Learning Compact Markov Logic Networks With Decision Trees

Hassan Khosravi, Oliver Schulte, Jianfeng Hu
{hkhosrav, oschulte, jhfu}@sfu.ca

School of Computing Science
Simon Fraser University
Vancouver-Burnaby, B.C., Canada

Abstract. Statistical-relational learning combines logical syntax with probabilistic methods. Generative models represent the joint distribution of relationships and attributes. Bayes net learning techniques can be applied to learn a set of probabilistic Horn clauses. Inference can be carried out by converting the Horn clauses to Markov Logic Network clauses (MLNs). This moralization approach takes advantage of the high-quality inference algorithms for MLNs and their ability to handle cyclic dependencies. A weakness of the moralization approach is that it leads to an unnecessarily large number of clauses when the Bayes net fails to capture local or context-sensitive independencies. As MLNs have one weight parameter for each clause, this decreases the accuracy of parameter estimates, and slows down inference. In this paper we show that using decision trees to represent conditional probabilities in the Bayes net is an effective remedy for reducing the number of model parameters. In experiments on benchmark datasets, the reduction ranged from a factor of 5-25, depending on the dataset. The accuracy of predictions is competitive with the unpruned model and in many cases superior.

1 Introduction

An important task in learning with relational data is to build a generative model that represents probabilistic patterns over both links and attributes. Markov Logic Networks (MLNs) are a prominent generative relational model [2]. An MLN is represented as a set of weighted clauses in first-order logic. MLNs have achieved impressive performance on a variety of relational learning tasks. Because they are based on undirected graphical models, they avoid the difficulties with cycles that arise in directed relational models [16, 2]. High-quality inference algorithms for MLNs are available with a sound theoretical foundation. An open-source benchmark system for MLNs is the Alchemy package [13].

Structure Learning via Moralization. MLN structure learning is challenging, especially in datasets with many descriptive attributes. One of the most effective methods is the moralization approach [10]. This method learns a 1st-order Bayes net model for an input relational database. The Bayes net is then converted to an MLN using the moralization method, as described by Domingos and Richardson [2, 12.5.3]. In graphical terms, moralization connects all co-parents, then omits edge directions. In logical terms, moralization converts Horn clauses to conjunctions of literals. The moralization approach combines efficient learning techniques of directed models with the high-quality inference and theoretical foundations of undirected

models. A disadvantage of the moralization approach is that it adds a clause for each conditional probability parameter in the Bayes net. This produces a relatively large number of rules. While this rich structure captures most of the relevant correlations in the data, the large number of clauses has several drawbacks. (i) The resulting MLN is harder for a user to understand. (ii) Parameter learning is slower. (iii) Inference is slower. (iv) Since each clause requires the estimate of a separate weight parameter, the larger number of clauses decreases the accuracy of parameter estimates [5]. This paper presents an extension of the moralization approach that produces significantly smaller MLN structures without sacrificing statistical power.

Decision Trees and MLN Clauses. 1st-order Bayes net structure learning searches for associations between functions or predicates, rather than associations between function/predicate values (literals). As discussed by Kersting and deRaedt, predicate-level search makes for efficient learning [9]; however, it may fail to capture local independencies that holds conditional on specific values of the random variables [5]. Decision trees are a common way to represent local independencies: instead of keeping a conditional probability table each for nodes of the Bayes net, we learn a decision tree that predicts the probability of a child node value given values for its parents [5]. To our knowledge, this is the first paper to apply decision tree learning to MLN parameter reduction. A Bayes net structure with decision trees can be converted to an MLN by adding a formula for each branch in each tree that is the conjunction of the literals along the branch. Previous methods for learning compact MLNs typically remove/shorten clauses based on optimizing a regularized likelihood score [8]. Compared with MLN pruning, decision tree learning has two novel aspects. (i) The pruning takes place at the level of predicates, rather than literals since removing a node in the decision tree removes a predicate along that branch. In the MLN view, decision tree pruning *merges* different clauses into one rather than simply removing an entire clause. This implicitly searches a larger space of MLN clauses than simple pruning by assigning 0 weights. (ii) Since pruning is based on a directed model, it can be done locally by considering only the conditional distribution of a single child given its parents.

