, Machine Learning Journal, 55, 175-210
3. Motik B., Sattler U., Studer R. (2004) Query Answering for OWL-DL with Rules. In Proc. of ISWC 2004, 549-563
4. Nijssen S., Kok J.N. (2001) Faster Association Rules for Multiple Relations. In Proc. of the IJCAI'01, 891-897
5. Santos Costa V., Srinivasan A., Camacho R., Blockeel H., Demoen B., Janssens G., Struyf J., Vandecasteele H. and Van Laer W. (2002) Query transformations for improving the efficiency of ilp systems. Journal of Machine Learning Research 4:465-491

[2] http://kaon2.semanticweb.org/

[3] For description of datasets see: http://www.ecmlpkdd2007.org/CD/workshops/PRICKLWM2/P_Joz/p7Final.pdf

Probabilistic Local Pattern Mining

Angelika Kimmig and Luc De Raedt

Dept. of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, POBox 2402, BE-3001 Heverlee, Belgium {firstname.lastname}@cs.kuleuven.be

Abstract. We study local pattern mining in the context of *probabilistic* relational databases, developing algorithms for probabilistic variants of both frequent and correlated pattern mining that use principles of statistical relational learning. As probabilistic selection criteria, we introduce both a likelihood function and a probabilistic frequency. We report on experiments on a challenging biological network mining task.

1 Introduction

Local pattern mining traditionally aims at identifying patterns that satisfy certain constraints w.r.t. a given database [8]. Numerous works have been devoted to local pattern mining since the introduction of item-set mining, cf. [1]. Prominent approaches search for all such patterns (frequent pattern mining [1]) or the best k patterns (correlated pattern mining [9]). Transferring local pattern mining techniques to *probabilistic databases* allows to account for uncertainty, which often arises naturally, e.g. in scientific data or robotics. This requires changing the membership relation determining when a query matches a tuple from deterministic to probabilistic. This work investigates the mining of local relational patterns in probabilistic databases. It is motivated by and empirically evaluated in a large biological database containing information about known or predicted relationships for various types of objects [10, 7, 3]. We employ ProbLog [3] to represent the database, but the techniques and principles are directly transferable to other probabilistic formalisms. As ProbLog is a direct extension of Prolog, our work builds upon existing work in multi-relational data mining and inductive logic programming, where local pattern mining is known under the names of *query mining* [5] and *query flocks* [11]. We adapt both correlated and frequent pattern mining, introducing two types of probabilistic membership functions: a kind of likelihood criterion as well as a probabilistic frequency measure.

2 Query Mining

Query mining upgrades traditional local pattern mining to the representations of multi-relational databases [5]. We use Datalog to represent databases and queries, abbreviating vectors of variables as $\boldsymbol{X}$. We assume a designated table ID containing the set of tuples or examples to be characterized using queries,

and restrict the language $\mathcal{L}$ of patterns to the set of conjunctive queries of the form

$$r(\boldsymbol{X}) : -ID(\boldsymbol{X}), l_1, ..., l_n \tag{1}$$

where the l_i are positive atoms or conjunctive conditions. Additional syntactic or semantic restrictions, called *bias*, can be imposed on the form of conjunctive queries by explicitly specifying the language $\mathcal{L}$, cf. [11, 4, 5].

Frequent Query Mining aims at finding all queries satisfying a selection predicate ϕ. It can be formulated as follows, cf. [5, 4]:

Given a language $\mathcal{L}$ containing queries of the form (1), a database $\mathcal{D}$ including the designated relation ID, and an anti-monotonic selection predicate ϕ
Find all queries $q \in \mathcal{L}$ such that $\phi(q, \mathcal{D}) = true$.

A prominent example of an anti-monotonic selection predicate is minimum frequency, requiring a minimum number of tuples covered. Anti-monotonicity is based on a generality relation between patterns. We employ OI-subsumption [6], as the corresponding notion of subgraph isomorphism is favorable within the intended application in network mining.

Correlated Pattern Mining [9] uses both *positive* and *negative* examples, given as two designated relations ID^+ and ID^- of the same arity, to find the top k patterns, that is, the k patterns scoring best w.r.t. a function ψ. The function ψ employed is convex, e.g. measuring a statistical significance criterion such as χ^2, cf. [9], and measures the degree to which the pattern is statistically significant or unexpected. The task of correlated pattern mining is defined as:

Given a language $\mathcal{L}$ containing queries of the form (1), a database $\mathcal{D}$ including the designated relations ID^+ and ID^-, and a correlation function ψ
Find $\arg_k \max_{q \in \mathcal{L}} \psi(q, \mathcal{D})$

Multi-relational query miners such as [5, 4] often follow a level-wise approach for frequent query mining [8], where at each level new candidate queries are generated from the frequent queries found on the previous level. In contrast to Apriori, instead of a "joining" operation, they employ a refinement operator ρ to compute more specific patterns, and also manage a set of infrequent queries to take into account the specific language requirements imposed by $\mathcal{L}$. To search for all solutions, it is essential that the refinement operator is optimal w.r.t. $\mathcal{L}$, i.e. ensures that there is exactly one path from the most general pattern to every pattern in the search space. This can be achieved by restricting the refinement operator to generate queries in a canonical form, cf. [4].

Morishita and Sese [9] adapt Apriori for finding the top k patterns w.r.t. a boundable function ψ, i.e. for the case where there exists a function u (different from a global maximum) such that $\forall g, s \in \mathcal{L} : g \preceq s \rightarrow \psi(s) \leq u(g)$. Again, at each level candidate queries are obtained from those queries generated at the previous level that qualify for refinement, but here, this requires that such a query either belongs to the current k best queries, or is still promising, that is, has an upper-bound higher than the value of the current k-th best query.

3 Probabilistic Query Mining

The framework for query mining as outlined above can directly be adapted towards the probabilistic case. While the overall structure of the algorithms remains the same, two key components change: the database $\mathcal{D}$ is *probabilistic*, and the selection predicate ϕ or the correlation measure ψ is based on the probabilities of queries. In principle, any formalism defining such probabilities could be used. We employ ProbLog [3], as it is a very simple yet powerful logic.

A *ProbLog program* T consists of a set of labeled facts $p_i :: c_i$ together with a set of definite clauses. Each ground instance of such a fact c_i is true with probability p_i, where all probabilities are assumed mutually independent. The program therefore naturally defines a probability distribution $P(L|T)$ over logic programs $L \subseteq L_T = \{c_1, \cdots, c_n\}$. It can be used to specify two types of query probabilities:

$$P_s(q|T) = \sum\nolimits_{L \subseteq L_T} P(q|L) \cdot P(L|T) \tag{2}$$

$$P_x(q|T) = \max_{e \in E(q)} P(e|T) = \max_{e \in E(q)} \prod_{c_i \in e} p_i \tag{3}$$

where $P(q|L) = 1$ if there exists a θ such that $L \models q\theta$, and $P(q|L) = 0$ otherwise, and $E(q)$ is the set of all explanations or proofs for query q [7]. The *success probability* $P_s(q|T)$ thus corresponds to the probability that q is *provable* in a randomly sampled logic program, the *explanation probability* $P_x(q|T)$ to that of sampling all clauses needed in the most likely proof. Evaluating P_s is computationally hard. In [3], an approximation algorithm is proposed that repeatedly computes an upper and a lower bound on P_s until their difference becomes sufficiently small. P_x can easily be calculated using a best-first search.

To define *probabilistic selection predicates*, we use either $P_s(q(t)|\mathcal{D})$ (2) or $P_x(q(t)|\mathcal{D})$ (3) as the probabilistic membership function $P(t|q, \mathcal{D})$. Note that P is anti-monotonic: if $q_1 \preceq q_2$ then $P(t|q_1, \mathcal{D}) \geq P(t|q_2, \mathcal{D})$. As in the case of the traditional frequency function, which can be seen as the sum of a deterministic membership relation over all tuples, a selection predicate can naturally be obtained by combining a minimum threshold with the aggregated probabilities of all tuples t in the relation ID, $agg_{t \in ID}(P(t|q, \mathcal{D})) > c$. As typical functions used in correlated pattern mining such as χ^2 rely on hard 0-1 decisions, they cannot easily be employed here. Instead, we modify the aggregation function to account for the class of each example, i.e. increase with increasing probability of positive examples, but decrease with increasing probability of negative examples. Note that this includes using unclassified instances, as in frequent pattern mining, as special case with $ID^+ = ID$ and $ID^- = \emptyset$. Choosing sum as aggregation function results in a *probabilistic frequency* pf (4) also employed by [2] in the context of item-set mining, whereas product defines a kind of *likelihood* LL (5). Notice that using the product in combination with a non-zero threshold implies that *all* positive examples must be covered with non-zero probability. We therefore introduce a softened version LL_n (6) of the likelihood, where $n < |ID^+|$

examples have to be covered with non-zero probability. This is achieved by restricting the set of tuples in the product to the n highest scoring tuples in ID^+, thus integrating a deterministic (anti-monotonic) selection predicate into the probabilistic one. More formally, the three functions used are defined as follows:

$$pf(q,\mathcal{D}) = \sum_{t \in ID^+} P(t|q,\mathcal{D}) - \sum_{t \in ID^-} P(t|q,\mathcal{D}) \tag{4}$$

$$LL(q,\mathcal{D}) = \prod_{t \in ID^+} P(t|q,\mathcal{D}) \cdot \prod_{t \in ID^-} (1 - P(t|q,\mathcal{D})) \tag{5}$$

$$LL_n(q,\mathcal{D}) = \prod_{t \in n(ID^+)} P(t|q,\mathcal{D}) \cdot \prod_{t \in ID^-} (1 - P(t|q,\mathcal{D})) \tag{6}$$

Here, $n(ID^+)$ contains the n highest scoring tuples in ID^+. Note that whenever the membership function P is monotone, combining one of these functions with a minimum threshold leeds to monotone selection predicates w.r.t. OI-subsumption. In correlated pattern mining, omitting the scores of negative examples provides an upper bound in all cases.

4 Implementation and Experiments

Our implementation of both frequent and correlated pattern mining in Yap-5.1.2 starts from the (non-probabilistic) frequent query mining system *c-armr* of [4], but employs a modified canonical form ordering literals primarily on their arguments. ProbLog is used to evaluate probabilistic membership. As in *c-armr*, the bias can be defined using type and mode restrictions as well as background knowledge. To deal with classified examples, we keep track of patterns that are infrequent, but cannot be pruned based on their upper bound. To increase pruning, we modified the search strategy for correlated pattern mining to use the score of the kth best pattern so far as threshold.

We perform experiments in the context of the weighted biological database of [10], concentrating on the 142 human genes in our database known to be related to Alzheimer disease according to Entrez. As the known practical limitations of frequent pattern mining are inherited in the probabilistic case, we focus on correlated pattern mining. We study the following questions:

Q1 How do P_s and P_x differ in performance?
Q2 Can the top queries discriminate unseen positive and negative examples?
Q3 Does the correlated pattern miner perform effective pruning?
Q4 Can the correlated pattern miner use the full network?

We used the full graph G0 of around 1M nodes and 6M edges, as well as two connected subgraphs G1 (658 nodes, 3544 edges) and G2 (3364 nodes, 17666 edges) around Alzheimer disease also used in [7]. Each graph is represented as a probabilistic table of typed edges and a deterministic table of labeled nodes. The refinement operator adds literals of the form `edge(X,Y,e)`, `edge(X,Y)` (abbreviating `edge(X,Y,_)`) and `node(X,n)` where `X` and `Y` are variables of type

	LL^s	LL^s_n	pf^s	LL^x	LL^x_n	pf^x
(a)	.95/.95/.94	.57/.28/.07	.66/.41/.16	1/1/.96	1/1/1	1/1/1
(b)	.13	.70	.77	.91	.79	.78

Table 1. Fraction of completed runs for $k = 1/5/20$ (a) and fraction of positives ranked before first negative, averaged over completed runs with at least 5 pairs of examples (b).

node name, `X` already appears in the pattern, and `e` and `n` are constants denoting labels. The bias further states that labels are mutually exclusive, that `edge(X,Y,e)` implies `edge(X,Y)`, and how to invert labels when using edges backwards. This ensures that edges in patterns map to database entries independent of direction. Training examples are gene nodes annotated (positive) resp. not annotated with AD (negative) randomly picked from G1. We use 100 datasets with 1 to 10 examples of each class. `q(X):-id(X),node(X,'Gene')` is used as most general query. We approximate P_s by the lower bound of the approximation algorithm with interval width $\delta = 0.1$ and a timelimit of 60 sec for the evaluation of each individual bound. We use the likelihood LL (5) and the probabilistic frequency pf (4). For $m \geq 5$ pairs of examples, we also consider the softened likelihood LL_n (6) with $n = \lceil m/2 \rceil$, indicating the probability used by superscripts where needed. Experiments are performed on a cluster, requesting at least 1GB of memory, with a timelimit of 23:20 hours per run.

To answer Q1 and Q2, we use G1 with $k = 1, 5, 20$. Table 1 illustrates the performance in terms of the fraction of successful runs. Unsuccessful runs are due to the timelimit for P_x and to exhausting memory for P_s. To compare P_s and P_x in terms of their results, we use the best pattern (omitting `id(X)`) to retrieve covered examples from G2, and rank those using the corresponding P, excluding training examples. In case of equally likely patterns, we choose the most specific one, and break remaining ties randomly. We found that all patterns return several positive examples first. Table 1 also shows that, except for the very general patterns obtained by LL^s, a large fraction of all positive examples is returned before the first negative one. Together, these results show that the best patterns are indeed able to distinguish the unseen positive from the unseen negative examples, thus answering Q2 positively. Combining resource requirements and results, the answer to Q1 is that using P_x is more favorable.

Mining on G2 with pf^x on average examines up to 200 patterns for $k = 1$ and at most 2500 for $k = 50$. Given more than 2M queries of length at most 4, this clearly answers Q3 affirmatively. Using pf^x on the full graph G0 with datasets of size ten, runs for $k = 1$ took 7 to 149 minutes on a 2.2GHz 4GB machine, with an average of 64 minutes. For $k = 5$, the timelimit was reached. Although probabilistic relational query mining is computationally challenging, large networks can thus in principle be used for small values of k. This answers Q4 affirmatively, but improving the efficiency of the ProbLog engine is clearly necessary and actually part of our current work.

5 Conclusions and Related Work

We extended local pattern mining towards a probabilistic relational framework by providing a frequent as well as a correlated pattern mining algorithm, with scoring functions aggregating probabilities. Results on challenging biological network mining tasks show that correlated pattern mining with pf^x is most effective in terms of both function and efficiency. Although a very large network can be used, improving the efficiency is needed and actually underway.

Our work builds upon existing multi-relational data mining systems such as c-armr [4]. As far as we are aware, the only existing approach to mining probabilistic data is that of [2], who study frequent item-set mining using expected support, corresponding to the probabilistic frequency of Equation (4), but neither consider relational data nor correlated pattern mining. Probabilistic explanation based learning (PEBL) [7] is a related approach in that it also results in a set of patterns. However, it is also significantly different: patterns are obtained by generalizing the logical structure of proofs of the examples w.r.t. a domain theory defining a target predicate. PEBL thus searches a highly constrained space of possible patterns, and hence, it is more efficient to use, but also more restricted.

Acknowledgments A. Kimmig is supported by the Research Foundation Flanders (FWO Vlaanderen). This work is supported by the GOA project 2008/08 Probabilistic Logic Learning; it uses HPC resources `http://ludit.kuleuven.be/hpc`.

References

1. R. Agrawal *et al.* Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, p. 307–328. The MIT Press, 1996.
2. C. K. Chui *et al.* Mining frequent itemsets from uncertain data. In *PAKDD*, vol. 4426 of *LNCS*, p. 47–58. Springer, 2007.
3. L. De Raedt *et al.* ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, p. 2462–2467, 2007.
4. L. De Raedt and J. Ramon. Condensed representations for inductive logic programming. In *KR*, AAAI Press, p. 438–446, 2004.
5. L. Dehaspe *et al.* Finding frequent substructures in chemical compounds. In *KDD*, p. 30–36. AAAI Press, 1998.
6. F. Esposito *et al.* Ideal refinement under object identity. In *ICML*, p. 263–270. Morgan Kaufmann, August 2000.
7. A. Kimmig *et al.* Probabilistic explanation based learning. In *ECML*, vol. 4701 of *LNCS*, p. 176–187. Springer, 2007.
8. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
9. S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *PODS*, p. 226–236. ACM Press, 2000.
10. P. Sevon *et al.* Link discovery in graphs derived from biological databases. In *DILS*, vol. 4075 of *LNCS*, p. 35–49. Springer, 2006.
11. S. Tsur *et al.* Query flocks: A generalization of association-rule mining. In *SIGMOD Conference*, p. 1–12, 1998.

HiFi: Tractable Propositionalization through Hierarchical Feature Construction

Ondřej Kuželka and Filip Železný

Intelligent Data Analysis Research Group
Dept. of Cybernetics, Czech Technical University in Prague
http://ida.felk.cvut.cz
{kuzelo1,zelezny}@fel.cvut.cz

Abstract. We present a novel propositionalization algorithm HiFi based on constructing first-order features with hierarchical structure. Unlike state-of-the-art algorithms, HiFi simultaneously constructs features and computes their extensions, while this merged operation takes time polynomial in the maximum feature length and in the total number of features. Moreover, all features produced by HiFi are the smallest in their semantic equivalence class. In our preliminary experiments we show runtime improvements with respect to a state-of-the-art propositionalization algorithm.

1 Introduction

This paper addresses the problem of propositionalization, i.e. converting structured descriptions of learning examples into attribute-value descriptions. A major stream of state-of-the-art approaches to propositionalization [3, 2, 5] is based on constructing *features* in the form of queries. A set of generated features then plays the role of the attribute set for the new representation. The range of all possible features is, for a given learning problem, usually constrained by declaring a language of admissible features. However, this popular form of propositionalization suffers from two sources of complexity: i) finding features complying to the specified language constraints and ii) computing extension of the generated features, i.e. determining for which examples they are true. These sub-problems usually introduce two exponential (in n) time complexity factors into propositionalization (RSD [5] or SINUS [2]). This paper mainly shows how both of these intractability sources can be simultaneously removed (reduced to polynomial-time) if one only works with *hierarchical features*.

In [7] it was shown that certain constraints imposed on the feature language enable to reduce the feature construction problem to the problem of satisfying a (polynomially large) set of propositional Horn clauses, which is a tractable problem. One of such sufficient constraints is the hierarchical structure of features. In this work we additionally remove the second source of complexity (finding extensions) for hierarchical features. For this we exploit the fact that verifying subsumption between a feature and an example can be reduced to a constraint

satisfaction problem, which in case of a hierarchical feature can be efficiently solved using a method known as directed arc consistency checking.

It would be thus theoretically straightforward to implement a propositionalization algorithm by two subsequent problem reductions (HornSAT and CSP). This would be impractical namely due to the overhead incurred by representation conversions and by the implied separation of feature construction from extension computation. HiFi works in the polynomial bound but brings further benefits by avoiding explicit reductions and by merging the mentioned two stages of propositionalization. A further advantage of HiFi is that it only produces *reduced* features; i.e. of all semantically equivalent features it only produces the smallest. We omit proofs, which will appear in an extended account of this work

2 Preliminaries

Most algorithms in inductive logic programming rely on a generality relation between clauses. While it would be natural to say that C is more general than D whenever C entails D (written $C \models D$), the entailment relation is undecidable and thus is typically approximated by θ-subsumption.

Definition 1 (Subsumption, equivalence, reduction). *Let C and D be clauses and let there be a substitution θ such that $lits(C\theta) \subseteq lits(D)$. We say that C θ-subsumes D (written $C \preceq_\theta D$). If further $D \preceq_\theta C$, we call C and D θ-equivalent (written $C \approx_\theta D$). We say that C is θ-reducible if there exists a clause C' such that $C \approx_\theta C'$ and $|C| > |C'|$. A clause C' is said to be a θ-reduction of C if $C \approx_\theta C'$ and C' is not θ-reducible.*

It is a NP-complete problem to decide θ-subsumption between two clauses and co-NP-complete to decide θ-reduction [4].

Following up on established approaches to propositionalization, we further constrain the language bias for features. We do this through the notion of a feature template.

