A Probabilistic Algorithm for Multi-Relational Data Mining

José E. Ochoa Luna[1], **Fabio G. Cozman**[1]

[1]Escola Politécnica – Universidade de São Paulo (USP)
Av. Prof. Mello Morais, 2231 – 05508-030 – São Paulo – SP – Brazil

eduardo.ol@gmail.com, fgcozman@usp.br

Abstract. *Inductive Logic Programming (ILP) algorithms learn a set of first-order logical rules from multi-relational data, and are thus well suited to several data mining tasks. In this work, we introduce a Probabilistic ILP algorithm to extract useful regularities from multiple data tables. We adopt a probabilistic cover approach that allows us to guide the search for rules; Noisy-OR functions are employed to encode probability distributions. Preliminary tests have been conducted on relational data in the Lattes curriculum platform.*

Resumo. *Algoritmos de Programação Lógica Indutiva (ILP) aprendem um conjunto de regras lógicas de primeira ordem a partir de dados multirrelacionais, e dessa forma são apropriados para várias tarefas de mineração de dados. Este trabalho introduz um algoritmo ILP Probabilístico para extrair regularidades úteis de múltiplas tabelas de dados. Adota-se uma abordagem de cobertura probabilística que possibilita a busca por regras. A distribuição de probabilidades dos predicados é baseada em funções Noisy-OR. Testes preliminares foram realizados em dados relacionais coletados da plataforma de currículos Lattes.*

1. Introduction

Multi-Relational Data Mining (MRDM) is concerned with finding useful patterns from data in multiples tables. Typically, statistical learning methods work with flat data representations, and we are often forced to convert relational data into a flat form, thereby not only losing its compact representation and structure but also potentially introducing statistical bias [Getoor 2001]. Instead we should work directly with the available, relational data. This simple guideline is followed by several methods; there have been efforts to turn propositional data mining methods into relational ones, and relational decision trees and relational distance methods are some successful examples [Dzeroski and Lavrac 2001]. The focus on relational data is also present in methods grouped under the label of Inductive Logic Programming (ILP). The central role of relational databases in data mining [Morik and Brockhausen 1996] greatly motivates the research on such ILP methods.

As a rule, ILP algorithms aim at learning a set of first-order logical rules from multi-relational data and are thus well suited to MRDM tasks [Bockhorst and Ong 2004]. In recent years, techniques for ILP and MRDM have undergone a significant evolution in the direction of more scalable and more efficient systems [Blockeel and Sebag 2003].

Logical and relational languages have been increasingly associated to probabilistic learning methods. Probabilistic logic has been extensively researched in the last decades [de Campos et al. 2009]. This combined approach aims at relaxing some of the strict

requirements of logic, and often aims at using probabilistic graphical models in general settings. In ILP systems, uncertainty has usually had a secondary role. Recently, however, several efforts have been taken to define a Probabilistic ILP framework. In Probabilistic ILP we are concerned with finding useful patterns in the form of annotated rules, i.e., logical predicates annotated with probability values. This sort of rule can be valuable to relational data mining tasks where incomplete or noisy data are present. This paper works in such a direction. We are influenced in several respects by the well-known ILP system FOIL [Quinlan and Mostow 1990, Quinlan and Cameron-Jones 1993]. Differently of the traditional separate-and-conquer approach, we use a probabilistic cover function that guides the search for rules. Furthermore, the probability values are obtained from Noisy-OR functions. Preliminary tests have been performed with the Lattes curriculum dataset, where a relational classification task has been conducted. Preliminary results suggest that our algorithm is as accurate as traditional ILP algorithms, and that uncertainty handling can improve relational data mining tasks.

2. Inductive Logic Programming and Multi-Relational Data Mining

Multi-relational data mining can be formalized either as a constraint satisfaction problem or as an optimization problem. In both cases, the task can be formulated as a search process. Given a hypothesis space H and some real-valued (respectively boolean) criterion c, find the clauses h in H such that they maximize c (resp. such that $c(h)$ holds). We adopt the following definition from [Blockeel and Sebag 2003].

