Fast Parameter Learning for Markov Logic Networks Using Bayes Nets

Hassan Khosravi

School of Computing Science
Simon Fraser University
Vancouver-Burnaby, B.C., Canada
hkhosrav@cs.sfu.ca

Abstract. Markov Logic Networks (MLNs) are a prominent statistical relational model that have been proposed as a unifying framework for statistical relational learning. As part of this unification, their authors proposed methods for converting other statistical relational learners into MLNs. For converting a first order Bayes net into an MLN, it was suggested to moralize the Bayes net to obtain the structure of the MLN and then use the log of the conditional probability table entries to calculate the weight of the clauses. This conversion is exact for converting propositional Markov networks to propositional Bayes nets however, it fails to perform well for the relational case. We theoretically analyze this conversion and introduce new methods of converting a Bayes net into an MLN. An extended imperial evaluation on five datasets indicates that our conversion method outperforms previous methods.

1 Introduction

The field of statistical relational learning (SRL) has developed a number of new statistical models for the induction of probabilistic knowledge that supports accurate predictions for multi-relational structured data [1]. Markov Logic Networks (MLNs) form one of the most prominent SRL model classes; they generalize both first-order logic and Markov network models [2]. Essentially, an MLN is a set of weighted first-order formulas that compactly defines a Markov network comprising ground instances of logical predicates. The formulas are the structure or qualitative component of the Markov network; they represent associations among ground facts. The weights are the parameters or quantitative component; they assign a likelihood to a given relational database by using the log-linear formalism of Markov networks. An open-source benchmark system for MLNs is the Alchemy package [3].

MLNs were proposed as a unifying framework for SRL since they are general enough that many of the other well known SRL models can easily be converted into them. Schulte et al show that it is desirable to use models where learning is performed on First-order Bayes nets and inference is performed on MLNs [4, 5, 6]. Their model combines the scalability and efficiency of model searches in directed models with the inference power and theoretical foundations of undirected models. For converting a first order Bayes net into an MLN, the suggested method has been to moralize the Bayes net

F. Riguzzi and F. Železný (Eds.): ILP 2012, LNAI 7842, pp. 102–115, 2013.

to obtain the structure of the MLN and then use the log of the conditional probability table entries to calculate the weight of the clauses [2].

In propositional data using the log of the conditional probability table entries of a Bayes net to obtain weights in a corresponding Markov random field, results in two predictively equivalent graphical models. Although there is no corresponding result for the case of relational data, Richardson and Domingos propose using the same method which is a plausible candidate [2]. In this paper, we examine this conversion theoretically and experimentally for the first time, and provide rationale for why it fails in the relational case. We propose another conversion method for converting weights from a Bayes net to an MLN that is theoretically justifiable and out-performs the current proposed method.

1.1 Related Work

Most of the work on parameter learning in MLNs is based on ideas developed for Markov networks (undirected graphical models) in the propositional case. Special issues that arise with relational data are discussed by Lowd and Domingos [7]. Most recent methods aim to maximize the regularized weighted pseudo log-likelihood [2, 8], and/or perform a scaled conjugate gradient descent using second-order derivative information [7].

We focus on parameter estimation algorithms to convert already calculated weights from directed models to MLNs, so our method can only be applied to a restricted class of MLN structures that are learned from, or can be converted into a Bayes net. The main motivation for converting the directed model into an undirected model and performing inference with an undirected model is that they do not suffer from the problem of cyclic dependencies in relational data [2, 9, 10]. Early work on this topic required ground graphs to be acyclic [11, 12]. For example, Probabilistic Relational Models allow dependencies that are cyclic at the predicate level as long as the user guarantees acyclicity at the ground level [12]. A recursive dependency of an attribute on itself is shown as a self loop in the model graph. If there is a natural ordering of the ground atoms in the domain (e.g., temporal), there may not be cycles in the ground graph; but this assumption is restrictive in general. The generalized order-search of Ramon *et al.* [13] instead resolves cycles by learning an ordering of ground atoms which complicates the learning procedure. The learn-and-join algorithm of Schulte *et al.* utilizes a pseudo-loglikelihood that measures the fit of a Bayes net to a relational database which is well-defined even in the presence of recursive dependencies [5].

1.2 Background

A **Bayes net structure** [14] is a directed acyclic graph (DAG) G, whose nodes comprise a set of random variables denoted by V. In this paper we consider only discrete finite random variables. When discussing a Bayes net, we refer interchangeably to its nodes or its variables. A Bayes net is a pair $\langle G, \boldsymbol{\theta}_G \rangle$ where $\boldsymbol{\theta}_G$ is a set of parameter values that specify the probability distributions of children conditional on assignments of values

to their parents. We use as our basic model class **Functor Bayes Nets** (FBN)[15][1], a relatively straightforward generalization of Bayes nets, for relational data. Our methods also apply to other directed graphical formalisms.