Definition 2 (Template). *A* template *τ is a pair (γ, μ) where γ is a ground function-free query and μ is a subset of all arguments in τ. Arguments of τ contained in μ are called* input arguments*, the other arguments are called* output arguments*. Let clause $C \preceq_\theta \gamma$. An occurence of variable v at the i-th argument of literal l in C is called an* input occurrence *(w.r.t. τ and θ) if the i-th argument of literal $\lambda_{C,\gamma,\theta}(l)$ is an input argument, otherwise it is an* output occurrence *(w.r.t. τ and θ). A literal in C containing only output variables is called a* root *of C (w.r.t. τ and θ).*

Definition 3 (Feature). *Let $\tau = (\gamma, \mu)$ be a template and $n \in N$. A query f containing no constants or functions is a* feature *correct w.r.t τ and n if $|f| \leq n$, $f \preceq_\theta \gamma$ and for every input (output, respectively) occurrence of a variable in f there is also an output (input) occurrence of the same variable in f, where both occurrences are taken w.r.t τ and θ.*

Note that a θ-reduction of a correct feature may not be a correct feature itself. This may represent a problem because, to avoid redundancy, we would like to work with reduced features. In the next section we will show how to reduce *hierarchical* features while maintaining correctness.

3 Hierarchical Features

In this section, we define hierarchical features and list some of their properties regarding θ-reduction and CSP representations.

Definition 4 (Hierarchical feature). *A template $\tau = (\gamma, \mu)$ is* hierarchical *if every literal in γ has at most one input argument and there is a partical irreflexive order $\prec$ on constants in γ such that $c \prec c'$ whenever c (c') occurs at an input (output) argument of γ. A feature correct w.r.t. a hierarchical template is a* hierarchical feature *if it has exactly one root.*

Graphically, a hierarchical feature f corresponds to a unique tree graph T_f with vertices v_i corresponding to literals l_i. There is an edge between v_i and v_j iff a variable has an output occurrence in l_i and an input occurrence in l_j. Consider a subtree S_f of T_f. A query q consisting of literals corresponding to vertices in S_f is called a *subfeature* of f, and q's literal corresponding to the root of S_f is its *subroot* (w.r.t. f).

The next definition introduces $H\theta$-subsumption and $H\theta$-reduction. Informally, $H\theta$-subsumption requires that θ maps literals at a given depth in one clause to literals at the same depth in the other clause.

Definition 5 ($H\theta$-reduction). *We say that hierarchical feature f $H\theta$-subsumes hierarchical feature g (written $f \preceq_{H\theta} g$) iff there is a substitution θ such that $f\theta \subseteq g$ and for every literal $l \in lits(f)$ there is a literal $l\theta \in lits(g)$ such that $depth_f(l) = depth_g(l)$. If further $g \preceq_{H\theta} f$, we call f and g $H\theta$-equivalent (written $f \approx_{H\theta} g$). We say that f is $H\theta$-reducible if there is a hierarchical feature f' such that $f \approx_{H\theta} f'$ and $|f| > |f'|$. Hierarchical feature f' is said to be a $H\theta$-reduction of f if $C \approx_{H\theta} f'$ and f is not $H\theta$-reducible.*

Clearly, $H\theta$-reducibility implies θ-reducibility, but not vice versa. The next lemma provides the basic means to detect $H\theta$-reducibility of a hierarchical feature.

Lemma 1. *A hierarchical feature f is $H\theta$-reducible if and only if it has subfeatures f_1, f_2 whose respective subroots share the same input variable, and $f_1 \preceq_{H\theta} f_2$.*

As a consequence of the lemma above, it is possible to decide reducibility of a hierarchical feature f in time polynomial in the size of f.

Lemma 2. *A constraint graph induced by a subsumption problem $f \preceq_\theta e$ where f is a hierarchical feature, has constraints only over those pairs of variables corresponding to literal pairs where the shared FOL variable is used as an output in one of them and as an input in the other one.*

Algorithm 1 *computeDomain*$(root, example)$

Input: Root of the constraint graph $root$, Example e;

$rootDomain \leftarrow$ { all literals built on the same predicate symbol }
for $\forall child \in children(root)$ **do**
    $childDomain \leftarrow computeDomain(child)$
    remove all values from $rootDomain$ for which there are no values in $childDomain$ such that the corresponding constraint would be satisfied
end for
return $rootDomain$

Lemma 2 allows us to reduce the problem of deciding θ-subsumption for hierarchical features to problem of solving a tree-structured constraint satisfaction problem. It is known that such CSP problems are efficiently soluble [1]. Here we follow an algorithm based on directed-arc-consistency. The basic idea of the directed-arc-consistency method is that when we filter all domains of CSP-variables in such a way that domains of these variables contain only values consistent with filtered domains of their children, then it is possible to find a solution by assigning values from the filtered domains to CSP-variables proceeding from root to leaves (Algorithm 1).

4 The Propositionalization Algorithm

In this section, we design a propositionalization algorithm HiFi, which runs in time polynomial in the maximum feature size and in the number of generated features. HiFi merges the two usual phases of propositionalization, i.e. feature construction and extension computation. Specifically, the core algorithm accepts a learning example and a feature template. It produces all features complying to the template and subsuming the example. These features are obtained by combinatorial composion of subfeatures, which act as the primitive building blocks. One of the advantages of this assembly approach is that subsumption of the given example can already be checked for individual subfeatures; if it is refuted for a given subfeature, this subfeature is not used in the subsequent feature assembly.

The algorithm exploits the partial irreflexive order, which is imposed on types of arguments by Def. 4. Due to existence of this order it is possible to sort all declared predicates $l \in \gamma$ topologically with respect to a graph induced by the partial order. When the topological ordering is found, it is possible to organize generation of features in such a way that subfeatures are built iteratively by combining smaller subfeatures into larger ones. We illustrate this through an example.

Example 1. Consider the following template

$$\tau = car(-c) \wedge load(+c, -l) \wedge triangle(+l) \wedge box(+l).$$

The toplogical order of literals $l \in \tau$ then corresponds to $(box(+l), triangle(+l), load(+c, -l), car(-c))$. Now, we take the first declared literal $box(+l)$ and build

the set of all possible subfeatures with $box(+l)$ as its subroot $S_{box(+l)} = \{box(L)\}$. Similarly, for $triangle(+l)$ we have $S_{triangle(+l)} = \{triangle(L)\}$. The third declared predicate $load(+c, -l)$ has outputs. Thus, subfeatures with this predicate in the subroots are obtained by all possible graftings of the already generated subfeatures onto the $load/2$ subroot. This results in

$$S_{load(+c,-l)} = \{load(C, L) \wedge triangle(L), load(C, L) \wedge box(L), \ldots$$

$$\ldots load(C, L) \wedge box(L) \wedge triangle(L)\}.$$

The set $S_{car(-c)}$ is then created similarly as $S_{load(+c,-l)}$. Note that so far we have not considered that a maximum allowed size of a feature is specified.

Generating features in this manner can be conveniently combined with computation of θ-subsumption. Brief inspection of Algorithm 1 reveals that in order to compute domain of a literal, which is a subroot of some subfeature, we only need to know the domains of its children. However, the domain of any literal l can be computed when l is added as a subroot of some subfeature to the set of already generated subfeatures and then it can be reused many times. Due to Lemma 1 we can also decide $H\theta$-reducibility for any subfeature s in polynomial time and remove s from the set of already generated features if it is found reducible. What remains to explain is how HiFi deals with the maximum declared feature size, which is answered by the following lemma.

Lemma 3. *Let m denote number of declared predicates and let a denote maximum arity of the predicates. Then, for all declared predicates p, we can find sizes of the smallest features F_{min} containing p in time $O(m^2 + m \cdot a)$.*

Due to Lemma 3, which allows us to decide whether a subfeature may be extended to a correct feature with size less than n, and due to the fact that no feature f has more than n subfeatures, the number of generated subfeatures can be bounded at any time of HiFi's run by $n \cdot C(n)$, where $C(n)$ is the total number of correct features w.r.t. τ. Brief combinatorial reasoning then implies that HiFi indeed runs in time polynomial in the maximum feature length and in the total number of features.

Theorem 1. *Let τ be a hierarchical template and n be maximum feature size, then HiFi finishes in time polynomial in the total number of features and in n.*

5 Experiments

Here we evaluate HiFI in comparison to a state-of-the-art propositionalization algorithm RSD [5]. Due to limited space, we perform experiments only in one relational domain. In this experiment the same language bias is applied for HiFi and RSD. The experiments pertain to class-labeled CAD data (product structures) described in [6], consisting of 96 CAD examples each containing several hundreds of first-order literals. Table 1 displays the results. J48 refers to leave-one-out predictive accuracies of J48 decision trees built using the generated

features. The results indicate that for large features HiFi outperforms RSD by several orders of magnitude. However, for very small features HiFi's more complicated algorithms cause some overhead, which manifests itself in RSD being faster for small sizes.

Length	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
HiFi [s]	12	14	15	15	30	53	115	261	618
RSD [s]	0.5	1.5	8	45	310	1749	12324	n.a.	n.a.
J48 [%]	88.54	87.5	94.79	93.75	95.83	94.79	91.66	91.66	92.7

Table 1. Propositionalization results on CAD data.

6 Conclusions

We have presented a novel propositionalization algorithm HiFi based on constructing first-order features with hierarchical structure. Our experiments indicate that HiFi performs substantially faster than state-of-the-art propositionalization system RSD [5]. Experiments with more datasets and comparing HiFi to other ILP algorithms should be done in future work.

Acknowledgements
This work is supported by the Grant Agency of the Czech Republic through the project 201/08/0486 Merging Machine Learning with Constraint Satisfaction.

References

1. R. Barták. Theory and practice of constraint propagation. In *Proceedings of the 3rd Workshop on Constraint Programming for Decision and Control (CPDC2001)*, pages 7–14, 2001.
2. M.-A. Krogel, S. Rawles, F. Železný, P. A. Flach, N. Lavrač, and S. Wrobel. Comparative evaluation of approaches to propositionalization. In *Procs. of the 13th International Conf. on Inductive Logic Programming*, pages 197–214, 2003.
3. N. Lavrač and P. A. Flach. An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, 2(4):458–494, 2001.
4. J. Maloberti and E. Suzuki. An efficient algorithm for reducing clauses based on constraint satisfaction techniques. In *ILP*, volume 3194 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 2004.
5. F. Železný and N. Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62(1-2):33–63, 2006.
6. M. Žáková, F. Železný, J. Garcia-Sedano, C. Masia Tissot, N. Lavrač, P. Křemen, and J. Molina. Relational data mining applied to virtual engineering of product designs. In *Procs of the 16th Int. Conference on Inductive Logic Programming*, volume 4455 of *LNAI*, pages 439–453. Springer, 2007.
7. F. Železný. Efficient sampling in relational feature spaces. In *Proceedings of the 15th Int. Conf. on Inductive Logic Programming*, pages 397–413. Springer, 2005.

Learning Conceptual Predicates for Teleoreactive Logic Programs

Nan Li, David J. Stracuzzi, and Pat Langley

School of Computing and Informatics, Arizona State University
Tempe, Arizona 85281 USA
{nan.li.3|david.stracuzzi|langley}@asu.edu

Abstract. Teleoreactive logic programs provide a formalism for describing conceptual and skill knowledge that is organized hierarchically. However, manual construction of the conceptual clauses is tedious and often requires expert knowledge. In this paper, we present an approach to defining new conceptual predicates from successfully solved problems. We provide experimental results that demonstrate these concepts improve the usefulness of skills learned from the same solutions.

1 Introduction

Teleoreactive logic programs encode both declarative and procedural knowledge into hierarchical first-order knowledge bases [1] using a syntax similar to the first-order Horn clauses in Prolog. The term "teleoreactive" [2] refers to the formalism's support for reactive execution of the goal-oriented skills over time. Teleoreactive logic programs are often created manually using expert knowledge, but this approach is both tedious and time-consuming.

There has been a growing body of work on learning teleoreactive logic programs, hierarchical task networks, and related structures [1, 3]. An important subtask involves acquiring the preconditions for the learned procedural clauses. These determine when specific clauses apply, and therefore guide the system to select procedures that take it toward the goal. In this paper, we report an approach to defining new predicates that encode these preconditions in ways that improve the behavior of learned skills over that of previous methods.

2 A Review of Teleoreactive Logic Programs

Teleoreactive logic programs incorporate ideas from traditional logic programming, but differ in that they carry out action over time. The formalism combines techniques from goal-driven and reactive control, and it incorporates constraints that make learning of hierarchical structures tractable. In this section, we briefly review the basic assumptions and operational procedures that Langley and Choi [1] introduced in their early work on this topic.

Programs in this framework distinguish conceptual and procedural knowledge. The conceptual knowledge base comprises a hierarchy of first-order Horn

Table 1. Sample conceptual clauses from freecell solitaire.

```
;; cards ?c1 and ?c2 are of different color, rank of ?c2 is one larger than ?c1
((stackable ?c1 ?c2)
   :percepts   ((card ?c1 color ?co1 val ?v1)
                (card ?c2 color ?co2 val ?v2))
   :tests      ((not (equal ?co1 ?co2))
                (= ?v2 (+ 1 ?v1))))

;; card ?c may be placed onto card ?dc, and ?cb is the card below ?c
((movable ?c ?dc ?cb)
   :percepts   ((card ?c) (card ?dc) (card ?cb))
   :relations  ((clear ?c) (clear ?dc)
                (moved-onto ?c ?cb) (stackable ?c ?dc)))
```

clauses with negation that provide a vocabulary to describe the agent's environment. Each conceptual clause consists of a head, which states its predicate and arguments, and a body that describes the conditions under which the predicate is true, as Table 1 demonstrates. Procedural knowledge, stated as a set of skill clauses, is similar to hierarchical STRIPS operators [4]. Each skill clause has a head that refers to the skill's goal, a start condition that must be satisfied before it can execute, an action or subgoal field that describes how to achieve the goal, and an effects field that describes the situation after successful execution. The predicate in a clause head may appear in a subgoal, so the framework support recursive programs. Table 2 shows sample skill clauses from the freecell domain. Notice how predicates such as *movable* refer to the concepts defined in Table 1.

Teleoreactive logic programs perform two primary operations during each execution cycle. First, the interpreter carries out bottom-up inference to determine a belief state based on the agent's percepts and conceptual knowledge. Second, the interpreter retrieves the first unsatisfied top-level goal and attempts to find an applicable path through the skill hierarchy. Such a path starts from

Table 2. Sample skill clauses from freecell solitaire.

```
;; Clear card ?cb by moving the card ?c on top of it to card ?dc
((clear ?cb ?c ?dc)
   :percepts   ((card ?c) (card ?dc) (card ?cb))
   :start      ((movable ?c ?dc ?cb))
   :actions    ((*sendtocol ?c ?dc))
   :effects    ((clear ?cb) (moved-onto ?c ?dc) (clear ?c)))

;; Move card ?c on to card ?dc
((moved-onto ?c ?dc)
   :percepts   ((card ?c) (card ?dc) (card ?cb))
   :start      ((precondition-moved-onto_s8 ?c ?dc))
   :subgoals   ((movable ?c ?dc ?cb)
                (clear ?cb ?c ?dc))
   :effects    ((effect-moved-onto_s8 ?c ?dc))
```

the agent's goal, which is an instance of a known concept, and descends through the hierarchy such that the preconditions of each skill clause match and the bindings of each subgoal unify with those of its parent.

If no applicable skill path exists, a problem solver decomposes the goal by chaining backward using domain knowledge. The problem solver only backchains over concept definitions and primitive skills, which refer to executable actions rather than subgoals. To chain off of a skill, the interpreter retrieves a skill that contains the current goal in its effect and attempts to achieve the preconditions for that skill. Similarly, when chaining off a concept, the system uses its definition to decompose the current goal into multiple subgoals.

Whenever the problem solver achieves a goal or subgoal, it constructs a new skill clause. The head of the skill is a generalized version of the goal that replaces constants with variables. If chaining off a skill achieved the goal, the new skill's subgoals are the precondition concept from the chained skill, plus the subgoals of the chained skill in order of execution. The precondition of the new skill is the precondition of the skill that achieved the first subgoal. If chaining off a concept achieved the goal, the new skill's subgoals are the subconcepts that were unsatisfied at the start of problem solving.

When the system encounters a similar situation in the future, its interpreter will test whether the new skill clause appears in an applicable path through the skill hierarchy, in which case it will execute that path. Experimental studies suggested that this approach to learning hierarchical skills rapidly replaced problem solving, which often required extensive backtracking, with reactive execution, which often led directly to the goal.

3 Learning and Using Conceptual Predicates

Langley and Choi's [1] skill-learning method produced encouraging results, but analysis suggested it has two drawbacks. First, skill clauses produced from solutions obtained via chaining off of concepts tend to have overly general preconditions because they ignore the preconditions of skill clauses that achieve the subgoals. Second, skill clauses constructed for goals achieved via skill chaining tend to have overly specific preconditions, since they consider only the particular primitive skills used to achieve the first subgoal and ignore other skill clauses that may achieve that subgoal.

We have addressed these issues by developing an extended approach which defines new conceptual predicates that better reflect a learned skill clause's applicability. Preconditions and effects define abstractions of the world before and after the skill executes. We can expand the system's knowledge about the world by constructing new predicates that encode these abstractions effectively.

The new approach introduces two kinds of terms: specialized predicates and generalized predicates. Each specialized precondition/effect is associated with a skill clause and describes the situations in which that skill applies/produces. The system uses specialized preconditions during execution to determine which skill to apply next. A generalized precondition/effect is associated with a goal

and encodes a disjunction over the specialized preconditions/effects from all skill clauses that achieve the goal. Our approach uses generalized preconditions and effects during learning to determine the specialized precondition and effect for a new skill that admits all possible uses of that skill, as detailed below.

The input to our method is a skill clause, built by the skill learner, which contains a goal and a list of subgoals. If the system achieved the goal by chaining off a concept, it first retrieves the generalized preconditions and effects associated with the subgoals. It then uses macro-operator composition [5] to compute the preconditions and effects by combining the generalized preconditions and effects of the subgoals. Finally, the system introduces new precondition and effect predicates using these two combined terms as their definitions. Similarly, if the system achieved the goal by chaining off a skill, the specialized precondition of the new skill is the generalized precondition of the first subgoal, S_1. Similarly, the specialized effect of the learned skill clause is the effect of the original skill.

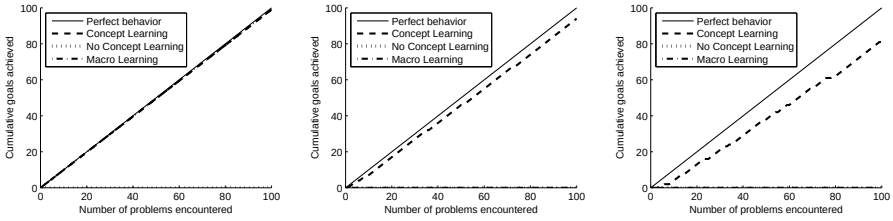
Irrespective of whether the problem solver used concept or skill chaining to achieve the goal, the algorithm updates the generalized predicates corresponding to the new specialized predicates by adding the precondition/effect predicates as disjunctive terms. This informs the interpreter that the given goal can be achieved in a new situation and lets it apply learned skill clauses to goals under circumstances in which it would otherwise have ignored them.

4 Experimental Evaluation

Our key claim is that expanding the representation of teleoreactive logic programs by defining new conceptual predicates improves the ability of learned skills to achieve goals and reduces their reliance on problem solving. To test this claim, we carried out an experiment that compared the new approach with the one that Langley and Choi [1] reported and with Mooney's [5] method for learning a non-hierarchical macro-operators. The latter determines preconditions using an analytic techniques similar to the one we described for computing specialized preconditions, but without introducing new predicates. We used the same inference, execution, and problem-solving modules in each case.

We presented each system with 100 randomly selected problems from the freecell domain [6]. For each problem, a system first tried to achieve the goal by executing its existing skills. If this failed, it called on the problem solver and learned new skill clauses, using one of the above three methods, whenever this achieved a goal or subgoal. We measured system behavior as the number of top-level goals achieved by execution without resorting to problem solving. Other metrics such as CPU time are secondary to our objectives. We provided the systems with initial knowledge bases (40 conceptual clauses and 13 primitive skills) sufficient to solve problems by execution that were one step away from the goal. We tested the system on problems with 8, 16 and 24 cards.

Figure 1 displays the cumulative number of goals achieved without problem solving by the three systems. Macro-operator learning fares the worst, while our predicate creation method learns more rapidly than Langley and Choi's tech-



(a) (b) (c)

Fig. 1. Cumulative number of goals achieved for the freecell solitaire domain on problems involving (a) eight cards, (b) 16 cards, and (c) 24 cards.

nique on tasks with 16 and 24 cards. Analysis of individual runs shows that the older approach often acquires overly general preconditions, which leads the system to execute skills that do not achieve the goal. The concept learner mitigates this drawback by learning more specific preconditions. Conversely, the macro-operator method collects too many relations among cards, which leads to overly specific preconditions that keep the interpreter from executing relevant skills. We conclude that, on average, the new mechanism creates more appropriate preconditions for skill clauses that reduce the chance of selecting irrelevant learned skills and increase the chance of selecting relevant ones.