Definition 1 *Multi-relational Data Mining is the process of finding all clauses h, hypotheses, in a language H, that satisfy a predicate c with respect to a database or set of examples E. Predicate c is most often related to the coverage of clause h, or a numerical expression thereof, which must either be greater than a user-fixed threshold, or reach an optimum value.*

Many MRDM algorithms have appeared within ILP. Positioned at the intersection of machine learning and logic programming, ILP has been concerned with finding patterns expressed as *logic programs*. Initially, ILP focused on automated programming synthesis from examples, formulated as a binary classification task. In recent years, however, the scope of ILP has broadened to cover the whole spectrum of data mining tasks [Dzeroski 2003].

In order to define ILP it is useful to review a few logical concepts [Dzeroski 2003]. Logic programs consist of *clauses*. We can think of clauses as first-order rules where the consequent is termed the head and the antecedent the body of the clause. The head and the body of a clause consist of *atoms*, an atom being a predicate applied to some arguments, which are called *terms*. Terms are variables and constants. *Ground clauses* have no variables. Variables in clauses are implicitly universally quantified. Clauses are also viewed as sets of literals, where a literal is an atom or its negation.

As opposed to full clauses, *definite clauses* contain exactly one atom in the head and only positive atoms in the body. As compared to definite clauses, *program clauses* can also contain negated atoms in the body. Logic programs are sets of program clauses. A set of program clauses with the same predicate in the head is called a *predicate definition*. Most ILP approaches learn predicate definitions. A predicate in logic programming corresponds to a relation in a relational database.

A set of examples contains tuples that belong to the target relation p (positive examples) and tuples that do not belong to p (negative examples). Background relations q_i constitute the background knowledge. Finally, a hypothesis language, specifying syntactic restrictions on the definition of p is also assumed given. The task in ILP is to find a definition of the target relation p that is consistent and complete, i.e., explains all the positive and none of the negative tuples.

Definition 2 *Given: a set of examples $E = E_p \cup E_n$, where E_p, E_n denote positive and negative examples respectively, and background knowledge B. The task of ILP is to find a hypothesis H such that $\forall e \in E_p : B \wedge H \models e$ (H is complete) and $\forall e \in E_n : B \wedge H \not\models e$ (H is consistent), where $\models$ stands for logical implication or entailment.*

From a data mining perspective, the task described above is a binary classification task. ILP has always focused on concept learning from positive and negative examples of a concept [De Raedt and Dehaspe 1996]. ILP systems dealing with the classification task typically adopt the *covering* approach of rule induction systems. In a main loop, a covering algorithm constructs a set of clauses. Starting from an empty set of clauses, it constructs a clause explaining some of the positive examples, add these clauses to the hypothesis, and removes the positive examples explained. These steps are repeated until all positive examples have been explained (the hypothesis is complete).

For efficiency reasons[1], the relational matching test used in the ILP literature is the *θ-subsumption* test [Blockeel and Sebag 2003]; it amounts to finding a variable instantiation for the clause body (resp. head) such that this body (resp. head) becomes a subset of all facts in the examples. A formal definition is given below

Definition 3 (θ-Subsumption). *Let c and c' be two program clauses. Clause c θ-subsumes c' if there exists a substitution θ such that $c\theta \subseteq c'$.*

Note that θ-subsumption testing is NP-complete [Dzeroski 2003] due to the fact that the literals in the clause and example need to be matched to each other, and the number of possible matchings grows combinatorially in the number of literals. The simplest way of reducing the θ-subsumption cost is to consider only short hypotheses, if at all possible. In fact, this heuristic is built-in in most ILP systems.

3. A Probabilistic ILP Algorithm

As noted by De Raedt [De Raedt and Kersting 2008], Probabilistic Inductive Logic Programming deals with machine learning and data mining in relational domains where observations may be missing, partially observed and/or noisy. ILP does not explicitly deal with uncertainty such as missing or noisy information. Dealing explicitly with uncertainty makes probabilistic ILP more powerful than ILP and, in turn, than traditional attribute-value approaches.

To extend ILP ideas to the probabilistic case we must make use of a probabilistic covers relation [De Raedt and Kersting 2008]:

Definition 4 (Probabilistic Covers Relation) *A probabilistic covers relation takes as arguments an example e, a hypothesis H and possibly the background theory B, and*

[1]Logical implication is not decidable in the general case. For this reason, the ILP literature uses a weaker covering test, correct but not complete [Blockeel and Sebag 2003].

returns the probability value $P(e|H,B)$ between 0 and 1 of the example e given H and B, i.e., $covers(e,H,B) = P(e|H,B)$.

