Parameter and Structure Learning Algorithms for Statistical Relational Learning

Elena Bellodi
Supervisor: Fabrizio Riguzzi

ENDIF, Università di Ferrara
Via Saragat, 1 – 44122 Ferrara, Italy
elena.bellodi@unife.it

1 Introduction

My research activity focuses on the field of Machine Learning. Two key challenges in most machine learning applications are uncertainty and complexity. The standard framework for handling uncertainty is probability, for complexity is first-order logic. Thus we would like to be able to learn and perform inference in representation languages that combine the two. This is the focus of the field of Statistical Relational Learning.

My research is based on the use of the vast plethora of techniques developed in the field of Logic Programming, in which the distribution semantics [16] is one of the most prominent approaches. This semantics underlies, e.g., Probabilistic Logic Programs,Probabilistic Horn Abduction,PRISM [16], Independent Choice Logic,pD,Logic Programs with Annotated Disjunctions (LPADs) [17], ProbLog [5] and CP-logic. These languages have the same expressive power: there are linear transformations from one to the others. LPADs offer the most general syntax, so my research and experimentations has been focused on this formalism. An LPAD consists of a finite set of disjunctive clauses, where each of the disjuncts in the head of a clause is annotated with the probability of the disjunct to hold, if the body of the clause holds. LPADs are particularly suitable when reasoning about actions and effects where we have causal independence among the possible different outcomes for a given action.

Various works have appeared for solving three types of problems for languages under the distribution semantics:

- inference: computing the probability of a query given the model and some evidence: most algorithms find explanations for queries and compute their probability by building a Binary Decision Diagram (BDD) [5,15,10];
- learning the parameters: for instance, LeProbLog [6] uses gradient descent while LFI-ProbLog [7] uses an Expectation Maximization approach where the expectations are computed directly using BDDs;
- learning the structure: in [13] a theory compression algorithm for ProbLog is presented, in [12] ground LPADs are learned using Bayesian network techniques.

A different approach from distribution semantics is represented by Markov Logic: a Markov Logic Network (MLN) is a first-order knowledge base with a weight attached to each clause, reflecting "how strong" it is. For this language inference, parameter and structure learning algorithms are available as well, see [14] and [11].

By means of the development of systems for solving the above problems, one would like to handle classical machine learning tasks, such as Text Classification, Entity Resolution, Link Prediction, Information Extraction, etc., in real world domains. An example of Text Classification problem is given by the WebKB dataset, containing the text of web pages from 4 universities; each page belongs to one of four classes: course, faculty, research project or student. Given words on the pages, one wish to infer the class. Entity Resolution concerns the problem of information integration from multiple sources, where the same entities can be differently described. An example is represented by the Cora dataset, containing citations of computer science publications, since citations of the same paper often appear differently.

I have considered the problem of learning the parameters and both the parameters and the structure of LPADs by developing three learning systems: section 2 presents a parameter learning algorithm based on Expectation Maximization, section 3 presents two algorithms for structure learning that exploit the first.

2 Learning LPADs Parameters with the EM Algorithm

An LPAD is composed of annotated disjunctive clauses C_i of the form $h_{i1} : \Pi_{i1}; \ldots; h_{in_i} : \Pi_{in_i} : -b_{i1}, \ldots, b_{im_i}$. where $h_{i1}, \ldots h_{in_i}$ are logical atoms, $b_{i1}, \ldots, b_{im_i}$ are logical literals and $\{\Pi_{i1}, \ldots, \Pi_{in_i}\}$ are real numbers in the interval $[0, 1]$ representing probabilities and sum up to 1.

An LPAD rule containing variables represents a number of "simple experiments", one for each ground instantiation of the rule (obtained by replacing variables with constants of the domain), with the disjuncts as the possible outcomes. For each ground instantiation of a rule *only* one pair $(h : \Pi)$ is chosen; in this way a normal non-disjunctive logic program is obtained, called *possible world*. A probability distribution is defined over the space of possible worlds by assuming independence among the selections made for each rule. The probability of a possible world is the product of probabilities of the individual heads chosen in each rule. The probability of a query Q according to an LPAD is given by the sum of the probabilities of the possible worlds where the query is true.