5 Concluding Remarks

In this paper, we reviewed teleoreactive logic programs, along with an initial approach to learning them from problem solutions. We also identified some drawbacks with this scheme and described an extension for defining new conceptual predicates to produce more appropriate preconditions on learned skill clauses. An experiment demonstrated that the new mechanism learned more rapidly than either the initial technique, which formed overly general conditions, or a method based on macro-operators formation, which formed overly specific ones.

Our predicate creation algorithm has superficial similarities to work on predicate invention [7], but our approach is analytic while the latter was driven by empirical regularities. More closely related is research on representation change in problem solving and game playing [8, 9], which also relied on goal-driven analytical learning. However, our approach differs from these efforts by supporting incremental learning that is interleaved with problem solving, by acquiring recursive precondition concepts that aid generalization, and by working jointly with a method for constructing hierarchical skills that support reactive execution. Also relevant is recent work on learning hierarchical methods from problem solutions and action models [10–12], which shares many features but does not construct new conceptual predicates that extend the representation language.

The main claim of this paper is that the analytic creation of new conceptual predicates produces more appropriate preconditions for learned skill clauses,

which in turn let a teleoreactive interpreter achieve more goals through execution, without the need for problem solving. However, the results we have reported remain preliminary and suggest several avenues for additional research. For example, we should replicate out experimental studies in other domains, both to demonstrate generality and better understand the quality of the learned preconditions. We must also combine the predicate learner with an evaluation mechanism that lets the system determine which of the new concepts are useful. Finally, we should augment the framework to acquire skills for achieving these invented concepts, thus closing the loop on conceptual and procedural learning.

6 Acknowledgements

The authors would like to thank Dongkyu Choi and Tolga Konik for helpful discussions and suggestions concerning this work. This material is based on research sponsored by ONR under grant N00014-08-1-0069 and by DARPA under agreement FA8750-05-2-0283.

References

1. Langley, P., Choi, D.: A unified cognitive architecture for physical agents. In: Preceedings of the Twenty-First National Conference on Artificial Intelligence, Boston, AAAI Press (2006)
2. Nilsson, N.: Teleoreactive programs for agent control. Journal of Artificial Intelligence Research **1** (1994) 139–158
3. Ilghami, O., Nau, D.S., Muñoz–Avila, H., Aha, D.W.: Camel: Learning method preconditions for HTN planning. In: Proceedings of the Sixth International Conference on AI Planning and Scheduling, Toulouse, France, AAAI Press (2002) 131–141
4. Fikes, R., Nilsson, N.: Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence **2** (1971) 189–208
5. Mooney, R.J.: A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding. Morgan Kaufmann, San Mateo, CA (1990)
6. Bacchus, F.: AIPS '00 planning competition. AI Magazine **22** (2001) 47–56
7. Muggleton, S.: Predicate invention and utility. Journal of Experimental and Theoretical Artificial Intelligence **6**(1) (1994) 121–130
8. Utgoff, P.E.: Shift of Bias for Inductive Concept Learning. PhD thesis, Department of Computer Science, Rutgers University, New Brunswick, NJ (1984)
9. Fawcett, T.: Knowledge-based feature discovery for evaluation functions. Computational Intelligence **12**(1) (1996)
10. Reddy, C., Tadepalli, P.: Learning goal decomposition rules using exercises. In Fisher, D., ed.: Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, TN, Morgan Kaufmann (1997)
11. Nejati, N., Langley, P., Konik, T.: Learning hierarchical task networks by observation. In: Proceedings of the 23nd International Conference on Machine Learning, Pittsburgh, PA, ACM (2006)
12. Hogg, C., Muñoz-Avila, H., Kuter, U.: HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required. In: Proceedings of the Twenty-Third Conference on Artificial Intelligence, Chicago, AAAI Press (2008)

Predicting Gene Coexpression from Pathway Relations

Karel Moulík and Filip Železný

Intelligent Data Analysis Research Group
Dept. of Cybernetics, Czech Technical University in Prague
http://ida.felk.cvut.cz
e-mail: {moulik1,zelezny}@fel.cvut.cz

Abstract. We address the problem of learning to predict the coexpression of gene pairs, given background knowledge consisting of the descriptions of biological pathways in which the genes are involved. We formalize this as a non-standard graph-mining problem and devise an algorithm to solve it. Our experiments conducted on yeast gene expression data indicate an under-fitting tendency of the algorithm, calling for a more expressive formalism, such as ILP, to be employed for this biologically significant problem.

1 Introduction

As the amount of facts we can systematically store about life's biological nature steeply increases, various approaches come to stage offering us ways of reusing the facts in order to mine new knowledge. An effort aimed at discovering the principles of gene coexpression is one of the examples. The particular research question we tackle here is whether the coexpression of gene pairs can be explained in terms of gene-gene relations implied by known biochemical pathways.

Pathways are representations of cellular processes pertaining to metabolism, gene regulation, signalling, etc. They usually take the form of directed graphs with labeled vertices and labeled edges. Vertices represent chemical compounds such as proteins and protein complexes. These often can be mapped to genes or gene sets by which their structure is encoded. Edges stand for various kinds of relations such as inhibition or activation.

Efforts to explain coexpression by relations derived from pathways receive due attention in biology research. It was for example shown that the correlation of activity between two genes decreases monotonically with the network distance among the two genes in a pathway [6]. In [2] an idea was introduced that gene corregulation can be accurately predicted using so called "flux coupling" in metabolic pathways. A metabolic flux can be viewed as a path between two nodes in the metabolic pathway graph, such that this path represents a directed current of metabolites with rare in-fluxes and no ex-fluxes. It was shown that gene couples interconnected by a flux are more probable to be active at the same time.

The study [6] went further to propose so called *network motives* (roughly corresponding to *graph patterns* in data mining terminology) which were empirically found powerful at explaining coregulation. The obvious assumption of this approach is that certain structural patterns indeed exist that are present in a significant number of pathways. One may legitimately challenge this assumption on the grounds of the diversity of processes described by the respective pathways. This point of doubt is mitigated by the recent study [1] showing strong similarities in network content not only across different species but also within the pathways of a single organism.

To summarize, numerous papers have been published hypothesising about gene coexpression from the pathway structure perspective. To our best knowledge, however, in all cases the usual life-science research methodology has been followed, in that a hypothesis is coined first by expert intuition, and computational means are used only later to test the hypothesis against empirical gene expression data.

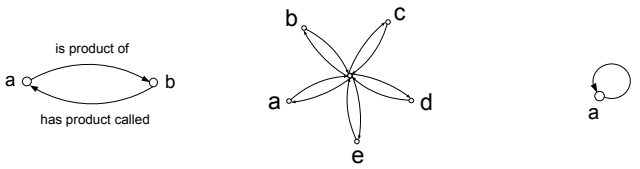
In this work we want approach the problem in a reversed way. We will try to induce the pattern of gene coexpression automatically, based on positive and negative samples of coexpressed genes (i.e. on pairs of genes that, respectively, indeed are coexpressed and those that are not) and pathway descriptions acting as background knowledge.

2 Experimental Data

To obtain learning examples for gene coexpression, we relied on study [3] which provides a correlation matrix for the expression of genes of the yeast genome. The first 800 most correlated gene pairs with at least 6 facts known about both the genes in the pair were selected as the set of positive samples of the coexpression relation. The 800 least correlated samples subject again to the former constraint were selected as the negative set.

Background knowledge, i.e. graphs describing pathways, was extracted from the Kyoto Encyclopedia of Genes and Genomes (KEGG) [5]. KEGG is a manually updated collection of pathways for many organisms. Formally, their structure corresponds to hypergraphs. In a hypergraph, a single edge can connect any number of nodes. The nodes are genes, enzymes or compounds, the edges are reactions or further relations in between them. A certain type of enzyme-enzyme relation, indicating two enzymes catalyzing successive reaction steps, is an example of an edge connecting 3 nodes: the two enzymes and a compound shared with two successive reactions.

In order to make the representation suitable for pattern searching we merged the set of pathway hypergraphs into a single oriented graph, called the *integrated pathway*. This change of representation, where multiple biological (hyper)graphs are joined into a single graph with rather rich semantics (i.e. a large alphabet for labels), is popular in current systems biology research and is represented e.g. by the popular Ingenuity Networks software. Our particular approach was as follows.



(a) Subgraph corresponding to an edge connecting 2 nodes in hypergraph.

(b) Subgraph corresponding to an edge connecting 5 nodes in hypergraph.

(c) Subgraph corresponding to an edge expressing an unary relation.

Fig. 1. Hypergraph to graph conversion

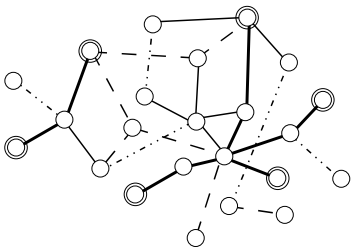


Fig. 2. An abstract example of the task addressed. Coexpressed genes (vertices) are inside circles. The coexpression pattern to be discovered here is "genes connected by a path consisting of thick continuous edges."

For each gene, compound and enzyme to be found in pathways a single unique node was created. For edges connecting 2 nodes A and B in some pathway, two oriented edges were added to corresponding nodes in the oriented graph: the first expressing the original relation $A \rightarrow B$, the second for inverted one, i.e. $A \leftarrow B$; see figure 1(a) (the duality of edges will be enlightened later in this article). For edges connecting $x > 2$ nodes an auxiliary anonymous node was created and $x * 2$ edges were connected to it (x of them leading from the auxiliary node to nodes corresponding to hypergraph and x in the opposite direction). This can be viewed as a sort of decomposition into graph edges; see figure 1(b). For edges expressing an unary relation a single edge was added to corresponding node in the oriented graph as you can see in figure 1(c).

3 Predicting Coexpression

Here we want to obtain an initial assessment of the feasibility of the problem approached by our work. For simplicity we first rely on means of limited expressivness and formalize the problem in the graph mining framework.

Established graph mining algorithms typically aim at discovering an as large as possible graph homomorphic to subgraphs present in a sufficient number of example graphs. The nature of our problem however requires a different formulation.

Figure 2 presents a simple example of the pattern discovery task we want to address. The exemplified pattern obviously can be easily transcribed into a first-order logic as:

$$Corregulated(x, y) \leftarrow Green(x, y)$$

$$Corregulated(x, y) \leftarrow Green(x, z) \wedge Corregulated(z, y)$$

In the elected graph framework we need to design a special construct able to express a pattern that in first-order logic would acquire a recursive form. For this purpose we define the *string of edges* concept.

Definition 1. *A* **string of edges** *in between nodes g_s and g_e is a succession of edge types $t_0t_1t_2...t_n$ such that it is possible to start in g_s, move along an oriented edge of type t_0 from g_s to x_0, from which node x_1 can be reached via an oriented edge of type t_1 and so on till g_e is reached.*

Definition 2. *The* **coexpression graph mining task**. *We are given an oriented graph M representing an integrated pathway and a set G of gene identifiers. We shall assume that labels of all gene nodes $g \in M$ are from G. We are further given a set of positive examples $E^+ \subset G \times G$ (coexpressed gene pairs) and negative examples $E^- \subset G \times G$ (non-coexpressed genes), $E^+ \cap E^- = \emptyset$. A desired result of our envisioned graph mining algorithm is a string of edges γ such that*

1. *For as many as possible $(g_1, g_2) \in E^+$ and as few as possible $(g_1, g_2) \in E^-$ there is an edge string γ in between g_1 and g_2.*
2. *γ is as short as possible.*

The algorithm (Alg. 1) we propose to solve the stated problem follows the usual artificial intelligence cookbook in that it is essentially a best-first heuristic search algorithm, which reduces its search space by exploiting the constraint that only a certain set of lengths of edge strings exists between the two genes in question. As the heuristic measuring the quality of edge strings we use weighted relative accuracy (WRAcc) [4] and we adhere to the weighted covering strategy [4] for the gradual removal of covered examples.

4 Preliminary Results

Table 1 summarizes the predictive accuracy of our algorithm estimated through 10-fold cross-validation. Given the balanced sample we worked with, the majority vote accuracy is 0.5.

Algorithm 1 *searchForPatterns*($sampleSet, maxLength, maxQSize$)

1: **Input:** Set of samples *sampleSet*, Maximum allowed edge string length *maxLength*, Maximum priority queue length *maxQSize*;

2: Initialize a priority queue Q; Set its maximum length to *maxQSize*
{Q is a priority queue of 3-tuples [node,edge string,length] ordered by the quality of edge strings (in terms of the WRAcc heuristic, see main text).}

3: Set $BEST$ to nothing.
{$BEST$ is a best so far edge string found.}

4: Select a gene pair [GS,GE] that was not yet covered from positive part of *sampleSet*.
5: $L \leftarrow$ All possible lengths of edge strings from GS to GE up to maxLength.
6: **for** $\forall l \in L$ **do**
7: Add 3-tuple [GS,e,l] to Q (e being an empty edge string).
8: **end for**

9: **while** Q is not empty **do**
10: $A \leftarrow Top(Q)$ ($N \leftarrow A[0]$; $ES \leftarrow A[1]$; $D \leftarrow A[2]$)
11: Pop(Q)
12: **for** $\forall E \in \{E \in edges |$ E is incident to N and leads to X that is in distance $D-1$ from GE$\}$ **do**
13: **if** $D-1 < 0$ **and** ES is better* than $BEST$ (in terms of the WRAcc heuristic, see main text) **then**
14: BEST$\leftarrow$ES
15: **else**
16: Add 3-tuple [X,NES,$D-1$] to Q, where NES is ES elongated by edge E
17: **end if**
18: **end for**
19: **end while**

20: Write $BEST$ to output and increment coverage of each pair $BEST$ covers.

21: If there is an uncovered positive gene pair in *sampleSet* **goto step 2.**

5 Conclusions and Future Work

We presented a graph mining approach for the prediction of gene coexpression from biological pathway data and tested it on data integrated from a yeast gene expression database and the KEGG database of pathways. We designed a graph pattern concept which overcomes one of the expressivness limitations of graphs vis-a-vis first-order logic. In particular, through the proposed *edge-string* pattern type we can, to a limited extent, express paths in graphs. The most imporant empirical observations gained from a 10-fold cross-validation procedure are (i) relatively small predictive accuracy: up to 64% compared to 50% of the baseline majority vote accuracy, and (ii) extremely small differences between training and testing accuracies. These two observations suggest that we are far on the

Ruleset size	TrnPre	TstPre	TrnGPre	TstGPre	TrnAcc	TstAcc	TrnGAcc	TstGAcc
10	0.696	0.692	0.632	0.634	0.582	**0.583**	0.640	0.640
9	0.703	0.699	0.641	0.638	**0.586**	**0.583**	**0.648**	**0.641**
8	0.701	0.696	0.636	0.634	0.565	0.563	0.626	0.623
7	0.704	0.702	0.648	0.649	0.573	0.572	0.628	0.630
6	0.712	0.702	0.655	0.659	0.561	0.558	0.617	0.612
5	0.715	0.713	0.682	0.665	0.556	0.555	0.618	0.607
4	0.741	**0.749**	0.676	0.656	0.556	0.553	0.599	0.593
3	0.720	0.733	0.706	0.707	0.562	0.560	0.616	0.615
2	**0.742**	0.715	0.729	0.728	0.541	0.541	0.566	0.569
1	0.723	0.738	**0.723**	**0.738**	**0.586**	0.577	0.586	0.577

Table 1. Gene coexpression pattern ruleset preliminary results:

Ruleset size = the number of rules written to output
$TrnPre$ = average precision of a rule in train set (crossvalidated)
$TstPre$ = average precision of a rule in test set (crossvalidated)
$TrnAcc$ = average accuracy of a rule in train set (crossvalidated)
$TstAcc$ = average accuracy of a rule in test set (crossvalidated)
$TrnGPre$, $TrnGAcc$, $TstGPre$, $TstGAcc$ measures for ruleset size i are related to a single classifier which consists of all the rules $1 \dots i$, and predicts that the two genes are coexpressed whenever one of these rules so predicts.

under-fitting end of the bias-variance trade-off and they literally call for using a more expressive formalism for this task of high biological significance. We are happy to forward this message to the ILP community.

Acknowledgement The authors are supported by the Czech Academy of Sciences through the project 1ET101210513 Relational Machine Learning for Biomedical Data Analysis and the Czech Technical University grant CTU0814413.

References

1. Cootes A. A., Muggleton S. H., and Sternberg M. J.E. The identification of similarities between biological networks: Application to the metabolome and interactome. *JMB*, 2007.
2. Notebaart R. A., Teusink B., Siezen R. J., and Papp B. Co-regulation of metabolic genes is better explained by flux coupling than by network. *PLOS Computational Biology*, 2008.
3. Zhang B. and Horvath S. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 2005.
4. Železný F. and Lavrač N. Propositionalization-based relational subgroup discovery with rsd. *Machine Learning*, 2005.
5. Kanehisa M., Araki M., Goto S., Hattori M., Hirakawa M., Itoh M., Katayama T., Kawashima S., Okuda S., Tokimatsu T., and Yamanishi Y. Kegg for linking genomes to life and the environment. *Nucleic Acids Research*, 2008.
6. Kharchenko P., Church G. M., and Vitkup D. Expression dynamics of a cellular metabolic network. *Molecular Systems Biology*, 2005.

TopLog: ILP using a logic program declarative bias

Stephen H. Muggleton, José C. A. Santos and Alireza Tamaddoni-Nezhad

Department of Computing, Imperial College, London
{shm,jcs06,atn}@doc.ic.ac.uk

Abstract. This paper introduces a new Inductive Logic Programming (ILP) framework called Top Directed Hypothesis Derivation (TDHD). In this framework each hypothesised clause must be derivable from a given logic program called top theory ($\top$). The top theory can be viewed as a declarative bias which defines the hypothesis space. This replaces the metalogical mode statements which are used in many ILP systems. Firstly we present a theoretical framework for TDHD and show that a standard SLD derivation can be used to efficiently derive hypotheses from $\top$. Secondly, we present a prototype implementation of TDHD within a new ILP system called TopLog. Thirdly, we show that the accuracy and efficiency of TopLog, on several benchmark datasets, is comparable with the accuracy and efficiency of a state of the art ILP system like Aleph.

1 Introduction

In this paper we introduce a new approach to providing declarative bias called Top-Directed Hypothesis Derivation (TDHD). The approach extends the use of the $\bot$ clause in Mode-Directed Inverse Entailment (MDIE) [5]. In Inverse Entailment $\bot$ is constructed for a single, arbitrarily chosen training example. Refinement graph search is then constrained by the requirement that all hypothesised clauses considered must subsume $\bot$. In TDHD we further restrict the search associated with each training example by requiring that each hypothesised clause must also be entailed by a given logic program, $\top$.

The $\top$ theory can be viewed as a form of first-order declarative bias which defines the hypothesis space, since each hypothesised clause must be derivable from $\top$. The use of the $\top$ theory in TopLog is also comparable to grammar-based declarative biases [2, 4]. However, compared with a grammar-based declarative bias, $\top$ has all the expressive power of a logic program, and can be efficiently reasoned with using standard logic programming techniques.

The SPECTRE system [1] employs an approach related to the use of $\top$. SPECTRE also relies on an overly general logic program as a starting point. However, unlike the TopLog system described in this paper, SPECTRE proceeds by successively unfolding clauses in the initial theory. TDHD is also related to Explanation-Based Generalisation (EBG) [3]. However, like SPECTRE, EBG does not make the key MDHD distinction between the $\top$ theory and background knowledge. Moreover, EBG is viewed as a form of deductive learning, while the clauses generated by TDHD represent inductive hypotheses.

This paper is arranged as follows: Section 2 provides a theoretical framework for TDHD. This framework is then used as the basis for Hypothesis Generation in TopLog as described in Section 3. Experiments comparing speed and accuracy of TopLog and Aleph are given in Section 4. In Section 5 we conclude and discuss further work.

2 Theoretical framework

MDIE was introduced in [5] as the basis for Progol. The input to an MDIE system is the vector $S_{MDIE} = \langle M, B, E\rangle$ where M is a set of mode statements, B is a logic program representing the background knowledge and E is set of examples. M can be viewed as a set of metalogical statements used to define the hypothesis language $\mathcal{L}_M$. The aim of the system is to find consistent hypothesised clauses H such that for each clause $h \in H$ there is at least one positive example $e \in E$ such that $B, h \models e$.