Given Definition 4 we can explain the Probabilistic ILP learning problem as follows:

Definition 5 (Probabilistic ILP Learning) *Given a probabilistic logical language $\mathcal{L}_H$ and a set E of examples over some language $\mathcal{L}_E$, find the hypothesis H^* in $\mathcal{L}_H$ that maximizes $P(E|H^*,B)$.*

Under the usual i.i.d. assumption (examples are sampled independently from identical distributions), this results in the maximization of $P(E|H^*,B) = \prod_{e\in E} P(e|H^*,B) = \prod_{e\in E} covers(e,H^*,B)$. Negative examples conflict somewhat with the usual techniques in statistical learning. Thus we seek to find that hypothesis H^*, which is most likely given the learning examples: $H^* = \arg\max_H P(H|E)$. In contrast to the purely logical case of ILP, we do not speak of *positive* and *negative* examples anymore but of *observed* and *unobserved* ones hence, the Probabilistic ILP problem is given by:

Definition 6 (The Probabilistic ILP Problem) *Given a set $E = E_p \cup E_i$ of observed and unobserved examples E_p and E_i (with $E_p \cap E_i = \emptyset$) over some example language $\mathcal{L}_E$, a probabilistic covers relation covers$(e,H,B) = P(e|H,B)$, a probabilistic logical language $\mathcal{L}_H$ for hypotheses, and a background theory B, find a hypothesis H^* such that $H^* = \arg\max_H$score(E,H,B) and the following constraints hold: $\forall e_p \in E_p$: $covers(e_p,H^*,B) > 0$ and $\forall e_i \in E_i : covers(e_i,H^*,B) = 0$. The score is an objective function, usually using the probabilistic covers relation of the observed examples such as the observed likelihood $\prod_{e_p\in E_p}$covers(e_p,H^*,B) or some penalized variant.*

We annotate a logic program H consisting of a set of clauses of the form $p \leftarrow b_i$, where p is an atom of the form $p(V_1,\ldots,V_n)$ the V_i different variables, and the b_i are different bodies of clauses. Furthermore, we associate to each clause in H the probability values $P(b_i|p)$; they constitute the conditional probability distribution that for a random substitution θ for which $p\theta$ is ground and true (resp. false), the query $b_i\theta$ succeeds (resp. fails) in the knowledge base B. Furthermore, we assume the prior probability of p is given by $P(p)$. It denotes the probability that for a random substitution θ, $p\theta$ is true (resp. false). This can then used to define the covers relation $P(p\theta|H,B)$ as follows (we delete the B as it is fixed):

$$P(p\theta|H) = P(p\theta|b_1\theta,\ldots,b_k\theta) = \frac{P(b_1\theta,\ldots,b_k\theta|p\theta)\times P(p\theta)}{P(b_1\theta,\ldots,b_k\theta)}. \quad (1)$$

By applying the naive Bayes assumption we obtain the nFOIL algorithm [Landwehr et al. 2007]:

$$P(p\theta|H) = \frac{\prod_i P(b_i\theta|p\theta)\times P(p\theta)}{P(b_1\theta,\ldots,b_k\theta)}.$$

In this paper we propose an alternative approach: instead of using naive Bayes assumption, we resort to a popular pattern of probabilistic reasoning, the *Noisy-OR* combination function [Pearl 1998, Cozman 2004]. The probability distribution $P(p\theta|b_1\theta,\ldots,b_k\theta)$ in Equation 1 can be represented as the Bayesian network fragment in Figure 1 (Left), where variables $b_i\theta$ are parents of $p\theta$. In this case $p\theta$ is also called a

collider. Figure 1 (Right) also illustrates a more general model, where $I_1 \ldots I_k$ denote inhibitory variables and g denotes a general combination function (deterministic such as OR function but may be also probabilistic).

Given that all variables are binary, then the complete specification of $P(p\theta|b_1\theta, \ldots, b_k\theta)$ requires 2^k probability values. An attractive strategy is to find methods that specify $P(p\theta|b_1\theta, \ldots, b_k\theta)$ using fewer parameters.