A **functor** is a function symbol or a predicate symbol. Each functor has a set of values (constants) called the **range** of the functor. A **population** is a set of individuals, corresponding to a domain or type in logic. A functor whose range is $\{T, F\}$ is a **predicate**, usually written with uppercase letters like P, R. A **functor random variable** is of the form $f(t_1, \ldots, t_k)$ where f is a **functor** (either a function symbol or a predicate symbol) and each t_i is a first-order variable or a constant. Each functor has a set of values (constants) called the **range** of the functor. The structure of a Functor Bayes Net consists of: (1) A directed acyclic graph (DAG) whose nodes are parametrized random variables, (2) a population for each first-order variable, and (3) an assignment of a range to each functor. Figure 1 is an example of an FBN.

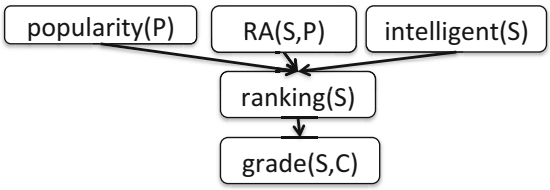

Fig. 1. A Functor Bayes Net which shows that $ranking$ of a student correlates with his $intelligence$, $grades$, and $popularity$ of the professors that he is a research assistant for

Moralization is a technique used to convert a directed acyclic graph (DAG) into undirected models or MLN formulas. To convert a Bayes net into an MLN using moralization, add a formula to the MLN for each assignment of values to a child and its parents [2](Sec. 12.5.3). Thus, an MLN obtained from a BN contains a formula for each CP-table entry. Figure 2 shows an arbitrary conditional probability table and its corresponding clauses for the $ranking$ node in Figure 1.

While the moralization approach produces graph structures that represent the dependencies among predicates well, converting each row of each conditional probability table to an MLN clause leads to a large number of MLN clauses and hence MLN parameters. **Local** or **context-sensitive** independencies are a well-known phenomenon that can be exploited to reduce the number of parameters required in a Bayes net. A **decision tree** can compactly represent conditional probabilities [16]. The nodes in a decision tree for a functor random variable c are themselves functor random variables. An edge that originates in $f(t_1, \ldots, t_k)$ is labeled with one of the possible values in the range of f. The leaves are labeled with probabilities for the different possible values of the c variable. Khosravi et al combine decision tree learning algorithms with Bayes nets to learn a compact set of clauses for relational data [17]. In their experiments, using

[1] Functor Bayes nets were called Parametrized Bayes Nets by Poole. The term "Parametrized" referred to the their semantics, and does not mean that parameters have been assigned for the structure. We modified the name to overcome this confusion.

Pop, RA, Int	r =1	r=2	r=3
1, 1, True	$r_{1,1}$	$r_{1,1}$	$r_{1,1}$
--- , ---, ---	---	---	---
3, 3, False	$r_{1,1}$	$r_{1,1}$	$r_{1,1}$

$w_{1,1}$	Pop(P,1), Int(S,1), RA(P,S,True), Rank(S,1)
$w_{2,1}$	Pop(P,1), Int(S,1), RA(P,S,True), Rank(S,2)
$w_{3,1}$	Pop(P,1), Int(S,1), RA(P,S,True), Rank(S,3)
$w_{1,2}$	Pop(P,1), Int(S,1), RA(P,S,False), Rank(S,1)
....	
$w_{3,18}$	Pop(P,3), Int(S,3), RA(P,S,False), Rank(S,3)

Fig. 2. A conditional probability table for node $ranking$. Assuming that the range of $popularity, intelligence, ranking = \{1, 2, 3\}$ and the range of $RA = \{True, False\}$. A tabular representation requires a total of $3 \times 3 \times 2 \times 3 = 54$ conditional probability parameters. The figure on the right shows the corresponding MLN clauses for this conditional probability table.

the decision tree representation instead of "flat" conditional probability tables, reduced the number of MLN clauses by a factor of 5-25. Figure 3 shows a decision tree and its corresponding MLN clauses for the ranking node in 1.