The input to an TDHD system is the vector $S_{TDHD} = \langle NT, \top, B, E\rangle$ where NT is a set of "non-terminal" predicate symbols, $\top$ is a logic program representing the declarative bias over the hypothesis space, B is a logic program representing the background knowledge and E is a set of examples.

The following three conditions hold for clauses in $\top$: (a) each clause in $\top$ must contain at least one occurrence of an element of NT while clauses in B and E must not contain any occurrences of elements of NT, (b) any predicate appearing in the head of some clause in $\top$ must not occur in the body of any clause in B and (c) the head of the first clause in $\top$ is the target predicate and the head predicates for other clauses in $\top$ must be in NT.

The aim of a TDHD system is to find a set of consistent hypothesised clauses H, containing no occurrence of NT, such that for each clause $h \in H$ there is at least one positive example $e \in E$ such that the following two conditions hold: (1) $\top \models h$ and (2) $B, h \models e$.
Due to space limitation we omit the proof of the following theorem.

Theorem 1. *Given $S_{TDHD} = \langle NT, \top, B, E\rangle$ assumptions (1) and (2) hold only if for each positive example $e \in E$ there exists an SLD refutation R of $\neg e$ from $\top, B$, such that R can be re-ordered to give $R' = D_h R_e$ where D_h is an SLD derivation of a hypothesis h for which (1) and (2) hold.*

According to Theorem 1, implicit hypotheses can be extracted from the refutations of a positive example $e \in E$. Let us now consider a simple example.

Example 1. Let $S_{TDHD} = \langle NT, \top, B, E\rangle$ where NT, B , e and $\top$ are defined as follows:

$$NT = \{\$body\}$$
$$B = b_1 = \text{pet(lassy)} \leftarrow$$
$$e = \text{nice(lassy)} \leftarrow$$
$$\top = \begin{cases} \top_1 : \text{nice}(X) \leftarrow \$\text{body}(X) \\ \top_2 : \$\text{body}(X) \leftarrow \text{pet}(X) \\ \top_3 : \$\text{body}(X) \leftarrow \text{friend}(X) \end{cases}$$

Given the linear refutation $R = \langle \neg e, \top_1, \top_2, b_1\rangle$, we now construct the re-ordered refutation $R' = D_h R_e$ where $D_h = \langle \top_1, \top_2\rangle$ derives the clause $h = \text{nice(X)} \leftarrow \text{pet(X)}$ for which (1) and (2) hold.

3 System Description

TopLog is a prototype ILP system developed by the authors to implement the TDHD described in section 2. It is fully implemented in Prolog and is ensured to run at least in YAP, SWI and Sicstus Prolog. It is publicly available at http://www.doc.ic.ac.uk/∼jcs06 and may be freely used for academic purposes.

3.1 From mode declarations to ⊤ theory

As the user of TopLog may not be familiar with specifying a search bias in the form of a logic program, TopLog has a module to build a general ⊤ theory automatically from user specified mode declarations. In this way input compatibility is ensured with existing ILP systems. Below is a simplified example of user specified mode declarations and the automatically constructed ⊤ theory.

modeh(mammal(+animal)).
modeb(has_milk(+animal)).
modeb(has_eggs(+animal)).

$$\top = \begin{cases} \top_1 : \text{mammal}(X) \leftarrow \$\text{body}(X). \\ \top_2 : \$\text{body}(X) \leftarrow .\%\mathit{emptybody} \\ \top_3 : \$\text{body}(X) \leftarrow \text{has\_milk}(X), \$\text{body}(X). \\ \top_4 : \$\text{body}(X) \leftarrow \text{has\_eggs}(X), \$\text{body}(X). \end{cases}$$

Fig. 1. Mode declarations and a ⊤ theory automatically constructed from it

The above illustrated ⊤ theory is extremely simplified. The actual implementation has stricter control rules like: variables may only bind with others of the same type, a newly added literal must have its input variables already bound.

It is worth pointing out that the user could directly write a ⊤ theory specific for the problem, potentially restricting the search better than the generic ⊤ theory built automatically from the mode declarations.

3.2 TopLog Learning Algorithm

The TopLog learning algorithm consists of three major steps: 1) hypotheses derivation for each positive example, 2) coverage computation for all unique hypotheses, H, derived in previous step, 3) construct the final theory, T, as the subset of H that maximizes a given score function (e.g. compression).

Hypotheses derivation In TopLog, contrary to MDIE ILP systems, there is no construction of the bottom clause but rather an example guided generalization, deriving all hypotheses that entail a given example with respect to the background knowledge.

The hypothesis derivation procedure is composed of two distinct steps. In the first step an example is proved from the background knowledge and the ⊤ theory. That is, the ⊤ theory is executed having the example matching the head of its start clause (i.e. $\top_1$). This execution yields a proof consisting of a sequence of clauses from the ⊤ theory and background knowledge.

For instance, using the ⊤ theory from figure 1 and $B = b_1 = \text{has\_milk(dog)}$ to derive refutations for example $e = \text{mammal(dog)}$, the following two refutations would be yielded: $r_1 = \langle \neg e, \top_1, \top_2 \rangle$ and $r_2 = \langle \neg e, \top_1, \top_3, b_1, \top_2 \rangle$.

In the second step, Theorem 1 is applied to r_1 and r_2 deriving, respectively, the clauses $h_1 = mammal(X)$ from $\langle \top_1, \top_2 \rangle$ and $h_2 = mammal(X) \leftarrow has\_milk(X)$ from $\langle \top_1, \top_3, \top_2 \rangle$.

Coverage computation Each $h \in H$ is individually tested with all the examples (positives and negatives) to compute its coverage (i.e. the examples it entails). The positive examples that were used to derive h do not need to be tested for entailment as it is guaranteed by the hypothesis derivation procedure that h entails them.

Constructing the final theory The final theory to be constructed, T, is a subset H' of H that maximizes a given score function (e.g. compression, coverage, accuracy). Each $h \in H$ has associated the set of examples from which it was derived, Eg_h, and the set of examples which it entails, Ec_h.

The compression score function (the default) evaluates T as the weighted sum of the examples it covers (positive examples have weights > 0 and negative examples < 0) minus number of literals in T. This is the minimum description length principle and is analogous to Progol's and Aleph's compression measure. T is constructed using a greedy approach where at each step the hypothesis, if any, that maximizes current T' score is added to the next round.

Efficient cross-validation Prior to N fold cross-validation all possible hypotheses are derived and their coverage is computed on all examples. This is the most time consuming step. Then, examples are randomly assigned a fold and N theories are built each using a distinct combination of $N - 1$ folds as training and one fold as testing.

Hypotheses generated exclusively from examples in the test set are not eligible for the theory construction step. Also, the merit of an hypothesis is evaluated only taking into account the hypothesis coverage on examples belonging to the training folds. At the end of cross-validation, N fold average training and test accuracies and standard deviations are reported.

It is not possible to do efficient cross-validation with Aleph or Progol as no relationship exists between hypotheses and the examples that generated it.

4 Experimental Evaluation

Materials In order to empirically evaluate TopLog we used four datasets: mutagenesis [7], carcinogenesis [6], alzheimers-amine [9] and DSSTox [10] mainly because they are well known to the ILP community and are good examples of practical problems where relational knowledge is important.

In these datasets the purpose is to characterize an active molecule (for the problem at hand). It is given examples of molecules that exhibit the property (i.e. positives) and examples of molecules that do not exhibit the property (i.e. negatives) together with background knowledge.

The task of the ILP system is to find a theory that entails as most of the positive examples while entailing as few of the negative examples as possible.

Methods TopLog was compared with Aleph because Aleph is a MDIE ILP system and is also implemented in YAP Prolog. The experiments were performed on a Intel Core 2 Duo @ 2.13 GHz with 2Gb of RAM using Ubuntu Linux with kernel version 2.60.3. Aleph current version (5.0) is publicly available at [8]. The four datasets may be freely downloaded from TopLog's webpage as well. Both TopLog and Aleph were executed on the latest YAP (version 5.1.3).

Aleph and TopLog were executed with as close as possible settings to ensure a fair test. Clause length (i.e. maximum literals in the body of an hypothesis) was set to 4 (except in DSSTox where it was set to 10), noise set to 100%, evaluation function set to compression, and number of search nodes (i.e. hypotheses) per example set to 1000.

Aleph was called both with *induce* and *induce_max* settings. The difference is that *induce* (the default), after finding a compressive clause for an example, retracts all positive examples covered by that clause while *induce_max* does not. TopLog also does not retract any example during the search and thus one should compare TopLog times with *induce_max* times rather than *induce*.

Results and Discussion The table below the time column has the running time, in CPU seconds, the ILP system took to build the model in the training data (Train column) and the total time it took to build the models for the ten folds (CV column). We distinguish between the two times to highlight the benefits of the efficient cross validation in TopLog. The accuracy column has the average (over the ten folds) percentage of correct predictions made by the model with the respective standard deviation.

	Aleph with induce			Aleph with induce_max			TopLog		
		Times			Times			Times	
Dataset	CV Accuracy	Train	CV	CV Accuracy	Train	CV	CV Accuracy	Train	CV
Mutagenesis	77.2%±9.2%	0.4s	4s	68.6%±11.4%	2s	17s	70.2%±11.9%	0.4s	0.5s
Carcinogenesis	60.9%±8.2%	6s	54s	65.1%±8.6%	29s	245s	64.8%±6.9%	7.0s	7.4s
Alzheimers	67.2%±5.0%	5s	40s	72.6%±6.2%	18s	156s	70.4%±5.6%	17s	16s
DSSTox	70.5%±6.5%	30s	253s	71.3%±3.4%	82s	684s	71.7%±5.6%	3.4s	3.6s

Table 1. Accuracy and time comparison between Aleph and TopLog

If we only consider the training time, TopLog is always faster than Aleph with the *induce_max* setting. Comparing with the *induce* setting the advantage is not clear (e.g. in Alzheimers Aleph is much faster than TopLog but in DSSTox the reverse occurs). Considering cross validation then TopLog is clearly faster. Although this may seem a side point, built-in efficient cross validation is important in practical applications in order to assess properly the model accuracy. The accuracies are identical with none being statistically significantly different.

5 Conclusions and Future work

The key innovation of the TDHD framework is the introduction of a first order $\top$ theory. We prove that SLD derivation can be used to efficiently derive hypotheses from $\top$. A new general ILP system, TopLog, is described implementing TDHD.

The empirical comparison demonstrates that the new approach is competitive, both in predictive accuracy and speed, with a state of the art system like Aleph. Below we discuss future work and some limitations of TopLog.

Parallelization Due to the way hypotheses are built in TopLog, where the construction of the set of hypotheses that covers one example is independent of the construction of the hypotheses set for the other examples, it is straightforward to parallelize TopLog main algorithm by dividing the number of examples by the number of processors available.

Sample hypothesis space Currently the hypothesis space is searched according to the $\top$ theory automatically constructed from the user mode declarations. Although this approach seems to work well in practice, for some problems possible clusters of interesting hypotheses may not be considered.

Acknowledgments

We thank James Cussens for illuminating discussions on the TDHD framework and Vítor Santos Costa for a great Prolog system and prompt help in fixing YAP's problems allowing us to stretch it to the limits. The first author thanks the Royal Academy of Engineering and Microsoft for funding his present 5 year Research Chair. The second author was supported by a Wellcome Trust Ph.D. scholarship. The third author was supported by the BBSRC grant BB/C519670/1. We are indebted to three anonymous referees for valuable comments.

References

1. H. Boström and P. Idestam-Almquist. Specialisation of logic programs by pruning SLD-trees. In S. Wrobel, editor, *Proceedings of the Fourth Inductive Logic Programming Workshop (ILP94)*, pages 31–48, Bonn, 1994. GDM-studien Nr. 237.
2. W. Cohen. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68:303–366, 1994.
3. S.T. Kedar-Cabelli and L.T. McCarty. Explanation-based generalization as resolution theorem proving. In P. Langley, editor, *Proc. of the Fourth Int. Workshop on Machine Learning*, pages 383–389,Los Altos, 1987. Morgan Kaufmann.
4. S. Džeroski and L. Todorovski. Discovering dynamics: From inductive logic programming to machine discovery. *Journal of Int. Inf. Systems*, 4(1):89–108, 1995.
5. S.H. Muggleton. Inverse entailment and Progol. *NGC*, 13:245–286, 1995.
6. A. Srinivasan, , R.D. King S.H. Muggleton, and M. Sternberg. Carcinogenesis predictions using ILP. In *Proceedings of the Seventh International Workshop on ILP*, pages 273–287. Springer-Verlag, Berlin, 1997. LNAI 1297.
7. A. Srinivasan, S. Muggleton, R. King, and M. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the Fourth International Inductive Logic Programming Workshop*. Gesellschaft fur Mathematik und Datenverarbeitung MBH, 1994. GMD-Studien Nr 237.
8. Ashwin Srinivasan. *The Aleph Manual*. University of Oxford, 2007.
9. R.D. King, A. Srinivasan, and M.J.E. Sternberg. Relating chemical activity to structure: an examination of ILP successes. *New Gen. Comp.*, 13:411–433, 1995.
10. A.M. Richard and C.R. Williams. Distributed structure-searchable toxicity DSSTox public database network: A proposal. *Mutation Research*, 499:27–52. 2000.

Mutlirelatonal GUHA Method and Genetic Data

Martin Ralbovský, Alexander Kuzmin, Jan Rauch

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
martin.ralbovsky@gmail.com, alexander.kuzmin@gmail.com, rauch@vse.cz

Abstract. The paper presents multirelational GUHA method, focusing on multirelational association rules. Background and principles of the method are introduced together with comparison with related methods. New implementation in the Ferda tool is presented and initial experiments in the genetic domain are shown.

Keywords: GUHA method, Multirelational GUHA, 4FT, virtual attribute, Ferda, genetic data

1 Introduction

The GUHA method is one of the first methods of exploratory data analysis, which has been in development since the mid-sixties. It is a general mainframe for retrieving interesting knowledge from data. The method has firm theoretical foundations based on logics, especially observational calculi and statistics [3]. Figure 1 shows the main principle of the method.

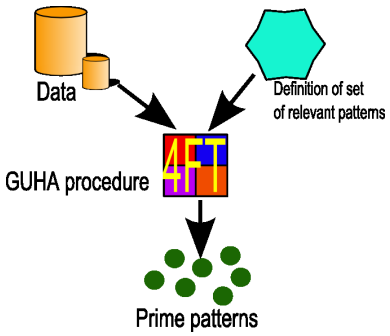


Fig. 1. The GUHA method

GUHA method is realized by GUHA procedures such as the 4FT procedure, located in the middle of the figure. A GUHA task consists of data and a simple definition of a possibly large set of relevant patterns defined with the

aid of observational calculi, which are inputs to the procedure. The procedure automatically generates all the relevant patterns and verifies them against the provided data. Patterns that are true and do not logically follow from the other true output and more simple patterns are called *prime patterns*. We call them also *hypotheses* as in [3]. The most known GUHA procedure is the ASSOC (*4ft-miner*, 4FT) procedure for mining generalized association rules [11, 12], based on different approach than the mainstream *apriori* algorithm [1].

In its initial form, procedures of the GUHA method were designed to mine over one relation only. In [10], proper theory for the multirelational form of the method was developed. However, until recently this form lacked suitable implementation and data to prove usability. We present in this paper recent implementation of the multirelational GUHA in the Ferda system [5] and we also start experiments with genetic data where the method seems to be perspective.

The paper is structured as follows: section 2 explains main principles of multirelational GUHA and briefly introduces the new implementation of the method. Section 3 compares our method to other mainly ILP methods and section 4 describes the initial experiments with genetic data. Finally section 5 concludes the paper.

2 Principles of Multirelational Mining with GUHA

Because of the short format of the paper, we explain only basics of the principles without going into detail. For more details, see [10, 4]. The multirelational GUHA method currently supports star-scheme of the database with one *master table* and several *detail tables*. The key term is *virtual attribute*, which is attribute from detail data table, that is created during the process of GUHA pattern verification and is treated as normal attribute of master table although not physically stored. The most interesting type of virtual attribute is the *hypotheses attribute*. Hypotheses attribute is defined by the GUHA (sub)task on the detail table. Value of the attribute corresponds to validity of a GUHA pattern for subset of records of the detail table that "belongs" to the master record. Validity can be expressed as boolean value or (in the future) generally as a real number. One GUHA (sub)task in on the detail table usually generates large amount of hypotheses attributes.

We present an example of the hypotheses attribute from the banking domain[1]. The used GUHA procedure is 4FT for association rules mining: there are two tables concerning clients of a bank. The master table contains information about clients' accounts and the detail table contains information about transactions of individual clients. One client in the master table can have several transactions in the detail table. Example of hypotheses attribute can be *client that often pays by credit card*, which can be formally written as

$$ClientID \approx Payment(CreditCard).$$

[1] We think that banking domain is more comprehensible than the genetic domain to the non-expert.

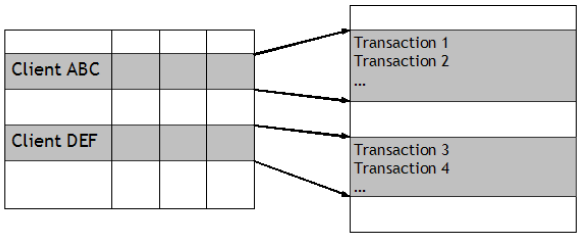


Fig. 2. Example of multirelational task setting

The situation is shown in figure 2. It is obvious, that this rule may be very useful as an attribute in the master table concerning clients' accounts. We name the virtual attribute $ClientPayingByCreditCard$. Then one can examine status of a client based on client's payments and address. Example of such generalized association rule is

$$District(SouthEast)\&ClientPayingByCreditCard$$

$$\approx Status(good).$$

There are two experimental implementation of relational GUHA method, one in the frame of LISp-Miner system [11] and the second one in the Rel-Miner system [4], but they are not used any more due to various reasons. The new implementation in the Ferda system [5] takes advantage of visual and modular environment, which makes the complex task setting comprehensible to the user. Figure 3 shows a sample multirelational task in Ferda. All the implementations of relational GUHA do not use *apriori*, they are based on representation of analyzed data by strings of bits [11].

3 Related Methods

The most known KDD technique for discovering knowledge from multirelational data is inductive logic programming (ILP). Principle of propositionalization approaches in ILP [6] is very close to principle of hypotheses attribute. Below is list of main differences:

- Propositioned attributes of ILP are conjunctions of (possibly negated) literals of predicate logic. In contrary, hypotheses attributes are formulas of observational calculus, enabling to represent i.e. implication or statistical significance of the attribute.
- In practical cases, multirelational GUHA is limited to star-scheme of the database. Relations in ILP does not have this restriction.

The *WARMR* algorithm [2] performs ILP propositionalization and then searches for association rules in *apriori*-like manner. Detailed comparison of *WARMR* and

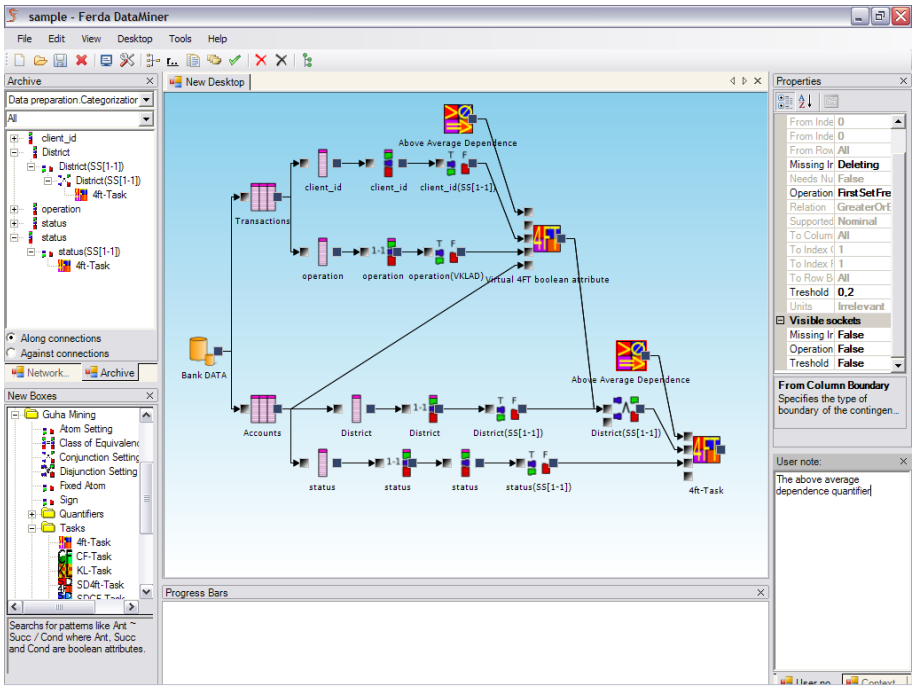


Fig. 3. The Ferda environment

multirelational GUHA can be found in [4]. Another approach based on *apriori* [9] adapts *support* and *confidence* measures and calculates them on multiple tables without joining them. The RELAGGS system [7] works in similar manner: it calculates aggregations of records of columns from tables bound by foreign keys. Because the method is based on database joins, it is not unable to calculate the expressive GUHA patterns on the detail tables.