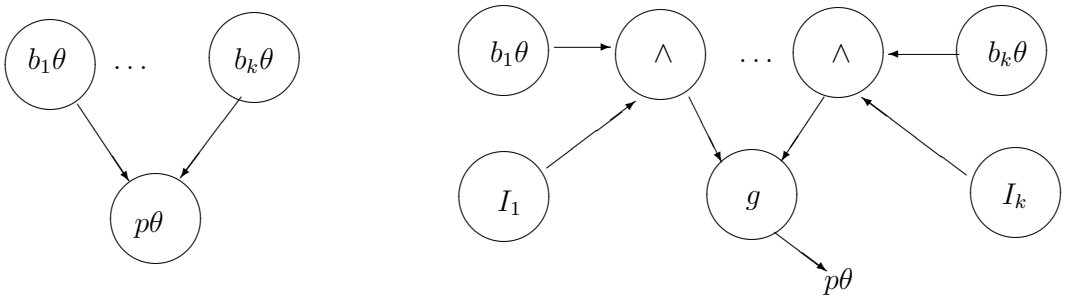

Figure 1. Left: A collider $p\theta$ with parents $b_1, \ldots, b_k\theta$. Right: Inhibitory variables and the combination function g.

The Noisy-OR function is a compact representation for the distribution of colliders. The idea is to start with k probability values l_i, where l_i is the probability that $\{p\theta = true\}$ conditional on $\{b_i\theta = true\}$ and $b_j\theta = false$ for $j \neq i$. That is,

$$l_i = P(p\theta = true|b_i\theta = true, \{b_j\theta = false\}_{j=1,j\neq i}^{k}).$$

The probabilities l_i are called *link probabilities*. Suppose each variable $b_i\theta$ is examined and, if it is $true$, then there is chance $(1 - l_i)$ that it is flipped to $false$; if $b_i\theta = false$, then it stays with $false$. Denote by $b_i^{'}\theta$ the result of flipping (or not) $b_i\theta$. Finally, suppose that

$$p\theta = (b_1^{'}\theta \vee b_2^{'}\theta \ldots b_k^{'}\theta);$$

that is, $p\theta$ is the OR combination of the $b_i^{'}\theta$. In short, the Noisy-OR function is a disjunction of "noisy" versions of $b_i\theta$. The distribution of $p\theta$ conditional on $b_1\theta, \ldots, b_k\theta$ is

$$P(p\theta = true|b_1\theta, \ldots, b_k\theta) = 1 - \prod_{i:b_i\theta=true} (1 - l_i). \quad (2)$$

To produce algorithms that solve the Probabilistic ILP learning problems, say for density estimation, one typically distinguishes two subtasks for $H = (L, \lambda)$:[2] 1) parameter estimation and 2) structure learning. We are interested in the second task; our algorithm is described in Figure 2 .

Given a training set E containing positive and negative examples (i.e. true and false ground facts), this algorithm learns from observed examples so as to induce a probabilistic logical model (to distinguish between the positive and negative examples). The algorithm computes Horn clause features $b_1, b_2, \ldots$ in an outer loop. It terminates when no further improvements in the score are obtained, i. e., when

[2] $H = (L, \lambda)$ is essentially a logic program L annotated with probabilistic parameters λ.

Input: a target predicate p, background knowledge B, a training set $E = E_p \cup E_i$ observed and unobserved examples for p.
Output: an induced predicate definition for p.
repeat
compute hypotheses $p \leftarrow b^1_{i+1}, \ldots, b^k_{i+1}$, where $b^j_{i+1} \in L_H$
let $b^1_{i+1}\theta, \ldots, b^k_{i+1}\theta$ be features of the probabilistic Noisy-OR model,
for all $b^j_{i+1}\theta$
compute maximum likelihood parameters λ^j_{i+1}
compute score $\prod_{e_p \in E_p} covers(e_p, (\{b_1, \ldots, b_i, b^j_{i+1}\theta\}, \lambda^j_{i+1}), B)$
let $b_{i+1}\theta$ the hypothesis with the best score
according to $b_{i+1}\theta$ add b_{i+1} to p
until $score(E, (\{b_1, \ldots, b_i\}, \lambda_i), B) > score(E, (\{b_1, \ldots, b_{i+1}\}, \lambda_{i+1}), B)$

Figure 2. Complete learning algorithm.