2 Conversion of Parameters from Functor Bayes Nets to Markov Logic Networks

In this section we focus on the problem of converting the parameters of an FBN with a fixed structure and parameters to an MLN. In the following discussion, fix an FBN B and a child node v with k possible values $v_1, \ldots, v_k$ and an assignment π of values to its parents. Then the conditional probability $p(v_i|\pi)$ is defined in the CP-table or decision tree leafs of v in B. The corresponding MLN contains a formula p_j that expresses that a child node takes on the value v_i and the parents take on the values π. The weight of the formula p_j is denoted as w_j. As a running example, we are interested in predicting the ranking of a student Jack given that we have about the five courses he has taken, and the one professor that he is a research assistant for. We also know that Jack is highly intelligent.

In order to convert the conditional probabilities from a Bayes net into weights for MLNs, the use of the logarithm of the conditional probabilities was suggested by Domingos and Richardson [2]. We refer to this method as LOGPROB. The LOGPROB method sets the weights using the following formula:

$$w_j := log(p(v_i|\pi)).$$

In the propositional case, combining moralization with the log-conditional probabilities as in the LOGPROB method leads to an undirected graphical model that is predictively equivalent to the original directed graphical model [14]. Theoretical support for the LOGPROB method is provided by considering the log-likelihood function for an MLN structure obtained from a Bayes net. The standard log-likelihood for an MLN M [2] is given by

$$L_M(\mathcal{D}) = \sum_j w_j n_j(\mathcal{D}) + ln(Z),$$

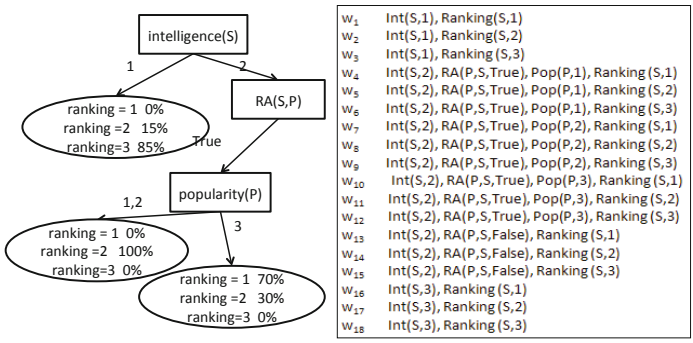

Fig. 3. A decision tree that specifies conditional probabilities for the $Ranking(S)$ node of Figure 1 and the corresponding MLN clauses generated from the decision tree. The number of the clauses has been reduced to 18.

where $n_j(\mathcal{D})$ denotes the number of instances of formula j in database $\mathcal{D}$ and Z is a normalization constant. Omitting the normalization term $ln(Z)$, this is the sum over all child-parent configurations of the corresponding Bayes net in $\log$ scale multiplied by $n_{j,\pi}(\mathcal{D})$, which is the number of instances of the child-parent configuration in the database:

$$L_M(\mathcal{D}) = \sum_i \sum_\pi \log(p(v_i|\pi))n_{i,\pi}(\mathcal{D}).$$

This unnormalized log-likelihood is maximized by using the observed conditional frequencies in the data table. While the normalization constant is required for defining a valid probabilistic inference, it arguably does not contribute to measuring the fit of a parameter setting and hence can be ignored in model selection; the constraint that weights are derived from normalized conditional probabilities in a Bayes net already bounds their range.

Although there is no corresponding result for the case of relational data, the propositional conversion result makes the log-probabilities a plausible candidate for weights in an MLN structure obtained from a 1st-order Bayes net. We analyze the proposed method on both FBNs with and without parameter reduction, and introduce our own conversion method for converting weights from a Bayes net to an MLN. This conversion method is theoretically justifiable and out-performs the current proposed method.

2.1 Conversion in Functor Bayes Nets Without Parameter Reduction

The complication which is introduced by relational data is the existence of different objects and relations which leads to having a diverse number of groundings for different clauses. Using the moralization technique to obtain an MLN from a Bayesian network introduces two types of clauses for each functor $f(t_1, \ldots, t_k)$:

- Clauses that are produced from the conditional probability table of $f(t_1, \ldots, t_k)$.
- Clauses that are produced from the conditional probability table of other functors where $f(t_1, \ldots, t_k)$ is in the set of their parents.

Clause	#Grounds
pop(P,POP), int(S,INT), ra(S,P,RA), ranking(S,RANK)	n1
ranking(S,RANK) grade(S,C,GRADE)	n2

Fig. 4. Clauses that the $ranking$ node participates in. P, S, and C are first order variables for professors, students, and courses. POP, INT, RA, $RANK$, $GRADE$ are constant values in the range of their corresponding functors.

Figure 4 shows the two types of clauses that the $ranking$ node participates in when the FBN of Figure 1 is moralized.