4 Mining over Genetic Data

The big problem throughout the history of multirelational GUHA was to find suitable domain to prove the usefulness of the method. The first attempts were made in the banking domain - dataset Barbora was used[2]. These attempts were unsuccessful.

During this spring, we initiated cooperation with Czech Technical University (CTU) concerning mining genetic data. Team from CTU lead by F. Zelezny compiled genetic dataset from publicly available datasets. The dataset contains genetic measurements acquired from Affymetrix DNA microarrays[3] from hu-

[2] Used in the PKDD 1999 Discovery challenge, see `http://lisp.vse.cz/challenge`.

[3] National Center for Biotechnology Information GEO Datasets `http://www.ncbi.nlm.nih.gov`

man, mouse and rat for two different types of cells: hematopoetic and stromal, both of which are involved in blood cells production in bone marrow. The gene measurements were enriched with gene semantics involving information about pathways (maps representing molecular interaction and reaction networks) and fully-coupled-fluxes (FCF), linear pathway subgraphs[4].

The longterm scientific goal is to examine how does the expression of genes in FCF correlate with types of cells (and possibly other characteristics) [8]. Because the expressiveness of hypotheses attribute, multirelational GUHA is especially fit for this purpose. We have tried initial experiments with hypotheses attributes such as *high expression of genes in FCF* showing promising results: one of the experiments included examination of 500K gene measurements concerning 500 FCF's. With procedure 4FT we obtained 1394 *hypotheses* out of 3187 verifications. All the hypotheses were in form:

$$[FluxID(\alpha) \approx_1 GeneLevel(\beta)] \approx_1 CellType(\gamma)$$

Where $\approx_1$ stands for $Conf = 100\%$. However following work need to be done in order to obtain scientifically sound results:

- **Proper discretization:** GUHA has ways to handle numeric data. Yet these ways are unsuitable mainly because of the fact that expressions of different genes have different ranges, but are contained in one attribute. We need to build a new genome expression table based on discrete values directly from Affymetrics DNA microarrays.
- **Scaling:** It is the first time that multirational GUHA has been used with data of such a large size and we have experienced performance problems. Effective ways to handle results of queries, which by far exceed the capacity of operating memory need to by found and implemented.
- **Chip handling:** The probes measuring one gene are placed on several chips. It remains an open question how much does this fact influence the gene measurement.

5 Conclusion

We present multirelational extension of the GUHA method of exploratory analysis. Like other methods such as ILP propositionalization, the principle is to enrich a data table with attributes taken from other data tables. The advantage of multirelational GUHA lies in providing an expressive language for virtual attributes based on observational calculus.

We also present recent implementation of multirelational GUHA method in the Ferda system and possible and promising usage of the method in genetic experiments. At present, we do not yet have any scientifically sound genetic results, the paper states next steps to be made in order to achieve them. Nonetheless, usage of multirelational GUHA seems to be suitable for exploratory analysis of complex data over multiple tables such as genetic data.

[4] Taken from KEGG genome database, `http://www.genome.jp.kegg`

Acknowledgements

This work was supported by the project MSM6138439910 of the Ministry of Education of the Czech Republic and grant 201/08/0802 of the Czech Science Foundation. We thank and acknowledge contribution of our research colleagues Matěj Holec, Filip Železný and Jiří Kléma from Czech Technical University for providing the genetic data and guidance in the genetic domain.

References

1. Agrawal R., Mannila H., Srikant R., Toivonen H., Verkamo A.: *Fast discovery of association rules.* Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park (1996) p. 307 – 328
2. Dehaspe L., De Raedt L.: *Mining Association Rules in Multiple Relations.* In Proceedings of the 7th International Workshop on Inductive Logical Programming, Volume 1297, LNAI, pp. 125–132, Springer-Verlag, 1997
3. Hájek P., Havránek, T.: *Mechanising Hypothesis Formation - Mathematical Foundations for a General Theory.* Springer-Verlag: Berlin - Heidelberg - New York, 1978.
4. Karban T.: *Relational Data Mining and GUHA.* in Richta K., Snášel V., Pokorný J.(eds.): Proceedings of the 5th annual workshop DATESO 2005(Databases, Texts, Specifications and Objects), ISBN:80-01-03204-3, pp.103–112
5. Kováč M., Kuchař T., Kuzmin A., Ralbovský M.: *Ferda, New Visual Environment for Data Mining.* Znalosti 2006, Conference on Data Mining, Hradec Králové 2006, p. 118 – 129 (in Czech)
6. Kramer S., Lavrač N., Flach P.: *Proposionalization Approaches to Relational Data Mining.* In: Džeroski, Lavrač: Relational Data Mining, ISBN 3-540-42289-7, Springer Verlag 1998, pp. 262–291
7. Krogel M.-A., Rawles S., Železný F., Flach P.A., Lavrač N., Wrobel S.: *Comparative Evaluation of Approaches to Propositionalization* In: Horváth T., Yamamoto A. (Eds.) Proceedings of the 13th International Conference on Inductive Logic Programming. LNCS 2835, Springer-Verlag, 2003
8. Notebaar R.A., Teusink B., Siezen R.J., Papp B.: *Co-Regulation of Metabolic Genes is Better Explained by Flux Than Network Distance.* PLoS Computational Biology 4(1), 2008: e26 doi:10.1371/journal.pcbi.0040026
9. Pizzi, L.C., Ribeiro, M.X., Vieira, M.T.P.: *Analysis of Hepatitis Dataset using Multirelational Association Rules.* ECML/PKDD Discovery Challenge, 2005.
10. Rauch J.: *Many Sorted Observational Calculi for Multi-Relational Data Mining.* In: Data Mining Workshops. Piscataway: IEEE Computer Society, 2006 ISBN 0-7695-2702-7 p. 417–422
11. Rauch J., Šimůnek, M.: *An Alternative Approach to Mining Association Rules* Lin T Y, Ohsuga S, Liau C J, and Tsumoto S (eds): Foundations of Data Mining and Knowledge Discovery, Springer-Verlag, 2005 p. 219 – 239
12. Ralbovský M., Kuchař T.: *Using Disjunctions in Association Mining.* In: Perner P.: Advances in Data Mining - Theoretical Aspects and Applications, LNAI 4597, Springer Verlag, Heidelberg 2007

On and Off-Policy Relational Reinforcement Learning

Christophe Rodrigues, Pierre Gérard, and Céline Rouveirol

LIPN, UMR CNRS 7030, Institut Galilée - Université Paris-Nord
first.last@lipn.univ-paris13.fr

Abstract. In this paper, we propose adaptations of Sarsa and regular Q-learning to the relational case, by using an incremental relational function approximator RIB. In the experimental study, we highlight how changing the RL algorithms impacts generalization in relational regression.

1 Introduction

Most works on Reinforcement Learning (RL, [1]) use propositional – feature based – representations to produce approximations of value-functions. If states and actions are represented by scalar vectors, the classical numerical approach to learn value-functions is to use regression algorithms. Recently, the field of Relational Reinforcement Learning (RRL) has emerged [2] aiming at extending Reinforcement learning to handle more complex – first order logic-based – representations for states and actions. Moving to a more complex language opens up possibilities beyond the reach of attribute-value learning systems, mainly thanks to the detection and exploitation of structural regularities in (state, action) pairs.

In this paper, we study how even slight modifications in the RL algorithm employed may impact significantly on the performance of the relational regression system. In section 2, we briefly present the RL problem in the relational framework and we present three very similar RRL algorithms: the former Q-RRL [2], and two regular algorithms upgraded to relational representations: Q-learning (off-policy) and Sarsa (on-policy). We combine all these RL techniques with the same relational function approximator: RIB [3]. In section 3, we compare those algorithms experimentally and show a significant impact on RIB performance. Indeed, the size of the models learned by RIB decrease, resulting in an overall reduction of computation time.

2 Relational Temporal Difference

Relational Reinforcement Learning (RRL) addresses the development of RL algorithms operating on relational representations of states and actions.

The relational Reinforcement Learning task can be defined as follows. Given:

- a set of possible states $\mathcal{S}$, represented in a relational format,
- a set of possible actions $\mathcal{A}$, also represented in a relational format,

- an unknown transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, (this function can be nondeterministic)
- an unknown real-valued reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$,

the goal is to learn a policy for selecting actions $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ from any time step t. This return is the cumulative reward obtained in the future, starting in state s_t. Future rewards are weakened by using a discount factor $\gamma \in [0, 1]$. In value-based RL methods, the return is usually approximated thanks to a value function $V : \mathcal{S} \to \mathbb{R}$ or a Q-value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that $Q(s, a) \approx E\{R_t \mid s_t = s, a_t = a\}$.

Algorithm 1 Off-policy TD RRL algorithm: Qlearning-RIB (RIB-Q)

Require: state and action spaces $\langle \mathcal{S}, \mathcal{A} \rangle$, RIB regression system for Q_{RIB}
Ensure: approximation of the action-value function Q_{RIB}
  initialize Q
  loop
    choose randomly start state s for episode
    repeat
      $a \leftarrow \pi^{\tau}_{Q_{RIB}}(s)$ (Boltzmann softmax)
      perform a; get r and s' in return
      if s' is NOT terminal **then**
        $Q_{RIB}(s, a) \xleftarrow{learn} r + \gamma \max_{a' \in \mathcal{A}(s')} Q_{RIB}(s', a')$
      else
        $Q_{RIB}(s, a) \xleftarrow{learn} r$
      end if
      $s \leftarrow s'$
    until s terminal
  end loop

States are relational interpretations, as used in the "learning from interpretations" setting [4]. In this notation, each (state, action) pair is represented by a relational interpretation, *ie* a set of relational facts. The action is represented by an additional ground fact.

Among other incremental relational function approximators used in RRL [2, 5–7], the RIB system [3] is a quite good performance/efficiency compromise. It adopts an instance based learning paradigm to approximate the Q-value function. RIB stores a number of prototypes each associated with a Q-value. These prototypes are employed to predict the Q-value of unseen examples, using a k-nearest-neighbor algorithm. It takes advantage of a relational distance adapted to the problem to solve (see [8] for a distance for the blocks world problem). RIB handles incrementality since it forgets prototypes that are not necessary to reach a good prediction performance or have a bad prediction performance.

As opposed to regular Q-learning, in the RRL algorithm introduced in [2], learning occurs only at the end of episodes, and not at each time step. It stores full trajectories s_0, a_0, r_1, s_1, $\cdots$, s_{T-1}, a_{T-1}, r_T, s_T. Then, back-propagation

of all the time-steps occurs at once only when reaching a terminal state, using the usual update rule:

$$Q(s_t, a_t) \xleftarrow{learn} r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} Q_(s_{t+1}, a)$$

In order to learn at each time step, we propose (algorithm 1) a regular adaptation of Q-learning to a relational framework and use RIB for the relational regression part.

Algorithm 2 On-policy TD RRL algorithm: Sarsa-RIB (RIB-S)

Require: state and action spaces $\langle \mathcal{S}, \mathcal{A} \rangle$, RIB regression system for Q_{RIB}
Ensure: approximation of the action-value function Q_{RIB}
  initialize Q
  loop
    choose randomly start state s for episode
    $a \leftarrow \pi^{\tau}_{Q_{RIB}}(s)$ (Boltzmann softmax)
    repeat
      perform a; get r and s' in return
      $a' \leftarrow \pi^{\tau}_{Q_{RIB}}(s')$ (Boltzmann softmax)
      if s' is NOT terminal **then**
        $Q_{RIB}(s, a) \xleftarrow{learn} r + \gamma Q_{RIB}(s', a')$
      else
        $Q_{RIB}(s, a) \xleftarrow{learn} r$
      end if
      $s \leftarrow s'$; $a \leftarrow a'$
    until s terminal
  end loop

In this algorithm, $Q_{RIB}(s, a)$ stands for the RIB prediction for the (s, a) pair. $\pi^{\tau}_{Q_{RIB}}$ means that the action is chosen according to a policy π derived from the action values Q_{RIB}. The action is selected according to a Boltzmann distribution with a temperature τ [1].

Q-learning is said off-policy because it learns an optimal Q-value function, even if it does not always choose optimal actions. With minor modifications, we propose (algorithm 2) an upgraded version of Sarsa [1], an on-policy algorithm which learns the Q-value function corresponding to the policy it actually follows.

With these new algorithms, there is no need anymore to keep complete trajectories in memory. In addition, the value function is modified at each time step. As a consequence, action selection improves along an episode. Although we expect little performance gain from a strict RL perspective, from an ILP point of view, these algorithms take full advantage of the incrementality of RIB. This method changes both the presented samples and their order of presentation to

[1] The probability of choosing action a in state s is $\frac{e^{\frac{Q_{RIB}(a)}{\tau}}}{\sum_{b \in \mathcal{A}(s)} e^{\frac{Q_{RIB}(b)}{\tau}}}$.

the regression algorithm, resulting in a different generalization of the Q-value function.

3 Experimental study

The experiments are performed on the blocks world problem as described in [9]. Each algorithm (RIB-S, RIB-Q and RIB-RL) is tested for 20 trials, and results are averaged. The trials are divided into episodes, each starting in a random state and ending depending on the task to solve (*stacking* or *on(a,b)*). The Q-value function is periodically evaluated during a trial. For each evaluation, 10 episodes of greedy exploitation without learning are performed, each starting randomly.

In every experiment, the discount factor γ is set to 0.9. Each episode is interrupted after N time steps, depending of the number of blocks in the environment: $N = (n_{blocks} - 1) \times 3$. All other parameters are set according to [9].

Figure 1 compares the different algorithms facing the *on(a,b)* problem with 5 and 7 blocks. Table 1 shows the average number of instances used by RIB after 1000 episodes, indicating the complexity of the model obtained by each algorithm by each algorithm on the different problems.

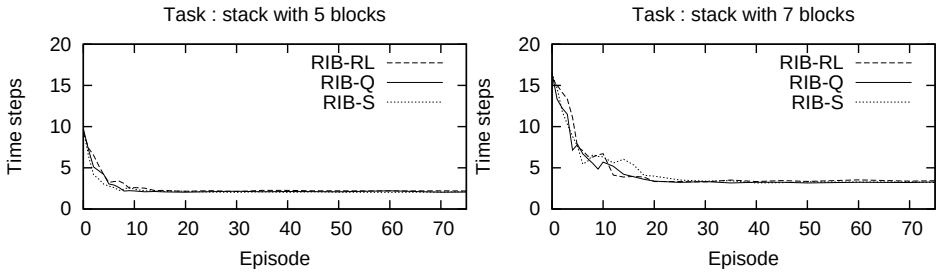


Fig. 1. Evolution of the number of time steps to complete an episode

stacking	5 blocks		6 blocks		7 blocks	
goal	Average	σ	Average	σ	Average	σ
RIB-RL	21	0	38	0	63	1.0
RIB-Q	19	0.2	32	0.6	51	1.9
RIB-S	19	0	32	0.5	50	1.3

Table 1. Number of prototypes used by RIB after 1000 episodes

Figure 2 compares the different algorithms facing the *stacking* problem with 5 and 7 blocks. Table 2 shows the average number of instances used by RIB after 1000 episodes (with standard deviation), indicating the complexity of the model obtained by each algorithm on the different problems.

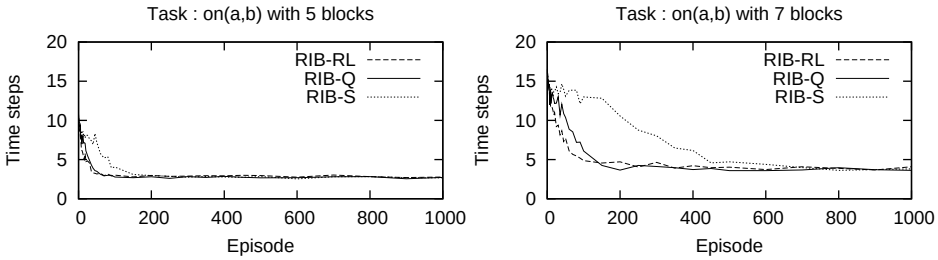


Fig. 2. Evolution of the number of time steps to complete an episode

$on(a,b)$	5 blocks		6 blocks		7 blocks	
goal	Average	σ	Average	σ	Average	σ
RIB-RL	325	3.0	757	18.4	1339	47.7
RIB-Q	254	2.9	566	7.8	1063	18.2
RIB-S	245	4.4	465	17.3	608	17.1

Table 2. Number of prototypes used by RIB after 1000 episodes

The experimental results show that, as one might expect on such a task where exploration actions do not lead to catastrophic actions, the off-policy TD algorithms (RIB-RL and RIB-Q) outperform slightly the on-policy one (RIB-S). Moreover, RIB-Q does not differ that much from the original Q-RRL (here, RIB-RL), considering the convergence speed wrt the number of episodes. Most important is the level of performance (computation time) reached by all presented RRL algorithms. Our adaptations of Q-learning and Sarsa, namely RIB-Q and RIB-S, don't provide more examples to the regression system than the former Q-RRL. Thus, since RIB's learning is linear in the number of prototypes, and having less prototypes saves computation time. RIB-S (on-policy) learns less prototypes for these relatively simple tasks, it explores a smaller portion of the state space than off-policy algorithms due to its policy, and therefore needs less prototypes to reach a good predictive accuracy on those states. As a consequence, RIB-S outperforms RIB-Q and RIB-RL as far as computation time is concerned, demonstrating than despite its slower convergence speed wrt the number of episodes, Sarsa remains a good candidate for scaling-up.

RIB is instance-based and thus strongly relies on its distance: generalization only takes place through the distance computation during the k-nearest-neighbor prediction. The distance used in RIB [9] is well suited for blocks world problems: it relies on a distance similar to the Levenshtein edit distance between sets of block stacks, seen as strings. The distance between (state, action) pairs is equal to 0 for pairs differing only by a permutation of constants that do not occur in the action literal. This distance also takes into account bindings of variables occurring in the goal. Without this prior knowledge, the system cannot solve

problems like $on(a, b)$, where two specific blocks have to be stacked on each other.

4 Conclusion

We have observed that even small differences in the RL techniques significantly influence the behavior of a fixed relational regression algorithm, namely RIB. We have proposed two RRL algorithms, RIB-Q and RIB-S, and have tested them on usual RRL benchmarks, showing performance improvements.

This work opens up several new research directions. We plan to adapt more sophisticated RL algorithms that will provide more useful information to the relational regression algorithms. We have already made experiments with relational TD(λ) with eligility traces, without noticing a significant improvement, neither on the number of prototypes nor on the computation time. A possible explanation is that the distance is to well fitted to the problem that eligibility traces are useless in that case. It might be interesting to study how RL may balance the effects of a misleading distance or even further, how RL may help in adapting the distance to the problem at hand.

Acknowledgements The authors would like to thank Kurt Driessens and Jan Ramon for very nicely and helpfully providing the authors with RIB-RL.

References

1. Sutton, R.S., Barto, A.: Reinforcement Learning: An Introduction. MIT Press (1998)
2. Dzeroski, S., De Raedt, L., Driessens, K.: Relational reinforcement learning. Machine Learning **43** (2001) 7–52
3. Driessens, K., Ramon, J.: Relational instance based regression for relational reinforcement learning. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003). (2003) 123–130
4. De Raedt, L., Džeroski, S.: First-order jk-clausal theories are PAC-learnable. Artificial Intelligence **70**(1–2) (1994) 375–392
5. Driessens, K., Ramon, J., , Blockeel, H.: Speeding up relational reinforcement learning through the use of an incremental first order decision tree algorithm. In: Proceedings of the European Conference on Machine Learning (ECML 2001), LNAI vol 2167. (2001) 97–108
6. Driessens, K., Dzeroski, S.: Combining model-based and instance-based learning for first order regression. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005). (2005) 193–200
7. Gartner, T., Driessens, K., , Ramon, J.: Graph kernels and gaussian processes for relational reinforcement learning. In: Proceedings of the 13th Inductive Logic Programming International Conference (ILP 2003), LNCS vol 2835. (2003) 146–163
8. Ramon, J., Bruynooghe, M.: A polynomial time computable metric between point sets. Acta Informatica **37**(10) (2001) 765–780
9. Driessens, K.: Relational reinforcement learning. PhD thesis, K. U. Leuven (2004)

Learning Complex Ontology Alignments – A Challenge for ILP Research

Heiner Stuckenschmidt, Livia Predoiu, Christian Meilicke

Computer Science Institute
University Mannheim, Germany
{heiner,livia,christian}@informatik.uni-mannheim.de

Abstract. In this paper, we propose the task of learning complex logical mappings between ontologies as a challenging task for ILP research. We motivate the need for complex ontology mappings using an example, formally define the task of learning complex mappings and identify a number of challenges for research on this issue in terms of tractability and uncertainty handling.