2 Additional Related Work

Bayes nets and Decision Trees. The use of decision trees has been long established to reduce the number of parameters after a Bayes net structure has been learned [5, Sec.1]. Friedman and Goldszmidt [5] provide evidence that integrating decision tree search with the Bayes net graph search leads to more accurate Bayes net models; we leave this option for future work.

Statistical-Relational Learning and Decision Trees. We use Parametrized Bayes Nets [17] as a relatively straightforward extension of Bayes nets for relational data. While the combination of Parametrized Bayes nets with decision trees appears to be new, previous statistical-relational formalisms also use decision trees for compact representation of conditional probabilities. We note in particular Logical Bayesian Networks [3] that use decision trees to represent CP-tables. The main difference with our use of decision trees and that of Logical Bayesian Networks, is that the latter interpret decision branches are interpreted as *existentially quantified* conjunctions of literals (as in Tilde), which is different from the grounding semantics of MLN formulas.

3 Relational Structure Learning With Decision Trees

Parametrized Bayes nets are a basic statistical-relational model [17]. A **functor** is a function symbol or a predicate symbol. In this paper we discuss only functors with a finite domain of possible values. A **parametrized random variable** has the form $f(X_1, \ldots, X_k) = f(\mathbf{X})$ where f is a functor and each first-order variable X_i is of the appropriate type for the functor. A **Parametrized Bayes Net** (PBN) is a Bayes net whose nodes are parametrized random variables. The nodes in a **decision tree** for a parametrized random variable c are parametrized random variables. The leaves are labelled with probabilities for the different possible values of the c variable. A decision tree is converted to a Markov Logic Network by adding one clause for the conjunction of literals that corresponds to each of its branches. the **join** of two tables is a new table that contains the rows in the Cartesian products of the tables whose values match on common fields. MLN learning with decision trees proceeds as follows. The full version of the paper will include pseudocode.

1. Learn a Parametrized Bayes net for the input database. We employ the recent learn-and-join algorithm [10].
2. For each node in the PBN, learn a decision tree that represents the conditional distribution of the node given its parents. For this phase, we employ a standard propositional decision tree learner (specifically, C4.5). Applying the decision tree learner to the *join data table* produces a decision tree for the child node. The join data table summarizes the counts of how often instantiations of the children and parents co-occur in the database [20].
3. Convert the decision tree to MLN clauses. Use an MLN parameter learning algorithm to assign weights to the clauses.

Example. The learn-and-join algorithm produces the PBN shown in Figure 1 on a university database. For the *ranking*(S) node, the join data table is given by the relational join of the Student table S with the Professor table P and the RA table RA, followed by selecting (projecting) the attributes *ranking*, *popularity*, *intelligence*. Applying J48 to this table produces the decision tree of Figure 1, which is converted to MLN clauses as shown.

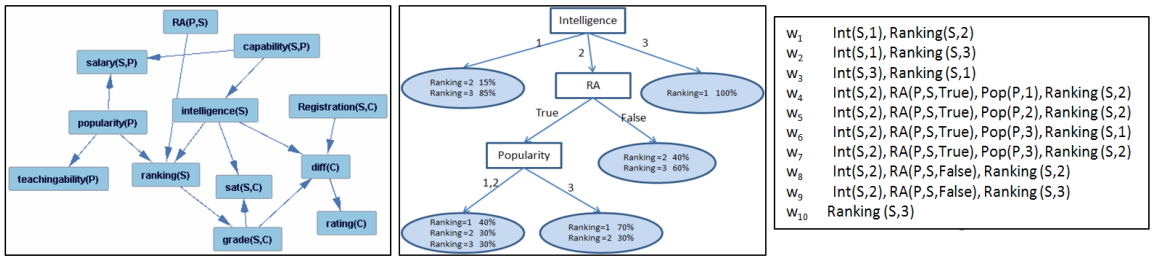

Fig. 1. The figure shows a Parametrized Bayes net, a decision tree for the Ranking node, and the Markov Logic Network clauses obtained from the decision tree.