Example 1. The following LPAD T encodes the result of tossing a coin depending on the fact that it is biased or not:

$C_1 = heads(Coin) : 0.5; tails(Coin) : 0.5 : -toss(Coin), \neg biased(Coin).$
$C_2 = heads(Coin) : 0.6; tails(Coin) : 0.4 : -toss(Coin), biased(Coin).$
$C_3 = fair(coin) : 0.9; biased(coin) : 0.1.$
$C_4 = toss(coin) : 1.$

This program models the fact that a fair coin lands on heads or on tails with probability 0.5, while a biased coin with probabilities 0.6 and 0.4 respectively. The third clause says that a certain coin *coin* has a probability of 0.9 of being fair and of 0.1 of being biased, the fourth one that *coin* is certainly tossed.

Each selection of a disjunct in a ground clause of an LPAD can be represented by the equation $X_{ij} = k$, where $k \in \{1, \ldots, n_i\}$ indicates the head chosen, X_{ij} is a multivalued random variable where i and j indicate the clause and the grounding. A function $f(\mathbf{X})$, built on a set of multivalued variables and taking Boolean values, can be represented by a Multivalued Decision Diagram (MDD), a rooted graph that has one level for each variable. Each node has one child for each possible value of the associated variable. The leaves store either 0 or 1, the possible values of $f(\mathbf{X})$. Given values for all the variables $\mathbf{X}$, a MDD can be used for computing the value of $f(\mathbf{X})$ by traversing the graph starting from the root and returning the value associated to the leaf that is reached. An example of such function is: $f(\mathbf{X}) = \{X_{11} = 1 \vee X_{21} = 2 \wedge X_{31} = 1 \vee X_{22} = 3 \wedge X_{31} = 1\}$. A MDD can be used to represent the set of selections over rules, and will have a path to a 1-leaf for each possible world where a query Q is true. It is often unfeasible to find all the worlds where the query is true so inference algorithms find instead *explanations* for the query, i.e. set of selections such that the query is true in all the worlds whose selection is a superset of them. Since MDDs split paths on the basis of the values of a variable, the branches are mutually disjoint so that a dynamic programming algorithm can be applied for computing the probability of a query by a summation. Usually one works on MDDs with a Binary Decision Diagram package, so one has to represent multivalued variables by means of Boolean variables.

The problem I faced is how to efficiently perform "parameter learning", i.e., using training data for learning correct probabilities Π_{ik}. The technique applied exploits the EM (Expectation Maximization) algorithm over BDDs proposed in [9,8] and has been implemented in the system EMBLEM, for "EM over BDDs for probabilistic Logic programs Efficient Mining" [1,2,3].

EMBLEM takes as input a set of interpretations (sets of ground facts), each describing a portion of the domain of interest, and a theory (LPAD). The user has to indicate which, among all predicates, are target predicates: the facts for these predicates will form the queries for which a SLD-proof is computed; from these proofs a BDD is built encoding the Boolean formula consisting of the disjunction of the explanations for the query.

Then EMBLEM performs an EM cycle, in which the steps of Expectation and Maximization are repeated until the log-likelihood of the examples reaches a local maximum. Expectations are computed directly over BDDs. EM is necessary to determine the parameters Π_{ik} since the number of times that head h_{ik} is chosen is required. The information about which selection was used is unknown, so the "choice" variables are latent and the number of times is a sufficient statistic.

Decision Diagrams are suitable to efficiently evaluate the expectations since the set of selections used for the derivation of the examples can be represented as the set of paths from the root to the 1-leaf.

3 Learning LPADs Structure

The first system developed for LPADs' structure learning is SLIPCASE, for "Structure LearnIng of ProbabilistiC logic prograAmS with Em over bdds" [4]. It learns a LPAD by starting from an initial theory and by performing a beam search in the space of refinements of the theory. The initial theory is inserted in the beam and, at each step, the theory with the highest log likelihood is removed from the beam and the set of its refinements, allowed by the language bias, is built. The possible refinements are: the addition/removal of a literal from a clause, the addition of a clause with an empty body and the removal of it. For each refinement an estimate of the log likelihood of the data is computed by running a limited number of iterations of EMBLEM. The best theory found so far is updated and each refinement is inserted in order of log likelihood into the beam.

I am now working on SLIPCOVER, an evolution of SLIPCASE which first searches the space of clauses and then the space of theories. SLIPCOVER performs a cycle for each predicate that can appear in the head of clauses, where a beam search in the space of clauses is performed: each clause (built according to a language bias) is tested on the examples for the predicate, its head parameters are learned with EMBLEM and the log likelihood of the data is used as its score. Then the clause is inserted into one of two lists of promising clauses: a list of target clauses, those for predicates we want to predict, and a list of background clauses, those for the other predicates. Then a greedy search in the space of theories is performed, in which each target clause is added to the current theory and the score is computed. If the score is greater than the current best the clause is kept in the theory, otherwise it is discarded. Finally parameter learning with EMBLEM is run on the target theory plus the clauses for background predicates.