1 Background: Ontology Alignment

The integration of information from heterogeneous sources is one of the major challenges of modern information technology. Researchers from different areas including databases, knowledge representation and more recently in semantic web technologies have addressed this problem. Ontologies have been identified as a key technology for resolving semantic heterogeneity by providing common terms as well as formal specifications of their intended meaning in some logic. In large distributed environments with a high number of different information sources, however, it is unlikely that people will agree on a single ontology as the basis for integrating information. Here, we often face a situation where multiple ontologies describing the very same domain co-exist. In such a situation, we first have to integrate the different ontologies before they can serve as a basis for integrating information.

A common way of integrating different ontologies describing the same or largely overlapping domains is to use formal representations of semantic correspondences between their concepts and relations - also referred to as 'ontology mappings'. Manual approaches for identifying semantic correspondences are often not feasible since real world ontologies, for example in the medical domain, often contain several thousand concepts. As a response to this problem, a number of automatic and semi-automatic tools for generating hypotheses about semantic correspondences have been developed (see [4] for an overview). A serious limitation of almost all existing tools is the inability to identify complex mappings. In particular, most systems are only able to identify simple equivalence statements between class or relation names. The true semantic relation between elements of different ontologies, however, is often more complex.

In the following section, we give an example that illustrates the need for complex mappings and the limitations of existing systems. We propose the automatic identification of complex mappings between ontologies as an interesting and relevant challenge for the ILP community. The problem is of great practical importance as ontology

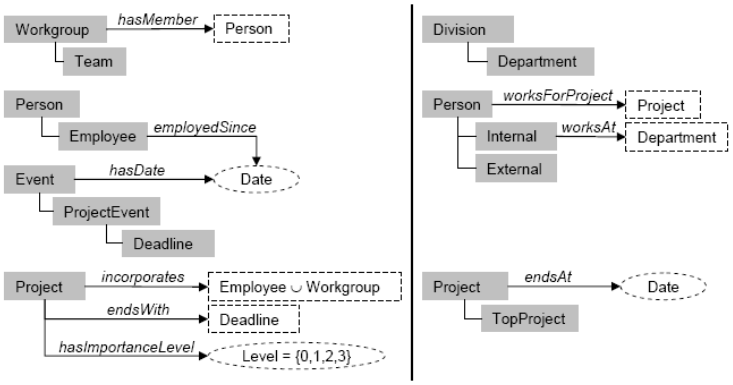


Fig. 1. An example of two ontology fragments describing employees and projects of a company. A labeled square represents a concept, a labeled ellipse a datatype, and a labeled arrow a role. The subsumption hierarchy of concepts is represented by indentation. Domain and range of a property are restricted to be the concepts connected by the accordant arrow.

matching is the Achilles heal of important research areas with a high potential impact, in particular the semantic web and enterprise application integration. Further, as we will argue below, the use of ILP as a paradigm for addressing the problem is a natural fit as the goal is to learn complex logical rules based on instances and background knowledge.

2 An Example Scenario

In the following example we focus on two ontologies describing human resources, projects and related topics. These ontologies are presented in figure 1. We refer to the ontologies as $\mathcal{O}_1$ (left side of the figure) and $\mathcal{O}_2$ (right side), and we use prefix $i\#$ to refer to the entities of $\mathcal{O}_i$. While both ontologies share some essential concepts, they differ especially with respect to the relations expressed via the properties. In particular these differences make the alignment process erroneous and require complex correspondences to express the correct semantic relations.

To better understand the capabilities of todays state of the art matching systems, we aligned these two ontologies with the Falcon-AO matching system [8], one of the top matching systems participating at the ontology alignment evaluation 2006 and 2007 [3]. As a result Falcon-AO generates two correspondences, namely $1\#Project(x) \leftrightarrow 2\#Project(x)$ and $1\#Person(x) \leftrightarrow 2\#Person(x)$. Are these correspondences sufficient to express the semantic relations that we might be interested in? Suppose that we would like to transfer instance data from $\mathcal{O}_1$ to $\mathcal{O}_2$. Which projects in $\mathcal{O}_1$ have to be classified as top projects in $\mathcal{O}_2$? These are projects with a high level of importance. We could for example use the following rule for migrating these projects to $\mathcal{O}_2$.

$$2\#TopProject(x) \leftarrow 1\#Project(x) \wedge 1\#hasImportanceLevel(x, 3) \qquad (1)$$

What about the deadline of a project? This relation is modeled via a single datatype property in $\mathcal{O}_2$ while we find a chain of properties in $\mathcal{O}_1$. Rule (2) represents this dependency.

$$2\#endsAt(x, z) \leftarrow 1\#endsWith(x, y) \wedge 1\#hasDate(y, z) \tag{2}$$

When we like to know which person are working in which projects, things are getting even more complicated, because the *1#incorporates* property relates both employees and workgroups to projects. We have to use the two rules to cope with the different modeling.

$$2\#worksForProject(x, z) \leftarrow 1\#incorporates(x, z) \wedge 1\#Employee(z) \tag{3}$$
$$2\#worksForProject(x, z) \leftarrow 1\#incorporates(x, y) \wedge 1\#hasMember(y, z) \tag{4}$$

We conclude that ontology alignment requires the use of complex and non-trivial correspondences. Otherwise the completeness of the alignment cannot be guaranteed and the semantic gap between different ontologies cannot be bridged in an appropriate way.

3 Formalization of the Problem

Based on [4] we can formalize the ontology matching problem as follows: Without loss of generality, we consider the case where we have two first-order theories or ontologies $\mathcal{O}_1 = T_1 \cup A_1$ and $\mathcal{O}_2 = T_2 \cup A_2$ given. The T component of an ontology determines the terminological knowledge definition and the A component determines the association of instances with predicates. Each of the ontologies $\mathcal{O}_i$ is represented in the language L_i and each of the ontologies has elements that can be defined by means of elements in the other ontology. Those elements are called the set of matchable elements $Q(\mathcal{O})$ of the ontology $\mathcal{O}$. Note that the set of matchable elements of an ontology depends of the other ontology that is involved in the matching process. The task of ontology matching is now to find correspondences between matchable elements in the two ontologies. Correspondences are 4-tuples $\{e_1, e_2, r, n\}$ such that

- n, a number between 0 and 1, expresses the degree of confidence in the correspondence.
- r is a relation between e_1 and e_2. We only consider implication between formulae as well as statements of the form $e_1 = e_2$ where e_1 and e_2 are constants. Statements of the latter form are called instance equivalences.
- each e_i is a formula represented in L_i. In the spirit of [4], we distinguish between three levels of expressivity. Given a level 0 correspondence, the formulae e_i consist simply of a single predicate. Level 1 corresponds to conjunctions of predicates on the right hand side of the implication relation $\leftarrow$ while the left hand side remains a single predicate. The final level 3 corresponds to arbitrary expressions in the languages L_i.

Based on these definitions, we can now more precisely define the learning task associated with the creation of complex ontology mappings as the ones described in the example above.

Definition 1 (Learning Task). *Given ontologies $\mathcal{O}_1 = T_1 \cup A_1$ and $\mathcal{O}_2 = T_2 \cup A_2$ in languages L_1 and L_2. Further given a mapping $\mathcal{M}$ that defines implications between predicates in T_1 and T_2 as well as equalities between constants in A_1 and A_2. Then the complex mapping learning problem is to find a set H of first order sentences such that:*

1. *Elements of H are of the form: $e_2 \leftarrow e_1$ where e_1 and e_2 are defined as above*
2. $(T_1 \cup T_2 \cup \mathcal{M}) \wedge A_1 \wedge H \models A_2$
3. $(\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}) \wedge H \not\models \bot$

From the point of view of Knowledge-Based Inductive Learning, the terminological part of the aligned ontologies together with the level 0 mappings and the instance equivalences play the role of background knowledge, while the instance information is used as training examples. More precisely, the hypothesis should explain the occurrence of instances in terms of A_2 based on their occurrence in A_1. Suppose for example that in A_2 we have instance *1#p-1762* with *1#Project*(*1#p-1762*) and *1#hasImportanceLevel*(*1#p-1762*, 3). If H contains equation 1 we can conclude that *2#TopProject*(*1#p-1762*). Further suppose that $\mathcal{M}$ contains instance equivalence *1#p-1762* = *2#P1762*. Finally, we can conclude that *2#TopProject*(*2#P1762*) and thus give a (partial) explanation of A_2. Further, we claim that the overall model consisting of the two ontologies, pre-existing and learned mappings is consistent to avoid solutions that trivially satisfy the second condition in the definition.

4 Problems and Challenges

The definition above seems to suggest that learning complex ontology mappings is quite straightforward as it can be phrased as a standard ILP learning problem. A closer look reveals, however, that there are a number of practical problems that make the task a challenging one. In this section we discuss three of these problems that we consider to be central to the endeavor and could be starting points for research in this area.

Tractability Work in inductive logic programming often focusses on supporting efficient subsets of first-order logic. In particular, there is a focus on first-order horn rules as a target language for learning and encoding background knowledge. In the context of ontologies, there is also some work concerned with languages that fall into this category (e.g. [6]). However, OWL-Lite and OWL-DL, the primary ontology languages, are based on expressive description logics $\mathcal{SHIF}(\mathcal{D})$ and $\mathcal{SHOIN}(\mathcal{D})$, respectively. It has been shown that disjunctive Datalog is needed to perform the kind of reasoning needed for testing the second condition of the definition [7]. Even worse, checking the consistency of the overall model cannot be done by a reduction to disjunctive Datalog but requires reasoning about a combined model consisting of description logic ontologies and rule-based mappings. It has been shown that reasoning for such models is highly intractable even for rather inexpressive ontology and rule languages. So far work in the ILP community has only addressed much weaker languages (e.g. [9, 10]. Recent work on a first major revision of the OWL language (OWL 2.0) addresses this problem and proposes an integration of ontologies and rules that can be reduced to a very expressive but still decidable description logic $\mathcal{SROIQ}$ [5]. While this enables us in

principle to test condition 3 in the definition, tractability is still a major issue claiming for highly optimized learning methods. While existing work on optimizing ILP seems to focus on the problem of dealing with large instance sets, learning ontology mappings comes with new challenges with respect to dealing with expressive models background knowledge that can also be very large - some ontologies contain tens of thousands of axioms.

Uncertainty The approach described above relies on the existence of an initial mapping between predicates and instances in the two ontologies. In order to determine these initial mappings an additional matching step is necessary (compare e.g. [12]). This problem which is referred to as entity and schema matching, respectively, is a research area in its own rights and a variety of methods have been proposed for this purpose. Most of these methods are based on weak criteria such as structural or linguistic similarity. As a result, the learning process has to cope with a significant degree of noise (recent papers report an F-Value of 70% to 90% for the instance matching task). For the problem of creating simple mappings as part of the background knowledge state of the art systems reach a performance of 60% to 90% on real world ontologies [3] which adds additional uncertainty into the learning process. This means that the development of highly robust learning methods is necessary to cope with the task. A possible way to go is to explicitly take the uncertainty introduced by entity and ontology into account [1]. As discussed above, mappings are annotated with a degree of confidence that can be interpreted in thee context of probabilistic ILP approaches [2]. This, however, requires that only probabilistic matching methods have been used to create simple mappings and entity correspondences. The majority of the existing approaches, however are based on different notions of similarity. Providing ways to exploit these similarities in the learning process would be a bit step forward.

Incompleteness and Inconsistency The third condition in our definition that claims the consistency of the overall model poses an additional and unexpected challenge to the learning task. We have shown that existing matching systems cannot guarantee that their result leads to a consistent model. This means that in many cases $(\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M})$ is already inconsistent. As the logical languages currently used for ontologies and mappings are monotonic, the third condition will never be satisfied in many cases. If we still want to learn complex mappings, we first have to fix the inconsistencies in the simple mappings. This is normally done by removing mappings from $\mathcal{M}$ that cause the overall model to become inconsistent. Simply removing all potential causes of inconsistency, however, will in cases remove too many mappings reducing the degree of overlap between the two ontologies which in turn can be expected to have a negative impact on the accuracy of the learning result. We therefore have to find a way to only remove the 'right' mappings in the sense that the set of removed mappings is minimal and contains only such mappings that are 'wrong' in the sense that their content does not correspond to reality. There are first results in this direction that apply techniques from model-based diagnoses to the debugging of inconsistent ontology mappings [11], but there is still a lot of space for improvement to get an optimal basis for learning complex mappings.

5 Summary and Conclusions

In this paper, we discussed the generation of complex ontology mappings as a challenging problem to be addressed by the ILP community. We think that this problem is interesting for ILP researchers because (1) the use of ILP for addressing this problem is a natural choice as the definition of the learning problem perfectly matches the ideas of ILP and because (2) a closer look reveals that the problem comes with some interesting challenges with respect to scalability and accuracy. In particular, ILP cannot be seen in isolation here, because the result of the learning phase is directly influenced by the quality of the instance and schema matching step. It is likely that there can be synergies between these two steps that have not been investigated so far, leaving space for interesting and challenging research on the border between ILP and semantic matching.

Acknowledgements: Research reported in this paper has been partially financed by the German Science Foundation (DFG) in the Emmy Noether Programme under contract STU 266/3-1.

References

1. A. Cali, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. Rule-based Approaches for Representing Probabilistic Ontology Mappings. In *Uncertainty Reasoning for the Semantic Web I*. Springer, to appear.
2. L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton. *Probabilistic Inductive Logic Programming - Theory and Applications*. Springer, 2008.
3. J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Svab, V. Svatek, W. van Hage, and M. Yatskevich. Results of the ontology alignment evaluation initiative 2007. In *Proc. of the ISWC 2007 Workshop on Ontology Matching*, Busan, Korea, November 2007.
4. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.
5. B. C. Grau, B. Motik, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. (W3C Working Draft 11 April 2008). Technical report, 2008.
6. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: combining logic programs with description logic. In *Proc. of the 12th international conference on World Wide Web (WWW03)*, 2003.
7. U. Hustadt, B. Motik, and U. Sattler. Reasoning in Description Logics by a Reduction to Disjunctive Datalog. In *Journal of Automated Reasoning (JAR)*, 2007.
8. N. Jian, W. Hu, G. Cheng, and Y. Qu. Falcon-AO: Aligning ontologies with falcon. In *K-CAP Workshop on Integrating Ontologies*, 2005.
9. J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, page 117132. Springer, 2003.
10. F. Lisi. Data mining in hybrid languages via ilp. In D. Calvanese, G. De Giacomo, and E. Franconi, editors, *Proc. of the 2003 International Workshop on Description Logics (DL'03)*, 2003.
11. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing ontology mappings. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, Vancouver, Canada, 2007.
12. H. Qin, D. Dou, and P. LePendu. Discovering executable semantic mappings between ontologies. In *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *LNCS*, pages 832–849, 2007.

A Simple Model for Sequences of Relational State Descriptions *

Ingo Thon, Niels Landwehr, and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. Artificial intelligence aims at developing agents that learn and act in complex environments. Realistic environments typically feature a variable number of objects, relations amongst them, and non-deterministic transition behavior. Standard probabilistic sequence models provide efficient inference and learning techniques, but typically cannot fully capture the relational complexity. On the other hand, statistical relational learning techniques are often too inefficient. In this paper, we present a simple model that occupies an intermediate position in this expressiveness/efficiency trade-off. Based on CP-logic, an expressive probabilistic logic for modeling causality. but specialized to represent a probability distribution over sequences of relational state descriptions, and employing a Markov assumption, inference and learning become more tractable and effective.

1 Introduction

One of the current challenges in artificial intelligence is the modeling of dynamic environments that change due to actions and activities people or other agents take. As one example, consider a model of the activities of a cognitively impaired person [1]. Such a model could be used to assist persons, using common patterns to generate reminders or detect potentially dangerous situations, and thus help to improve living conditions.

As another example and one on which we shall focus in this paper, consider a model of the environment in a *massively multi-player online game* (MMOG). These computer games support thousands of players in complex, persistent, and dynamic virtual worlds. They form an ideal and realistic test-bed for developing and evaluating artificial intelligence techniques. A model of human cooperative behavior can be useful in several ways. Analysis of in-game social networks are not only interesting from a sociological point of view but could also be used to give advice to inexperienced players. More ambitiously, the model could be used to build computer-controlled players that mimic the cooperative behavior of human players, form alliances and jointly pursue goals that would be impossible to attain otherwise. Mastering these social aspects of the game will be crucial to building smart and challenging computer-controlled opponents, which are currently lacking in most MMOGs. Finally, the model could also serve to detect non-human players in todays MMOGs — accounts which are played by automatic scripts to give one player an unfair advantage, and are typically against game rules.

* An extend version of this work was accepted at ECML 2008 [10].

From a machine learning perspective, this type of domain poses three main challenges: 1) world state descriptions are inherently relational, as the interaction between (groups of) agents is of central interest, 2) the transition behavior of the world will be strongly stochastic, and 3) a relatively large number of objects and relations is needed to build meaningful models, as the defining element of environments such as MMOGs are interactions among *large* sets of agents. Thus, we need an approach that is both computationally efficient and able to represent complex relational state descriptions and stochastic world dynamics.

Artificial intelligence has already contributed a rich variety of different modeling approaches, for instance, Markov models [2] and decision processes [3], dynamic Bayesian networks [4], STRIPS [5], statistical relational learning representations [6], etc. Most of the existing approaches that support reasoning about uncertainty (and satisfy requirement 2) employ essentially propositional representations (for instance, dynamic Bayesian networks, Markov models, etc.), and are not able to represent complex relational worlds, and hence, do not satisfy requirement 1). A class of models that integrates logical or relational representations with methods for reasoning about uncertainty (for instance, Markov Logic, CP-logic, or BLPs) is considered within statistical relational learning [6] and probabilistic inductive logic programming [7]. However, the inefficiency of inference and learning algorithms causes problems in many realistic applications, and hence, such methods do not satisfy requirement 3).

We aim to alleviate this situation, by contributing a novel representation, called CPT-L (for **CPT**ime-**L**ogic), that occupies an intermediate position in this expressiveness/efficiency trade-off. A CPT-L model essentially defines a probability distribution over sequences of interpretations. Interpretations are relational state descriptions that are typically used in planning and many other applications of artificial intelligence. CPT-L can be considered a variation of CP-logic [8], a recent expressive logic for modeling causality. By focusing on the sequential aspect and deliberately avoiding the complications that arise when dealing with hidden variables, CPT-L is more restricted, but also more efficient to use than its predecessor and alternative formalisms within the artificial intelligence and statistical relational learning literature.

2 CPT-L

We are interested in describing complex world states in terms of *relational interpretations*. A relational interpretation I is a set of ground facts $a_1, ..., a_N$. A *relational stochastic process* defines a distribution $P(I_0, ..., I_T)$ over sequences of interpretations of length T, and thereby completely characterizes the transition behavior of the world.

The semantics of CPT-L is based on CP-logic, a probabilistic first-order logic that defines probability distributions over interpretations [8]. CP-logic has a strong focus on causality and constructive processes: an interpretation is incrementally constructed by a process that adds facts which are probabilistic *outcomes* of other already given facts (the *causes*). CPT-L combines the semantics of CP-logic with that of (first-order) Markov processes. Causal influences only stretch from I_t to I_{t+1} (Markov assumption), are identical for all time-steps (stationarity), and all causes and outcomes are observable. Models in CPT-L are also called CP-theories, and can be formally defined as follows:

Definition 1. *A* **CPT-theory** *is a set of rules of the form*

$$r = (h_1 : p_1) \vee \ldots \vee (h_n : p_n) \leftarrow b_1, \ldots, b_m$$

where the h_i are logical atoms, the b_i are literals (i.e., atoms or their negation) and $p_i \in [0, 1]$ probabilities s.t. $\sum_{i=1}^{n} p_i = 1$.

It will be convenient to refer to $b_1, ..., b_m$ as the body $body(r)$ of the rule and to $(h_1 : p_1) \vee \ldots \vee (h_n : p_n)$ as the head $head(r)$ of the rule. We shall also assume that the rules are range-restricted, that is, that all variables appearing in the head of the rule also appear in its body. Rules define conditional probabilistic events: the intuition behind a rule is that whenever $b_1\theta, ..., b_m\theta$ holds for a substitution θ in the current state I_t, exactly one of the $h_i\theta$ in the head will hold in the next state I_{t+1}. In this way, the rule models a (probabilistic) causal process as the condition specified in the body causes one (probabilistically chosen) atom in the head to become true in the next time-step.