The two main factors that directly influence the inference procedure of a functor are the weights and the number of groundings of the clauses where the functor node is present. This means that clauses with more groundings will have a higher impact on the final predicated value however, the number of groundings of a clause usually correlate with the number of free variables in it which does not always reflect its importance. In fact longer clauses with a diverse set of functors for different objects tend to have many groundings with limited predictive information, but a short clause with only functors related to the object may carry more predictive information but has fewer number of groundings. For example, Jack is a research assistant for only one professor but has taken 5 courses, so there are five times more groundings for the second clause compared to the first one however, it may be the case that the first clause is a better predictor for the $ranking$ node.

The other factor, as discussed, is the weight of the clauses. The LOGPROB method assigns negative weights to all the clauses. A weight distribution that punishes all clauses performs well in the propositional cases because all of the clauses have the same number of groundings however, it fails to achieve good predictive performance on the relational case.

It may seem trivial that normalizing the weight of the clauses with their number of groundings may overcome the problem (i.e. divide the weight of the clause by its number of groundings) but this is not possible as the weight of the clauses are fixed during the learning phase and the number of groundings for each model is determined during the inference phase. Schulte *et al* propose a new log-linear $inference$ method that uses the *geometric mean* rather than the *arithmetic mean* to use this idea.

We propose using a weight conversion, referred to as LOG-LINEAR, that has meaningful interpretation for the weights of the clauses. The LOG-LINEAR method uses the following formula to assign weight to clauses:

$$w_j := log(p(v_i|\pi)) - log(1/k).$$

Weights set using this method are measuring the information gain provided by the parent information π relative to the uniform probability baseline. These weights can be interpreted as usual in a linear model: a positive weight indicates that a predictive factor increases the baseline probability, a negative weight indicates a decreased probability relative to the baseline. A zero weight indicates a condition that is irrelevant in the

sense of not changing the baseline probability. With the LOG-LINEAR transformation some of the formulas receive positive and some negative weights so a clause with many groundings can have a small impact if the weight of the clause is close to zero. Figure 5 shows the two conversion functions.

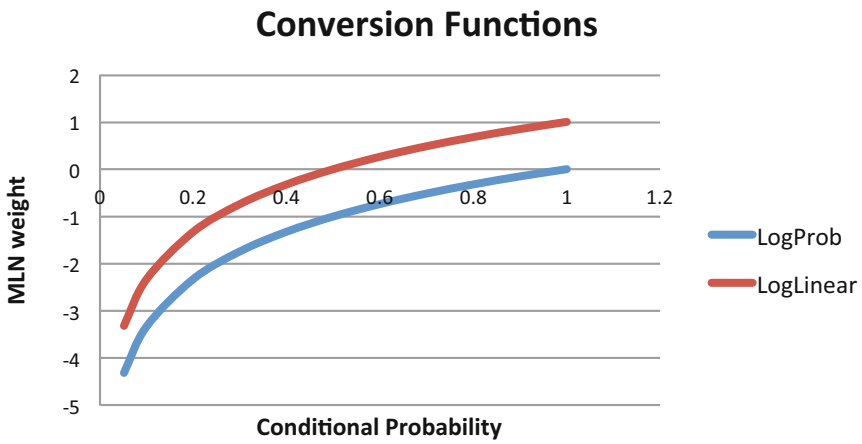

Fig. 5. Comparison of LOGPROB and LOG-LINEAR for a functor with a range of two values. Conditional probabilities smaller than 0.5 have negative weights and conditional probabilities larger than 0.5 have positive weights.

2.2 Conversion in Functor Bayes Nets with Parameter Reduction

The LOGPROB method has short-comings with conversion in FBNs without parameter reduction but still performs reasonably well. In this section we explain why this method performs very poorly when parameter reduction techniques are used on Bayes nets.

Focusing on the inference procedure in MLNs using our running example, the MLN inference model evaluates the likelihood for all possible ranking values, so likelihood of Ranking(Jack, 1), Ranking(Jack, 2), and Ranking(Jack, 3) are calculated and the most likely one will be picked as the ranking for Jack. With the tabular conditional probabilities that have no parameter reduction, all possible combinations of values are converted into MLN clauses. For instance, clauses with weights w_4 -w_{12} are all the different combinations of values assigned to the first clause in Figure 4. The same is also true for the second clause in Figure 4. This is not the case when parameter reduction techniques are used. Figure 6 shows the decision tree for $grade$, and Figure 7 shows the two types of clauses that are used for the $ranking$ node when parameter reduction techniques are used.