$score(E, (\{b_1, \ldots, b_i\} \cup \lambda_i), B) < score(E, (\{b_1, \ldots, b_{i+1}\} \cup \lambda_{i+1}), B)$, where λ denotes the maximum likelihood parameters. A major difference with FOIL is, however, that the covered positive examples are not removed. The inner loop is concerned with inducing the next feature b_{i+1} top-down, from general to specific. To this aim it starts with a clause with an empty body. This clause is then specialized by repeatedly adding atoms to the body. For each refinement b^j_{i+1} we then compute the maximum-likelihood parameters λ^j_{i+1} and $score(E, (\{b_1, \ldots, b_i, b^j_{i+1}\} \cup \lambda^j_{i+1}), B)$. The refinement that scores best, say b_{i+1}, is added to the target predicate.

4. Relational Datasets and Preliminary Results

Preliminary tests have been conducted on two well-known ILP datasets. First, the Kinship dataset,[3] a small relational database that consists of 24 unique names in two families (with equivalent structures). The basic task consists in learning rules for the following relations: wife, husband, mother, father, daughter, son, sister, brother, aunt, uncle, niece, and nephew (input data is given by tuples such as: $\texttt{father}(\texttt{Christopher}, \texttt{Arthur}), \texttt{niece}(\texttt{Charlotte}, \texttt{Arthur}) \ldots$). Given a name and a relation, the learned rule must be true for only those individuals that satisfy the relation. In our experiments we have used 100 tuples for training and 4 for testing, this experiment was repeated 20 times (the same as FOIL). The algorithm was correct 78 out of 80 times on the test cases.

Second, the Chess (King-Rook vs. King) dataset (KRK)[4], is a set of examples of chess positions described only by the coordinates of the pieces on the board. We have used a restricted KRK dataset where we aim at learning `illegal` positions. After running 10000 trials on provided test data we have obtained only 41 errors. These results can be compared, in terms of accuracy, to [Alphonse and Rouveirol 2006] which for the KRK (illegal) dataset has obtained: FOIL $(97, 2)$, TILDE $(75, 1)$, RELAGGS $(72, 3)$, RSD$(76, 2)$ PROPAL (100).

In order to prove suitability of the algorithm to real world domains, we have run preliminary tests on relational data elicited from the Lattes curriculum platform[5], the Brazilian government scientific repository. The Lattes platform is a public source of relational data about scientific research. Because the available format is HTML, we have

[3]http://archive.ics.uci.edu/ml/datasets/Kinship.
[4]http://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King).
[5]http://lattes.cnpq.br/.

implemented a semi-automated procedure to extract content. A restricted database has been constructed based on randomly selected documents.

In these preliminary tests we have focused on relational classification. We aim at learning a rule that allows us to identify conflicting publication entries. One can be interested in retrieving the number of publications for a given research group. Whereas this task might seem trivial, difficulties may arise mainly due to multi-authored documents. In principle, each co-author would have a different entry for the same publication in the Lattes platform, and it must be emphasized that each entry procedure can be prone to errors. To overcome this problem, one can employ text analysis methods. Alternatively, we have used a relational approach for addressing this issue.

True conflicting documents have been collected and some of them have been used for testing the target concept: $\texttt{samePublication}(\texttt{A}, \texttt{B})$. Background knowledge includes relations such as: `researcher`, `similarTitle`, `collaborates` and `publication`. After learning, we have obtained the following definition:

$$
\begin{array}{ll}
\texttt{samePublication}(\texttt{A}, \texttt{B}) :- & \texttt{document}(\texttt{A}, \_, \texttt{D}, \_), \texttt{document}(\texttt{B}, \_, \texttt{D}, \_), \\
& \texttt{similarTitle}(\texttt{B}, \texttt{A}). \\
\texttt{samePublication}(\texttt{A}, \texttt{B}) :- & \texttt{document}(\texttt{A}, \_, \texttt{D}, \_), \texttt{document}(\texttt{B}, \_, \texttt{D}, \_), \\
& \texttt{similarTitle}(\texttt{A}, \texttt{B}), \texttt{samePublication}(\texttt{B}, \_). \\
\texttt{samePublication}(\texttt{A}, \texttt{B}) :- & \texttt{similarTitle}(\texttt{A}, \texttt{B}), \texttt{document}(\texttt{A}, \_, \texttt{D}, \texttt{E}), \\
& \texttt{document}(\texttt{B}, \texttt{F}, \texttt{D}, \texttt{E}), \texttt{document}(\texttt{G}, \texttt{F}, \_, \texttt{E}), \\
& \texttt{samePublication}(\texttt{G}, \_).
\end{array}
\tag{3}
$$