From a CPT-theory, the rule $(h_1 : p_1\theta) \vee \ldots \vee (h_n : p_n\theta) \leftarrow b_1\theta, \ldots, b_m\theta$ is obtained for a substitution θ. A ground rule r is applicable in I_t if and only if $body(r) = b_1\theta, \ldots, b_m\theta$ is true in I_t, denoted $I_t \models b_1\theta, \ldots, b_m\theta$.

One of the main features of CPT-theories is that they are easily extended to include *background knowledge*. The background knowledge B can be any logic program, cf. [9].

The set of all applicable ground rules in state I_t will be denoted as $\mathbf{R}_t$. Each ground rule applicable in I_t will cause one of its head elements to become true in I_{t+1}. More formally, let $\mathbf{R}_t = \{r_1, ..., r_k\}$. A *selection* σ is a mapping $\{(r_1, j_1), ..., (r_k, j_k)\}$ from ground rules r_i to indices j_i denoting that head element $h_{ij_i} \in head(r_i)$ is selected. In the stochastic process to be defined, I_{t+1} is a possible successor for the state I_t if and only if there is a selection σ such that $I_{t+1} = \{h_{1\sigma(1)}, ..., h_{k\sigma(k)}\}$. We shall say that σ *yields* I_{t+1} in I_t, denoted $I_t \xrightarrow{\sigma} I_{t+1}$, and define

$$P(I_{t+1}|I_t) = \sum_{\sigma : I_t \xrightarrow{\sigma} I_{t+1}} P(\sigma) = \sum_{\sigma : I_t \xrightarrow{\sigma} I_{t+1}} \Big(\prod_{(r_i, j_i) \in \sigma} p_{j_i} \Big) \tag{1}$$

As for propositional Markov processes, the probability of a sequence $I_0, ..., I_T$ given an initial state I_0 is defined by $P(I_0, ..., I_T) = P(I_0) \prod_{t=0}^{T} P(I_{t+1} \mid I_t)$. Intuitively, it is clear that this defines a distribution over all sequences of interpretations of length T much as in the propositional case. More formally:

Theorem 1 (Semantics of a CPT theory). *Given an initial state I_0, a CPT-theory defines a discrete-time stochastic process, and therefore for $T \in \mathbb{N}$ a distribution $P(I_0, ..., I_T)$ over sequences of interpretations of length T.*

3 Inference and Parameter Estimation in CPT-L

As for other probabilistic models, we can now ask several questions about the introduced CPT-L model:

– **Sampling**: how to sample sequences of interpretations $I_0, ..., I_T$ from a given CPT-theory $\mathcal{T}$ and initial interpretation I_0?

- **Inference**: given a CPT-theory $\mathcal{T}$ and a sequence of interpretations $I_0, ..., I_T$, what is $P(I_0, ..., I_T \mid \mathcal{T})$?
- **Parameter Estimation**: given the structure of a CPT-theory $\mathcal{T}$ and a set of sequences of interpretations, what are the maximum-likelihood parameters of $\mathcal{T}$?
- **Prediction**: Let $\mathcal{T}$ be a CPT-theory, $I_0, ..., I_t$ a sequence of interpretations, and F a first-order formula that constitutes a certain property of interest. What is the probability that F holds at time $t + d$, $P(I_{t+d} \models_B F \mid \mathcal{T}, I_0, ..., I_t)$?

The solution to this problems and the algorithms are described in detail in [10].

4 Experimental Evaluation

The proposed CPT-L model has been evaluated in two different domains. First, we discuss experiments in a stochastic version of the well-known *blocks world* domain, an artificial domain that allows to perform controlled and systematic experiments e.g. with regard to the scaling behavior of the proposed algorithms. Second, the model is evaluated on real-world data collected from a live server of a massively multi-player online strategy game. Experiments in these two domains will be presented in turn.

4.1 Experiments in a Stochastic Blocks World Domain

As an artificial test bed for CPT-L, we performed experiments in a stochastic version of the well-known *blocks world* domain. The domain was chosen because it is truly relational and also serves as a popular artificial world model in agent-based approaches such as planning and reinforcement learning. Application scenarios involving agents that act and learn in an environment are one of the main motivations for CPT-L.

In a first experiment, we explore the convergence behavior of the EM algorithm for CPT-L. The world model together with the policy for the agent, which specifies which block to stack next, is implemented by a (gold-standard) CPT-theory $\mathcal{T}$, and a training set of 20 sequences of length 50 each is sampled from $\mathcal{T}$. From this data, the parameters are re-learned using EM. Figure 1, left graph, shows the convergence behavior of the algorithm on the training data for different numbers of blocks in the domain, averaged over 15 runs. It shows rapid and reliable convergence. Figure 1, right graph, shows the running time of EM as a function of the number of blocks. The scaling behavior is roughly linear, indicating that the model scales well to reasonably large domains. Absolute running times are also low, with about 1 minute for an EM iteration in a world with 50 blocks[1].

4.2 Experiments in a Massively Multi-player Online Game

As an example for a massively multi-player online game, we consider *Travian*[2], a commercial, large-scale strategy game with a player community of about $3.000.000$ players

[1] All experiments were run on standard PC hardware, 2.4GHz Intel Core 2 Duo processor, 1GB memory.

[2] www.travian.com;www.traviangames.com

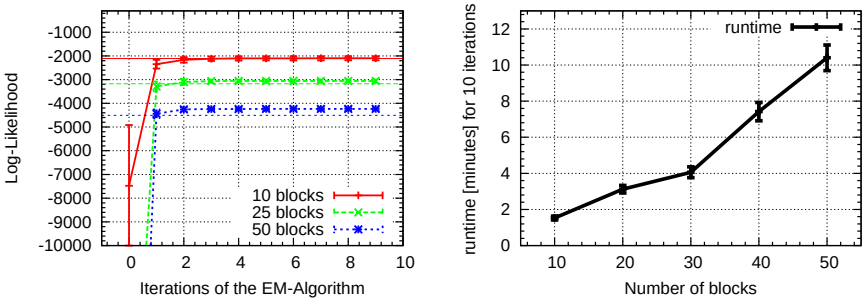


Fig. 1. Left graph: per-sequence log-likelihood on the training data as a function of the EM iteration. Right graph: Running time of EM as a function of the number of blocks in the world model.

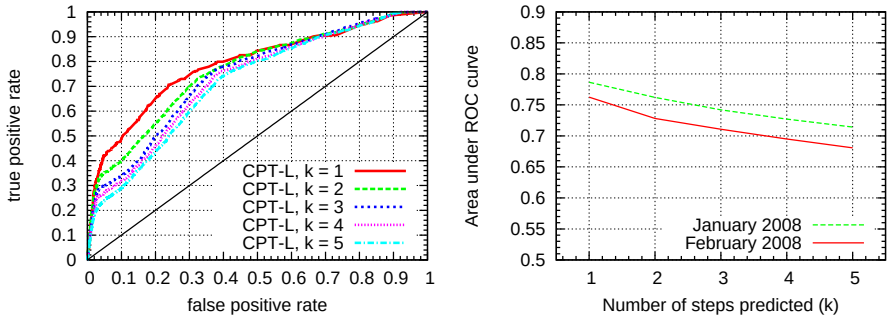


Fig. 2. Left figure: ROC curve for predicting that a city C will be conquered by a player P within the next k time-steps, for $k \in \{1, 2, 3, 4, 5\}$. The model was trained on 10 sequences of local game state descriptions from December 2007, and tested on 10 sequences from January 2008. Right figure: AUC as a function of the number k of future time-steps considered in the same experiment. Additionally, AUC as a function of k is shown for 10 test sequences from February 2008.

worldwide. In Travian, players are spread over several independent *game worlds*, with approximately 20.000–30.000 players interacting in a single world. Travian game play follows a classical strategy game setup. Players can build different military units which can be used to attack and conquer other cities on the map.

We are interested in the *dynamic* aspect of this world: as players are acting in the game environment (e.g. by conquering other players' cities and joining or leaving alliances), the game graph will continuously change, and thereby reflect changes in the social network structure of the game.

We consider the task of predicting the "conquest" action $conq(P, C)$ based on a learned generative model of world dynamics. The collected sequences of (local) game states were split into one training set (December 2007) and two test sets (January and February 2008). Maximum-likelihood parameters of a hand-crafted CPT-theory $\mathcal{T}$ as described above were learned on the training set using EM. The learned model was used to predict the player action $conq(P, C)$ on the test data in the following way. Let S denote a test sequence with states $I_0, ..., I_T$. For every $t_0 \in \{0, ..., T-1\}$, and every player p and city c occurring in S, the learned model is used to compute the probability that the conquest event $conq(p, c)$ will be observed in the next world state, $P(I_{t_0+1} \models conq(p, c) \mid \mathcal{T}, I_0, ..., I_{t_0})$. This probability is obtained from the

sampling-based prediction algorithm. The prediction is compared to the known ground truth (whether the conquest event occurred at that time in the game or not). Figure 2, left, shows ROC curves for this experiment with different values $k \in \{1, 2, 3, 4, 5\}$, evaluated on the first test set (January 2008). Figure 2, right, shows the corresponding AUC values as a function of k for both test sets. The achieved area under the ROC curve is substantially above 0.5 (random performance), indicating that the learned CPT-theory $\mathcal{T}$ indeed captures some characteristics of player behavior and obtains a reasonable ranking of player/city pairs (p/c) according to the probability that p will conquer c. In summary, we conclude that player actions in Travian are indeed to some degree predictable from the social context of the game, and CPT-L is able to learn such patterns from the data. Parameter learning for the CPT-L theory $\mathcal{T}$ on the training set takes approximately 30 minutes, and the model needed 5 iterations of EM to converge.

5 Conclusions and Future Work

We have introduced CPT-L, a probabilistic model for sequences of relational state descriptions. In contrast to other approaches that could be used as a model for such sequences, CPT-L focuses on computational efficiency rather than expressivity. This is essential for many real-life applications. The main direction for future work is to further evaluate the trade-off between representational power and scaling behavior in challenging real-world domains. Furthermore, we want to explore how the model can be extended, for instance to account for hidden data, without sacrificing efficiency.

References

1. Pollack, M.E.: Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. AI Magazine **26**(2) (2005) 9–24
2. Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2) (1989) 257–286
3. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc. (1994)
4. Ghahramani, Z.: Learning dynamic bayesian networks. In: Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks. (1997) 168–197
5. Fikes, R.E., Nilsson, N.J.: STRIPS: a new approach to the application of theorem proving to problem solving. In: Computation & intelligence: collected readings, Menlo Park, CA, USA, American Association for Artificial Intelligence (1995) 429–446
6. Getoor, L., Taskar, B., eds.: Statistical Relational Learning. MIT press (2007)
7. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S., eds.: Probabilistic Inductive Logic Programming - Theory and Applications. Volume 4911 of Lecture Notes in Computer Science., Springer (2008)
8. Vennekens, J., Denecker, M., Bruynooghe, M.: Representing causal information about a probabilistic process. In: Logics In Artificial Intelligence. Volume 4160 of Lecture Notes in Computer Science. (2006) 452–464
9. Bratko, I.: Prolog Programming for Artificial Intelligence. Addison-Wesley (1990) 2nd Edition.
10. Thon, I., Landwehr, N. adn De Raedt, L.: A simple model for sequences of relational state descriptions. In: Proceedings of the 19th European Conference on Machine Learning

Relational Data Mining in Crisis Management

Martin Večeřa and Luboš Popelínský

KD Lab at Faculty of Informatics
Masaryk University,
Brno, Czech Republic
{xvecera1,popel}@fi.muni.cz

Abstract. We describe a proposal of a new framework called CriMi to support spatio-temporal reasoning in crisis management. CriMi is based on PostGIS and the multi-relational learner GRAPE. Data on floods in the Czech Republic will be used for testing.

Key words: crisis management, geographic information system, spatio-temporal data, multi-relational data mining, frequent patterns, association rules.

1 Introduction

The aim of this work is to develop a technique for generating civil crisis management plans in different areas based on multi-relational data mining. This involves spatio-temporal data preprocessing, implementing a set of appropriate rules for ILP system, and integrating the whole process into local civil crisis management planning. To the best of our knowledge, to date, there is no comprehensive work on spatio-temporal multi-relational data mining for civil crisis management. Outcome of this work will be a proof of concept on floods data that could be used in local civil crisis management planning. Relational data mining (or inductive logic programming) [8] is the most powerful technology for mining multi-relational data. However, mining spatio-temporal data is more complex task than mining multi-relational data and the following facts need to be taken in account [2]:

1. implicit spatial and temporal relations
2. granularity of spatial objects
3. granularity of temporal intervals

The history of spatial data mining tools started in 1997 with *GeoMiner* [9] that also contained a module for association rules mining, *GeoAssociator*. However, the system suffered from single-table assumption. The very first work related to usage of ILP in spatio-temporal data mining described the *GWiM* system [11]. It was a proof of concept showing that the idea of *Horn clauses* can be very useful. *SPADA* (Spatial Pattern Discovery Algorithm) [10], a part of ARES [1], is another ILP data mining system which solves hierarchical structure problem. Most studies in association rule mining have focused on mining rules at single

concept levels. Either at the primitive level or at a rather high concept level. Yet many applications would benefit from concept hierarchies that are often available as part of the domain knowledge. Due to the evolution in the expressiveness of target languages, the discovery of multi-level association rules is one of those data mining problems to which ILP can supply an elegant solution. This is especially useful for spatio-temporal mining where different spatial hierarchies can be specified.

In the next section we describe PostGIS, a geographic extension of PostgreSQL database system. Section 3 brings a brief overview of GRAPE, a multi–relational system for mining first–order frequent patterns. Main features of CriMi are then described in Section 4. We conclude with information about test data in Section 5 with concluding remarks in Section 6.

2 PostGIS

PostGIS is an extension of PostgreSQL[1]. All the existing functions of PostgreSQL remain unchanged. In addition PostGIS enables to store various geometries. Supported objects are points, line strings, polygons, multi-points, multi-line strings, multi-polygons and a geometry collection (collection of any of the previous objects). For indexing this new type effectively PostGIS also adds new index type GiST (R-Tree).

There are many new operators for working with *GEOMETRY* data like same as (∼=), completely contains (∼), completely contained (@),overlaps (&&), strictly to the left/right (<<, >>) and so on. Also functions are added to the system, the most interesting of them include `ST_Distance`, `ST_Within`, `ST_Disjoint`, `ST_Intersects`, `ST_Touches`, `ST_Crosses`, `ST_Covers`, `ST_Relate`, `ST_Area`, `ST_Length`, `ST_Boundary`, `ST_Intersection`.

An example of PostGIS usage is shown in Figure 1. It presents an easy way of exporting geometry data into textual representation, and a SQL query that finds all pubs that are maximally 250 m far from any hospital. Many of the functions

```
SELECT id, name, AsText(geom) FROM body;
 id |     name      |          astext
----+---------------+--------------------
  1 | Point no. 1   | POINT(18.053 49.689)

SELECT h.name, p.name FROM hospitals h, pubs p
       WHERE Distance(h.the_geom, p.the_geom) < 250;
```

Fig. 1. PostGIS usage example

[1] PostgreSQL home page `http://www.postgresql.org`

are performed by *Geometry Engine Open Source*[2] (GEOS) which is strongly recommended to be installed together with PostGIS. Different data usually use different coordinate system. To solve the coordinate system incompatibilities, there is option to install *Proj.4 Cartographic Projections*[3] library. Every database table can then have its own coordinate system specified.

3 RAP and GRAPE

RAP is a system for searching long frequent patterns in dense data using different search strategies [6]. Input of RAP consists of three separate files that contain a learning set, a definition of domain knowledge and language specification. RAP offers rich set of constraints for search space. Except for basic frequent patterns, RAP can also find frequent patterns typical for particular classification class. This is often referenced as emerging patterns.

For use with spatio-temporal data there has been proposed extension to RAP called GRAPE [5] which implements new refinement operator. This refinement operator extends standard refinement operator by six other operators:

1. $\rho((P, is\_a(Attr, X_N), R)) = (P, is\_a(Attr, X_{N+1}), R)$ where X_N is a value in a level N in a hierarchy for the attribute $Attr$ and X_{N+1} is an ancestor of X_N in this hierarchy.
2. $\rho(P) = (P, \Diamond(X))$ where X is a new variable and there is no other temporal predicate in P with a free variable (i.e. unused in P).
3. $\rho((P, \Diamond(T), S)) = (P, \Box(T), S)$ if there is no term $T1\Theta$-equivalent (in terms of Θ-subsumption) with T in the rest of the pattern.
4. $\rho((P, \Diamond(T), S)) = (P, \bigcirc^n(T), S)$ if there is no term $T_1\Theta$-equivalent with T in the rest of the pattern.
5. $\rho(P) = (P, \Diamond(T_1))$ if P contains $\Box(T)$, where T, T_1 are terms, T_1 is a proper specialization of T, and T_1 does not appear elsewhere in the pattern.
6. $\rho(\star(X)) = \star(\rho(X))$ for $\star \in \Diamond, \Box, \bigcirc^k$.

GRAPE uses spatio-temporal language ST_0 introduced in [4]. This language combines RCC-8 with the propositional temporal logic (PTL). RCC-8 [7] consists of region variables $X_0, X_1, \ldots$ and the eight binary predicates: disconnected (DC), external contact (EC), equal (EQ), partially overlapping (PO), tangential part (TPP), inverse tangential part (TPPi), non-tangential part (NTPP), in-versed non-tangential part (NTPPi). A spatial formula is then constructed from the predicates above and the logical connectives $\vee, \wedge, \rightarrow, \neg$.

Time is assuming to be isomorphic with the set of natural numbers and the relation $<$ is defined together with two temporal operators *Since* and *Until*. When allowing application of *Since* and *Until*, or other standard operators like $\bigcirc$ (next), $\Diamond$ (sometimes) or $\Box$ (always) to spatial formulas we have got the spatio-temporal language ST_0.

[2] GEOS home page `http://trac.osgeo.org/geos/`
[3] Proj.4 home page `http://trac.osgeo.org/proj/`

4 CriMi: Mining in crisis management data

CriMi consists of three basic building blocks - PostgreSQL/PostGIS, Weka and GRAPE. Having spatio-temporal data in PostgreSQL we have to transformed them into the typed first-order predicate logic. We also need to extract language bias specification from the data and the database scheme. This process cannot be fully automated but the proposed system offers tools to minimize human intervention. Weka[4] is data preprocessing and data mining system that can be seamlessly connected to PostgreSQL and used for pre-processing the non-spatial part of data.

GRAPE, a system for mining in spatio-temporal data will be adapted for mining in crisis management data. Especially the refinement operator [5] implemented in GRAPE will be modified. The system will also incorporate different measures for finding the most interesting frequent patterns and association rules [3].

There should be a wizard for creating queries that offers PostGIS functions and shows results - predicates, types of arguments and language bias. It will be possible to generate predicates in ST_0 logic [4]. The expected number of tuples returned by the query should be displayed. For each predicate there should be a bitmap specifying presence of particular predicate in different hierarchical level. Hierarchical levels will also have specific support and confidence configured. The system is able to track changes in the source database and to offer an update of predicates. Using predefined meta-rules, CriMi can automatically generate predicates for any two tables according to the object types, e.g. road, railway, forest, city, district. This also results in minimizing the need of human intervention and in generating reasonable predicates. For cooperation with other tools for crisis management data a Web service interface will be as a part of CriMi.

There should be a possibility to use a kind of object cache for the data in a database, represent the data in native Java internal structures, and use Prolog only to verify hypotheses. Walking through the search space and refinement would be done in Java directly. Such a solution could bring a great performance speed up.

5 Test data

The test data come from the FLOREON (FLOods REcognition on the Net) system[5] which has been developed to bring the information about any possible approaching flood to the end user. The main purpose of the FLOREON system is to present those information to arbitrary type of users including citizens, majors, governments, and or specialists. Therefore, the system can provide information in required level of detail. Individual parts of the system are implemented as web services and thus easily reusable by other systems. As data warehouse there is used – among others – PostGIS. Meteorological module uses various data sources

[4] Weka home page `http://www.cs.waikato.ac.nz/ml/weka/`
[5] FLOREON home page`http://floreon.vsb.cz/web/`

including ALADIN (numerical weather forecasting system engaged by Czech Hydro-meteorologic Institute) and Medard (weather forecast system developed at Institute of Computer Science on Academy of Sciences of the Czech Republic). Other parts of the system are third party flood calculation modules, geographical module for providing terrain data, map server for combining and visualization of all the data, and web user interface. Data provided to our system are rainfall and river flow rates by measure sites, and the list of measure sites with coordinates. Results from our system will be accessible using web service for it to be easily incorporated into FLOREON.