The two systems can learn general LPADs including non-ground programs.

4 Experiments

We experimented EMBLEM on the real datasets IMDB, Cora, UW-CSE, WebKB, MovieLens and Mutagenesis and evaluated its performances by means of the Area under the PR curve and under the ROC curve, in comparison with five logic-probabilistic learning systems. It achieves higher areas in all cases except two and uses less memory, allowing it to solve larger problems often in less time.

We tested SLIPCASE on the real datasets HIV, UW-CSE and WebKB, and evaluated its performances - in comparison with [11] and [12] - through the same metrics, obtaining highest area values under both.

We have tested the second structure learning algorithm on HIV, UW-CSE, WebKB, Movielens, Mutagenesis and Hepatitis and evaluated it - in comparison with SLIPCASE, [11] and [12] - through the same metrics. It has overcome them in all cases. In the future we plan to test the systems on other datasets and to experiment with other search strategies.

References

1. Bellodi, E., Riguzzi, F.: EM over binary decision diagrams for probabilistic logic programs. In: Proceedings of the 26th Italian Conference on Computational Logic (CILC2011), Pescara, Italy, 31 August 31-2 September, 2011. pp. 229–243. No. 810 in CEUR Workshop Proceedings, Sun SITE Central Europe, Aachen, Germany (2011), http://www.ing.unife.it/docenti/FabrizioRiguzzi/Papers/BelRig-CILC11.pdf
2. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intel. Data Anal. 16(6) (2012), to appear
3. Bellodi, E., Riguzzi, F.: Experimentation of an expectation maximization algorithm for probabilistic logic programs. Intelligenza Artificiale (2012), to appear
4. Bellodi, E., Riguzzi, F.: Learning the structure of probabilistic logic programs. In: Inductive Logic Programming 21st International Conference, ILP 2011, London, UK, July 31 - August 3, 2011. Revised Papers. LNCS, Springer (2012), to appear
5. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic prolog and its application in link discovery. In: International Joint Conference on Artificial Intelligence. pp. 2462–2467. AAAI Press (2007)
6. Gutmann, B., Kimmig, A., Kersting, K., Raedt, L.D.: Parameter learning in probabilistic databases: A least squares approach. In: European Conference on Machine Learning and Knowledge Discovery in Databases. LNCS, vol. 5211, pp. 473–488. Springer (2008)
7. Gutmann, B., Thon, I., Raedt, L.D.: Learning the parameters of probabilistic logic programs from interpretations. In: European Conference on Machine Learning and Knowledge Discovery in Databases. LNCS, vol. 6911, pp. 581–596. Springer (2011)
8. Inoue, K., Sato, T., Ishihata, M., Kameya, Y., Nabeshima, H.: Evaluating abductive hypotheses using an em algorithm on bdds. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI). pp. 810–815. Morgan Kaufmann Publishers Inc. (2009)
9. Ishihata, M., Kameya, Y., Sato, T., Minato, S.: Propositionalizing the em algorithm by bdds. In: Late Breaking Papers of the International Conference on Inductive Logic Programming. pp. 44–49 (2008)
10. Kimmig, A., Demoen, B., De Raedt, L., Santos Costa, V., Rocha, R.: On the implementation of the probabilistic logic programming language problog. Theory and Practice of Logic Programming 11(2-3), 235–262 (2011)
11. Kok, S., Domingos, P.: Learning markov logic networks using structural motifs. In: International Conference on Machine Learning. pp. 551–558. Omnipress (2010)
12. Meert, W., Struyf, J., Blockeel, H.: Learning ground CP-Logic theories by leveraging Bayesian network learning techniques. Fundam. Inform. 89(1), 131–160 (2008)
13. Raedt, L.D., Kersting, K., Kimmig, A., Revoredo, K., Toivonen, H.: Compressing probabilistic prolog programs. Mach. Learn. 70(2-3), 151–168 (2008)
14. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
15. Riguzzi, F.: Extended semantics and inference for the Independent Choice Logic. Logic Journal of the IGPL 17(6), 589–629 (2009)
16. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: International Conference on Logic Programming. pp. 715–729. MIT Press (1995)
17. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: International Conference on Logic Programming. LNCS, vol. 3131, pp. 195–209. Springer (2004)