For the sake of simplicity we have used _ instead of uninformative variables. The first pattern states that a same publication can be found when two different entries (A and B) have both the same type (D) and similar title. The second pattern adds a recursive rule. Further constraints have been added in the third pattern; same publications have both the same type (D) and year of publication (E). This predicate definition has been successful in identifying duplicate documents; the algorithm has been run over 115 trials and produced only 5 errors. This preliminary result suggest the method is useful in finding useful relationships in relatively complex data.

5. Conclusion

A large portion of real-world data is stored in commercial relational database systems, and we would like to discover interesting statistical correlations in that data. Most statistical learning methods work only with flat data representations; a new trend of research is to focus the probabilistic databases. MRDM can use probabilistic ideas to extend ILP algorithms, so as to obtain scalable MRDM frameworks.

In this paper we have introduced a probabilistic learning algorithm for ILP tasks that is suitable for MRDM. Tests have been carried out on relational data elicited from the Lattes curriculum platform; two well-known ILP datasets have also been explored. Preliminary results shows that our algorithm is as accurate as traditional ILP algorithms. In the short future we plan to further investigate probabilistic algorithms for relational data minings tasks such as probabilistic relational clustering and probabilistic relational rules association.

Acknowledgements

The first author is supported by CAPES. The second author is partially supported by CNPq. The work reported here has received substantial support through FAPESP grant 2008/03995-5.

References

Alphonse, É. and Rouveirol, C. (2006). Extension of the top-down data-driven strategy to ILP. In *Proceedings ILP 2006* , pages 49–63.

Blockeel, H. and Sebag, M. (2003). Scalability and efficiency in multi-relational data mining. *SIGKDD Explor. Newsl.*, 5(1):17–30.

Bockhorst, J. and Ong, I. (2004). FOIL-D: Efficiently scaling FOIL for multi-relational data mining of large datasets. In Magnenat-Thalmann, N. and Thalmann, D., editors, *14th International Conference, ILP 2004*, pages 63–79. Springer Berlin - Heidelberg.

Cozman, F. G. (2004). Axiomatizing Noisy-OR. In *Proceedings of the European Conference on Artificial Intelligence*, pages 979–980.

de Campos, C. P., Cozman, F. G., and Ochoa-Luna, J. E. (2009). Assembling a consistent set of sentences in relational probabilistic logic with stochastic independence. *Journal of Applied Logic*, 7:137–154.

De Raedt, L. and Dehaspe, L. (1996). Clausal discovery. In *Machine Learning*, pages 1058–1063. Morgan Kaufmann.

De Raedt, L. and Kersting, K. (2008). Probabilistic inductive logic programming. In *Probabilistic ILP - LNAI 4911*, pages 1–27. Springer-Verlag Berlin.

Dzeroski, S. (2003). Multi-relational data mining: An introduction. In *SIGKDD Explorations*.

Dzeroski, S. and Lavrac, N. (2001). *Relational Data Mining*. Springer-Berlin.

Getoor, L. (2001). Multi-relational data mining using probabilistic relational models: research summary. In *Proceedings of the First Workshop in Multi-relational Data Mining*.

Landwehr, N., Kersting, K., and DeRaedt, L. (2007). Integrating Naïve Bayes and FOIL. *J. Mach. Learn. Res.*, 8:481–507.

Morik, K. and Brockhausen, P. (1996). A multistrategy approach to relational knowledge discovery in databases. *Machine Learning Journal*, 27:17–27.

Pearl, J. (1998). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo.

Quinlan, J. R. and Cameron-Jones, R. M. (1993). FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20. Springer-Verlag.

Quinlan, J. R. and Mostow, J. (1990). Learning logical definitions from relations. In *Machine Learning*, pages 239–266.