4 Experimental Design

Datasets. We used 3 benchmark real-world databases. For each dataset, we evaluated the learning methods with a 5-fold cross-validation scheme. We formed 5 subdatabases for each by randomly selecting entities from each entity table and restricted the relationship tuples in each subdatabase to those that involve only the selected entities (cf. [10]). The databases and their main characteristics are as follows. For more details please see the references.

MovieLens Database. The second dataset is the MovieLens dataset from the UC Irvine machine learning repository [10].

Mutagenesis Database. This dataset is widely used in Inductive Logic Programming research. It contains information on Atoms, Molecules, and Bonds between them. We use the modifications of [10].

Hepatitis Database. This data is a modified version of the PKDD02 Discovery Challenge database [4]. The database contains information on the laboratory examinations of hepatitis B and C infected patients.

Comparison Systems. All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. For single table Bayes net search we used GES search [1] with the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; [21]). For decision tree learning, we used the J48 program of the Weka package [7], which implements the C4.5 decision tree algorithm, in the probability estimation setting. We turned off pruning and applied the Laplace correction, which avoids leaves with 0 conditional probabilities, as recommended by [19]. For general MLN learning, we used the Alchemy package [13], Version 30. We compare four MLN learning algorithms.

MBN The structure is learned using the learn-and-join algorithm. The weights of clauses are learned using Alchemy's current default weight learning procedure of [14]. This method is called MBN for "moralized Bayes Net" by Khosravi *et al.* [10].

LAJ+DT The structure is learned using the learn-and-join algorithm, then made more compact by learning decision trees for local independencies . Weight learning is carried out with Alchemy.

LHL LHL is the lifted hypergraph learning algorithm of Kok and Domingos [11].

LSM Learning Structural Motifs [12] uses random walks to identify densely connected objects in data, and groups them and their associated relations into a motif.

Performance Metrics. We use 4 performance metrics: Number of Clauses or Parameters, Runtime, Accuracy (ACC), and Conditional log likelihood (CLL) [15, 11]. ACC and CLL have been used in previous studies of MLN learning. The CLL of a ground atom in a database given an MLN is its log-probability given the MLN and the information in the database. Accuracy is evaluated using the most likely value for a ground atom. For ACC and CLL the values we report are averages over all predicates. We do not use Area under Curve (AUC), as it mainly applies to binary values, and most of our attributes are nonbinary. Like previous studies, we used the MC-SAT inference algorithm [18] to compute a probability estimate for each possible value of a descriptive attribute for a given object or tuple of objects.

	LAJ +DT	MBN	LSM	LHL
MovieLens	39	327	10	NT
Mutagen	50.33	880	13	NT
Hepatitis	120	793	23	27

Table 1. Average number of parameters in learned model.

	LAJ + DT	MBN	LSM	LHL
MovieLens	359	3414	34.03	NT
Mutagen	289.3	4437	26.47	NT
Hepatitis	848	6219	10.94	72452

Table 2. Average induction times in seconds.

	LAJ + DT	MBN	LSM	LHL
MovieLens	0.55 ± 0.04	**0.56 ± 0.04**	0.39 ± 0.04	NT
Mutagen	**0.58 ± 0.064**	0.54 ± 0.074	0.45 ± 0.043	NT
Hepatitis	**0.51 ± 0.04**	**0.51 ± 0.01**	0.3 ±0.01	0.37 ± 0.052

Table 3. The 5-fold cross-validation estimate for the accuracy and standard deviation of predicting the true values of descriptive attributes, averaged over all descriptive attribute instances.

	LAJ +DT	MBN	LSM	LHL
MovieLens	-0.8 ± 0.25	-0.79 ± 0.12	**-0.65 ± 0.10**	NT
Mutagen	**-0.97 ± 0.0134**	-1.31 ± 0.197	-1.01± 0.065	NT
Hepatitis	**-1.16 ±0.04**	-1.74 ± 0.08	-1.36 ±0.03	-2.13 ± 0.011

Table 4. The 5-fold cross-validation estimate for the conditional log-likelihood assigned to the true values of descriptive attributes, averaged over all descriptive attribute instances.