6 Conclusions

The aim of this work is to design a system , implement for data analysis that would assist in building crisis management plans. The CriMi system that has been described consists of three parts - transformation raw data to first-order logic and preprocessing, learning first-order frequent patterns, and different use of (a subset) of those rules in analyzing FLOREON data.. CriMi is the first system designed especially for spatio-temporal multi-relational data mining for civil crisis management.

CriMi is primarily intended for a direct use by experts in data mining. In future we plan to extend the CriMi framework with an interface for non-experienced users.

Acknowledgment

This work has been partially supported by the Grant Agency of the Czech Republic under the Grant No. MSM0021622418 Dynamic Geovisualization in Crisis Management and by Faculty of Informatics, Masaryk University. VŠB – Technical University of Ostrava provided FLOREON system. Special thanks to Jan Blaťák for his valuable advices.

References

1. Annalisa Appice, Margherita Berardi, Michelangelo Ceci, and Donato Malerba. Mining and filtering multi-level spatial association rules with ARES. In *Foundations of Intelligent Systems*, volume Volume 3488/2005, pages 342–353. Springer Berlin / Heidelberg, 2005.
2. Annalisa Appice, Michelangelo Ceci, Antonietta Lanza, Francesca A. Lisi, and Donato Malerba. Discovery of spatial association rules in geo-referenced census data: A relational mining approach. *Intelligent Data Analysis*, 7(6):541–566, 2003.
3. Paulo J. Azevedo and Alípio Mário Jorge. Comparing rule measures for predictive association rules. In *Machine learning: ECML*, volume Volume 4701/2007, pages 510–517. Springer Berlin / Heidelberg, 2007.
4. Brandon Bennett, Anthony G. Cohn, Frank Wolter, and Michael Zakharyaschev. Multi-dimensional modal logic as a framework for spatio-temporal reasoning. *Applied Intelligence*, 17(3):239–251, 2002.

5. J. Blaťák and L. Popelínský. Toward mining of spatiotemporal maximal frequent patterns. In *Proceedings of the Workshop on Mining Spatio-Temporal Data at ECML/PKDD 2005*, pages 31–40, 2005.
6. J. Blaťák, L. Popelínský, and M. Nepil. RAP: Framework for mining frequent datalog patterns. In *Proceedings of the first KDID workshop at ECML/PKDD 2002*, pages 85–86, 2002.
7. A.G. Cohn, B. Bennett, J.M. Gooday, and N. Gotts. RCC: A calculus for region based qualitative spatial reasoning. *GeoInformatica*, (1):275–316, 1997.
8. Sašo Džeroski and Nada Lavrač, editors. *Relational Data Mining*. Springer Verlag, Berlin, September 2001.
9. Jaiwei Han, Krzysztof Koperski, and Nebojsa Stefanovic. GeoMiner: a system prototype for spatial data mining. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 553–556, 1997.
10. Francesca A. Lisi and Donato Malerba. Inducing multi-level association rules from multiple relations. *Machine Learning*, 55(2):175–210, May 2004.
11. Luboš Popelínský. Knowledge discovery in spatial data by means of ILP. In *PKDD '98: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 185–193, London, UK, 1998. Springer-Verlag.

A Sample Complexity for PILP

Hiroaki Watanabe and Stephen Muggleton

Department of Computing, Imperial College London
Email: {hw3, shm}@doc.ic.ac.uk

Abstract. Probabilistic inductive logic programming (PILP) is a probabilistic extension of ILP. We explore fundamental relationships between ILP and PILP by considering generality orders between their underlying representations first. This leads to an alternative view of PILP as a way of providing approximate generalisation. We study a sample complexity of PILP based on the approximate generalisation by extending PAC learning framework. To the authors' knowledge, this is a first attempt to characterise PILP in terms of computational complexity theory.

1 Introduction and Approximative Generalisation

The ability of inducing relational concepts on the basis of examples with background knowledge plays an important role in intelligent activities. The task becomes harder if the examples contains uncertainties. Probabilistic inductive logic programming (PILP) provides a framework for such a task. PILP studies Machine Learning algorithms for learning probabilistic relational concepts from noisy examples associated with background knowledge. In this paper we study a new sample complexity of a concept learning by introducing a probabilistic classifier[1] as a first step to construct computational learning theory of PILP.

In ILP[1], given sets of first-order logical clauses: hypotheses (H), background knowledge (BK), and positive examples (E), we consider the following entailment relation:

$$BK \cup H \models E \tag{1}$$

where H is a generalisation of E associated with BK. We propose an alternative approach by adding the following new three statements:

$$BK \cup E' \models E \tag{2}$$

$$BK \cup H' \models E' \tag{3}$$

$$BK \cup H' \models H \tag{4}$$

where H' is a set of hypotheses and E' is a set of examples. Intuitively (2) and (4) express that (a) E' is more general than E associated with BK and (b) H' is more general than H associated with BK respectively. The entailment relation between H' and E can be proved associated with BK as follows.

[1] A probabilistic classifier returns the results of the classification in probability values instead of in Boolean values.

Theorem 1.

$$BK \cup H' \models E$$

is held.

Proof. As a consequence of (4) and (1), $BK \cup H' \models H \cup BK \models E$. Alternatively, as a consequence of (2) and (3), $BK \cup H' \models E' \cup BK \models E$. □

Fig. 1 shows "*more general*" relations associated with BK defined by (1), (2), (3), and (4). For example, E' is more general than E and H' is more general than E' in the figure. The route (1) in Fig. 1 is the usual route in the normal ILP setting, however, the new route, (2)(3)(4), can also be used as an alternative. Surprisingly in (4) H is deductively obtained from H'. We view that the route (3) can be

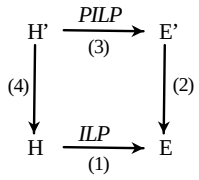


Fig. 1. ILP and PILP from Approximative Generalisation Point of View

associated with PILP[2] by expressing E' as probability-labelled examples, so called *probabilistic examples* in which a probability label attached to an example expresses the probability of the associated example being classified as positive. Note that in ILP (route (1)) Boolean-labelled examples are used for representing positive examples (*true*) and negative examples (*false*) instead of probability values. This approximative generalisation view of PILP is on the basis of E. That is, probability labels are required to be estimated using some statistical information whose source is in the generalisation from E to E'. In this paper, we assume that probability labels of probabilistic examples have already been estimated so that we can focus on the analysis of the sample complexity in the route (3). We introduce a variant of PAC learning [3, 5] framework in the next section.

2 Sample Complexity Analysis

2.1 Definitions

Let us assume that Learner obtains a set of m probabilistic examples, $E' = \{\langle x_1, \hat{p}_1\rangle, ..., \langle x_m, \hat{p}_m\rangle\}$ where $x_i \in X$ is an example and $\hat{p}_i$ is an *estimated* probability label being guaranteed to have an error at most ϵ in a confidence level $1 - \delta$. Note that ϵ and δ are small constants such that $0 \leq \epsilon \leq 1$ and $0 \leq \delta \leq 1$.

In PAC learning, Teacher and Learner classify the examples in Boolean values, however, in Approximative learning they classify in probability values.

Definition 1 (Classification in probability value). *A hypothesis h is said to be probabilistic classifier which takes a probabilistic example $\langle x_i, \hat{p}_i \rangle \in E'$ and returns the probability, $h(x_i)$, of x_i being accepted as true.*

Fig. 2 shows the two cases: (a) $\hat{p}_x = 0.80$, $h(x) = 0.76$ and (b) $\hat{p}_x = 0.62$ and $h(x) = 0.90$. Intuitively, the dotted zones in the figures express the disagreements between Teacher and Learner. The probability value, $|\hat{p}_x - h(x)|$, is the error of h for the probabilistic example $\langle x, \hat{p}_x \rangle$.

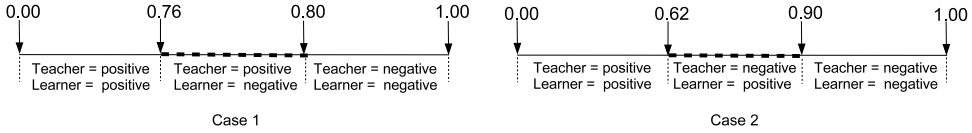


Fig. 2. Probabilistic Decision Makings

Definition 2 (Distributional Consistency). *Assume a set of m probabilistic examples $E' = \{\langle x_1, \hat{p}_1 \rangle, ..., \langle x_m, \hat{p}_m \rangle\}$ is given. Then hypothesis h is distributionally consistent to E' if*

$$\hat{p}_i - \epsilon \leq h(x_i) \leq \hat{p}_i + \epsilon \tag{5}$$

for every $\langle x_i, \hat{p}_i \rangle \in E'$.

Intuitively, the hypothesis h is said to be distributionally consistent to E' if the error of the Learner's classification $h(x_i)$ for the Teacher's classification $\hat{p}_i$ is within ϵ at the $1 - \delta$ confidence level. Based on this new consistency definition, we introduce a new class of *version space*[4] for the given probabilistic examples.

Definition 3 (Distributional Version Space). *Consider a hypothesis space H' and probabilistic training examples E'. Distributional version space is the set of all hypotheses $h \in H'$ that are distributionally consistent to the training example E'.*

$$VS_{H',E'} = \{h \in H' | (\forall \langle x_i, \hat{p}_i \rangle \in E', \hat{p}_i - \epsilon \leq h(x_i) \leq \hat{p}_i + \epsilon\}$$

Intuitively, this distributional version space is a set of hypotheses which are *distributionally consistent* to the given probabilistic examples with the estimation error ϵ. The true error of $h \in H'$ can be defined associated with this parameter as follows.

Definition 4. *Assume a set of m probabilistic examples $E' = \{\langle x_1, \hat{p}_1 \rangle, ..., \langle x_m, \hat{p}_m \rangle\}$ is given where $x_i \in X$. For any distribution D_X over X, the true error of $h \in H'$ is defined as the sum of the weighted errors.*

$$error_{D_X}(h) = \sum_{x \in X} [D_X \begin{cases} \{h(x) - (\hat{p}_i + \epsilon)\}\ ] & (if\ \ \hat{p}_i + \epsilon < h(x)) \\ \{(\hat{p}_i - \epsilon) - h(x)\}\ ] & (if\ \ \hat{p}_i - \epsilon > h(x)) \\ 0\ ] & (if\ \ \hat{p}_i - \epsilon \leq h(x) \leq \hat{p}_i + \epsilon) \end{cases}$$

Note that the given probabilistic examples are random variables since m probabilistic examples are randomly sampled from D_X. That is, the constructed distributional version space might have a large true error if we obtain some unfair examples. Since we cannot avoid such a case, we only consider a class of distributional version space with a small true error ε.

Definition 5 (ε-exhausted Distributional Version Space). *Consider a hypothesis space H' and training probabilistic example E'. The distributional version space is said to be ε-exhausted if every hypothesis $h \in VS_{H',E'}$ has a true error less than equal to ε.*

$$(\forall h \in VS_{H',E'} \quad error_{D_X}(h)) \leq \varepsilon$$

2.2 Result of Sample Complexity in Route 3

In Definition 5, we introduce the class of hypotheses with the two parameters: ϵ and ε. Our next interest is if the hypotheses in ε-exhausted distributional version space can classify a given probabilistic example well or not. First, we analyse how

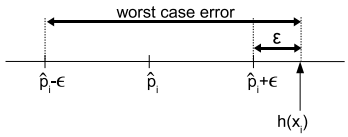


Fig. 3. Worst case error of hypothesis h in ε-exhausted distributional version space for a probabilistic example $\langle x_i, \hat{p}_i \rangle$ with estimation error ϵ

severely a hypothesis in ε-exhausted distributional version space can misclassify any probabilistic example.

Lemma 1. *Assume a hypothesis $h \in VS_{H',E'}$ is given associated with ϵ and ε. Then the error of h for classifying any probabilistic example $\langle x_i, \hat{p}_i \rangle \in E'$ is at most $\varepsilon + 2\epsilon$.*

Proof. Let $h(x_i)$ be the probability returned by h for $\langle x_i, \hat{p}_i \rangle$. Since $h \in VS_{H',E'}$, $|h(x_i) - (\hat{p}_i + \epsilon)|$ is at most ε as shown in Fig. 3. In this case, the error $\varepsilon + 2\epsilon$ is the worst case error for h in the ε-exhausted distributional version space. If $h(x_i) \leq \hat{p}_i + \epsilon)$, $|(\hat{p}_i - \epsilon) - h(x_i)|$ is at most ε and h has the same worst case error, $\varepsilon + 2\epsilon$.

We consider the sample error of the ε exhausted distributional version space for any m probabilistic examples next.

Lemma 2 (ε exhausting the distributional version space). *If the hypothesis space H' is finite and E' is a sequence of $m \geq 1$ independent randomly drawn*

probabilistic examples of some target concept, then for any $0 \leq \varepsilon \leq 1$ *such that* $\varepsilon \leq 1 - 2\epsilon$ *, the probability that the distributional version space* $VS_{H',E'}$ *is not* ε *exhausted (with respect to the target concept) is less than or equal to* $|H'|e^{-(\varepsilon+2\epsilon)m}$.

Proof. Let $h_1, .., h_k$ be all the hypotheses in H' that have true error greater than ε with respect to the target concept. We fail to ε-exhaust the version space if and only if at least one of these k hypotheses happens to be distributionally consistent with all m' independent random probabilistic training examples. Thus the probability that this hypothesis will be distributionally consistent with m independently drawn probabilistic examples is at most $\{1 - (\varepsilon + 2\epsilon)\}^m$. Given that we have k hypotheses with error greater than ε, the probability that at least one of these will be distributionally consistent with all m probabilistic training examples is at most $k\{1 - (\varepsilon + 2\epsilon)\}^m$. Since $k \leq |H'|$, this is at most $|H'|\{1 - (\varepsilon + 2\epsilon)\}^m$. Finally, we use a general inequality stating: $(1 - x) \leq e^{-x}$ if $0 \leq x \leq 1$. For $0 \leq \varepsilon + 2\epsilon \leq 1$, $|H'|\{1 - (\varepsilon + 2\epsilon)\}^m \leq |H'|e^{-(\varepsilon+2\epsilon)m}$ which proves the lemma.

Now, we obtain a sample complexity for the route (3) as follows.

Theorem 2. *Assume we obtain probabilistic training examples with error* ϵ. *For any* ϵ, ε, *and* δ' *such that* $0 \leq \varepsilon + 2\epsilon \leq 1$ *and* $0 \leq \delta' \leq 1$, *let* m *be the number of the probabilistic training examples sufficient for any distributionally consistent learner to successfully learn any target concept in* H' *with true error* ε *in confidence* $(1 - \delta')$. *Then* m *is bounded as follows.*

$$m \geq \frac{ln|H'| + ln\frac{1}{\delta'}}{\varepsilon + 2\epsilon}.$$

Proof. The probability value $|H'|e^{-(\varepsilon+2\epsilon)m}$ monotonically decreases as m increases. Let δ' be a constant for the upper bound of the error $|H'|e^{-(\varepsilon+2\epsilon)m}$:

$$|H'|e^{-(\varepsilon+2\epsilon)m} \leq \delta'.$$

By solving this inequality, $m \geq \frac{ln|H'|+ln\frac{1}{\delta'}}{\varepsilon+2\epsilon}$. is found.

3 Discussions on Sample Complexity

Let us analyse Theorem 2. In PAC learning, Blumer bound [4] provides a lower bound: $m_B \geq \frac{ln|H|+ln\frac{1}{\delta}}{\varepsilon}$ for the consistent learner who classifies each given Boolean labelled examples into binary classes. Our result shows that in the route (3) we need fewer probabilistic examples than the standard PAC learning if the given examples have already been labelled by probability values with their error information. This becomes a positive result for PILP.

Why can PILP have an advantage in terms of the number of examples? The particular difference between our result and Blumer bound is in the term, 2ϵ,

which comes from the definition of distributional consistency. It is the *error tolerance level* for the classification in probability values; the required number of examples could be fewer once we are more tolerant for mistakes. In the standard PAC learning, the consistent learner does not allow to have any mistakes for constructing the version space with the given training examples whereas the distributionally consistent learner can have a small error for making the distributional version space. This is achieved by the fine probability-based classification.

The estimation of the probability labels of probabilistic examples is an important topic in PILP. The *law of large numbers* suggests that ϵ becomes smaller if more non-probabilistic examples are used for estimating the probability label. In that case, we could be less tolerant for the mistakes in the route (3) since we can strongly believe the estimated label. If we prefer the sample complexity being inversely proportional to ϵ, one possible alternation of the definition of (5) is:

$$\hat{p}_i - \frac{1}{\epsilon} \leq h(x_i) \leq \hat{p}_i + \frac{1}{\epsilon}$$

which leads the following new sample complexity:

$$m \geq \frac{ln|H'| + ln\frac{1}{\delta'}}{\varepsilon + 2/\epsilon}.$$

4 Conclusion

The alternative view of PILP is introduced as a way of providing approximate generalisation. Our result of sample complexity of PILP shows that fewer probabilistic examples are required in PILP than the standard PAC learning if the given examples have already been labelled in probability with their error information. This is essentially because of the introduction of error tolerance in the definition of the distributional consistency. In this paper, some initial results of complexity analysis of PILP are stated. More effort is required for clarifying if PILP eases ILP or not in terms of machine learnability. We expect that the new view of PILP develops better understandings of PILP both in theory and applications.

References

1. S.H. Muggleton and L. De Raedt.: Inductive logic programming: Theory and methods. Journal of Logic Programming, 19,20:629-679, 1994.
2. Luc De Raedt and Kristian Kersting.: Probabilistic Inductive Logic Programming. Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT-2004). LNCS 3244. pp. 19-36. Springer. 2004.
3. Leslie G. Valiant.: A Theory of the Learnable. Communications of the ACM 27(11): 1134-1142 (1984)
4. Machine Learning, Tom Mitchell, McGraw Hill, 1997
5. M. J. Kearns and R. E. Schapire.: Efficient distribution-free learning of probabilistic concepts. 31st Annual Symposium on Foundations of Computer Science (pp. 382–391). IEEE Press. 1990.

Author Index

Agnew, Jeffrey, 32
Alphonse, Erick, 1

Buryan, Petr, 7

Costa Florêncio, Christophe, 14

Dědek, Jan, 20
De Raedt, Luc, 38, 63, 111

Eckhardt, Alan, 20

Flach, Peter, 26

Gérard, Pierre, 99
Gao, Qingyi, 26
Goadrich, Mark, 32
Gutmann, Bernd, 38

Holec, Matěj, 50

Ishihata, Masakazu, 44

Józefowska, Joanna, 57

Kameya, Yoshitaka, 44
Kersting, Kristian, 38
Kimmig, Angelika, 38, 63
Kléma, Jiří, 50
Kuželka, Ondřej, 69
Kuzmin, Alexander, 93

Landwehr, Niels, 111
Langley, Pat, 75
Lawrynowicz, Agnieszka, 57
Li, Nan, 75
Lukaszewski, Tomasz, 57

Meilicke, Christian, 105
Minato, Shin-ichi, 44
Moulík, Karel, 81
Muggelton, Stephan, 123
Muggleton, Stephen H., 87

Popelínský, Luboš, 117
Predoiu, Livia, 105

Ralbovský, Martin, 93
Rauch, Jan, 93
Rodrigues, Christophe, 99
Rouveirol, Céline, 99

Santos, Joé C. A., 87
Sato, Taisuke, 44
Stracuzzi, David J., 75
Stuckenschmidt, Heiner, 105
Svoboda, Jiří, 50

Tamaddoni-Nezhad, Alireza, 87
Thon, Ingo, 111
Tolar, Jakub, 50

Večeřa, Martin, 117
Vojtáš, Peter, 20

Watanabe, Hiroaki, 123

Železný, Filip, 50, 69, 81

Title:	**Inductive Logic Programming 2008**
Type of Publication:	Late Breaking Papers
Submitted:	Authors, Co-Authors
Edited:	Filip Železný, Nada Lavrač
Number of Pages:	140
Year of Issue:	2008
Edition:	first
Published:	Zeithamlová Milena, ing. - Agentura Action M Vršovická 68 101 00 Praha 10 actionm@action-m.com http://www.action-m.com
Printed:	Reprostředisko UK MFF Sokolovská 83 186 75 Praha 8

No editorial and stylistic revision.

ISBN 978-80-86742-26-7