5 Evaluation Results

Number of Parameters/Clauses. Table 1 shows the average number of clauses produced by the MLN models. *Adding decision tree pruning to the learn-and-join algorithm leads to much more compact models*, with improvement ratios in the range of 5-25. The LSM and LHL algorithms learn a very small number of clauses. Inspection of the learned MLNs shows that the rules are mostly just the unit clauses that model marginal probabilities (e.g., *intelligence*(S, I)) [2]. This underfits the data as our measurements of ACC and CLL indicate (Tables 3, 4).

Runtimes. Table 2 shows average runtimes over the subdatabases. The LSM method scales well, as does the learn-and-join structure learning method, even with decision tree learning added. The computational bottleneck for the two learn-and-join methods is the weight optimization that uses the relatively slow Alchemy routines. Since decision tree pruning reduces the number of model parameters, it speeds up parameter estimation by a factor of about 10.

Accuracy. On the real-world datasets, the average accuracy of the learn-and-join methods is quite similar, and *about 10-15% higher than that of LSM.* The accuracy numbers are fairly low overall because many of the descriptive attributes have many possible values (e.g. 9 for Lumo in Mutagenesis). The LSM accuracy variance is low, which together with poor average accuracy is consistent with the hypothesis that LSM underfits the data.

Conditional Log-likelihood. This measure is especially sensitive to the quality of the parameter estimates. The MBN method without decision tree pruning performs clearly worse on 2 of the 3 datasets, both in terms of average CLL and variance. The CLL performance of LSM is acceptable on average. The parameter estimates are biased towards uniform values, which leads to predictions whose magnitudes are not extreme.

6 Summary

Augmenting Bayes net learning with decision tree learning leads to a compact set of Horn clauses that represent statistical patterns in a relational database. Evaluated on three benchmark relational databases, decision tree pruning significantly reduced the number of model parameters, by factors ranging from 5-25. After converting the Horn clauses to Markov Logic Networks, MLN inference can be used to evaluate the predictive accuracy of the resulting models. The predictive performance of the pruned models is competitive with or superior to the unpruned models.

References

1. David Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2003.
2. Pedro Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning*. 2007.
3. Daan Fierens and et al. Logical bayesian networks and their relation to other probabilistic logical models. In *ILP*, 2005.
4. Richard Frank and et al. A method for multi-relational classification using single and multi-feature aggregation functions. In *Proceeding of PKDD*, 2007.
5. Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *NATO ASI on Learning in graphical models*, pages 421–459, 1998.
6. Lise Getoor and Ben Tasker. *Introduction to statistical relational learning*. MIT Press, 2007.
7. Mark Hall and et al. The weka data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.
8. Tuyen N. Huynh and Raymond J. Mooney. Discriminative structure and parameter learning for Markov logic networks. In *ICML*, pages 416–423, 2008.
9. Kristian Kersting and Luc de Raedt. Bayesian logic programming. In *Introduction to Statistical Relational Learning* [6], chapter 10, pages 291–318.
10. Hassan Khosravi and et al. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, pages 487–493, 2010.
11. Stanley Kok and Pedro Domingos. Learning Markov logic network structure via hypergraph lifting. In *ICML*, pages 64–71, 2009.
12. Stanley Kok and Pedro Domingos. Learning Markov logic networks using structural motifs. In *ICML*, pages 551–558, 2010.
13. Stanley Kok and et al. The Alchemy system for statistical relational AI. Technical report, University of Washington., 2009.
14. Daniel Lowd and Pedro Domingos. Efficient weight learning for Markov logic networks. In *PKDD*, pages 200–211, 2007.
15. Lilyana Mihalkova and Raymond J. Mooney. Bottom-up learning of Markov logic network structure. In *ICML*, pages 625–632, 2007.
16. Jennifer Neville and David Jensen. Relational dependency networks. In *An Introduction to Statistical Relational Learning* [6], chapter 8.
17. David Poole. First-order probabilistic inference. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 985–991. Morgan Kaufmann, 2003.
18. Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, 2006.
19. Foster J. Provost and Pedro Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.
20. Oliver Schulte. A tractable pseudo-likelihood function for bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.
21. CMU The Tetrad Group. The Tetrad project, 2008. http://www.phil.cmu.edu/projects/tetrad/.