Based on these clauses, the likelihood of $Ranking(Jack, 2)$ and $Ranking(Jack, 3)$ will be calculated using one grounding on the first clause, and the likelihood of $Ranking(Jack, 1)$ will be calculated using the grounding on the first clause plus the five groundings on the second clause. Log-probabilities are negative, so using the LOGPROB method means that frequently many negative weights will be added up in evaluating $Ranking(Jack, 1)$ compared to $Ranking(Jack, 2)$ and

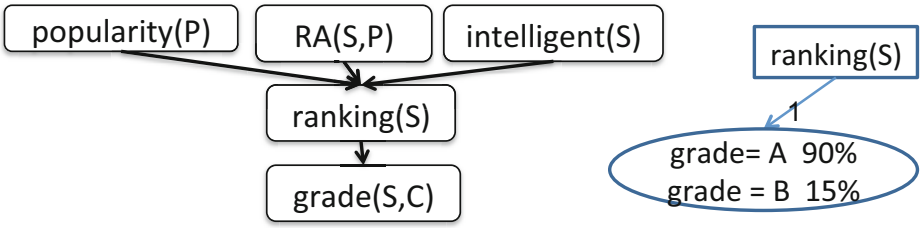

Fig. 6. The decision tree used to store the parameters of $grade$. $Ranking$ node only contributes in this decision tree with value 1, so values 2 and 3 are independent of $grade$

Clause	#Grounds
Pop(P,POP), Int(S,INT), RA(S,P,RA), Rank(S,RANK)	n1
Ranking(S,RANK_1) Grade(S,C,Grade)	n2

Fig. 7. Clauses that the $ranking$ node participates in. P, S, and C are first order variables for professors, students, and courses. POP, INT, RA, $RANK$, $GRADE$ are constant values in the range of their corresponding functors. $RANK_1$ in the second clause indicates that this clause can only be instantiated when Ranking=1.

$Ranking(Jack, 3)$. Thus, LOGPROB induces a bias against values that satisfy formulas with more groundings.

With the LOG-LINEAR transformation, some of the formulas receive positive and some negative weights, so there is no bias against values that are involved in formulas with more groundings. That is, the influences of the different groundings are more balanced against each other.

3 Experimental Design

The main objective of this evaluation is to show that our proposed conversion function performs better than the previously proposed method. The second objective is to show that Moralization methods are very fast and competitive with state-of-the-art MLN learners. We first introduce the datasets used, then the systems compared, and finally the comparison metrics.

3.1 Datasets

We used five benchmark real-world datasets. Table 1 lists the resulting databases and their sizes in terms of total number of tuples and number of ground atoms, which is the input format for Alchemy. Each descriptive attribute is represented as a separate function, so the number of ground atoms is larger than that of tuples.

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

Mutagenesis Database. This dataset is widely used in ILP research [18]. It contains information on Atoms, Molecules, and Bonds among them.

Hepatitis Database. This data is a modified version of the PKDD02 Discovery Challenge database, following [19]. The database contains information on the laboratory examinations of hepatitis B and C infected patients.

Mondial Database. This dataset contains data from multiple geographical web data sources. We follow the modification of [20], and use a subset of the tables and features. Our dataset includes a self-relationship table *Borders* that relates two countries.

UW-CSE Database. This dataset lists facts about the Department of Computer Science and Engineering at the University of Washington (UW-CSE) (e.g., Student, Professor) and their relationships (i.e. AdvisedBy, Publication). The dataset was obtained by crawling pages on the department's Website (www.cs.washington.edu).

Table 1. Size of datasets in total number of table tuples and ground atoms

Dataset	#tuples	#Ground atoms
Movielens	82623	170143
Mutagenesis	15218	35973
Hepatitis	12447	71597
Mondial	814	3366
UW-CSE	2099	3380

3.2 Comparison Systems and Performance Metrics

Structure learning. We fix the structure for all the methods to evaluate just the parameters. We use two different structure learning methods to evaluate the parameter learning methods both with and without parameter reduction. We used the **MBN** [10] to get tabular representation without parameter reduction and **MBN-DT** [17] to get a sparse structure with decision trees. Both methods use GES search [21] and the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; [22]).

Our experiments compare the two conversion functions as well as state-of-the-art MLN learning methods.

LOGPROB. Weight learning is carried out using log-probabilities as suggested by Richardson and Domingos. Parameters are given by Tetrad's maximum likelihood estimation method and the LOGPROB conversion.

LOG-LINEAR. weight learning is the same as the LOGPROB method but we use our proposed conversion method.

MLN. Weight learning is carried out using the procedure of Lowd and Domingos [7, 23] , implemented in Alchemy.

LSM. Learning Structural Motifs [8] uses random walks to identify densely connected objects in the data, and groups them and their associated relations into a motif. We input the structure of the learn-and-join algorithms to LSM. Running LSM's structure learning algorithm tries to prune the structure.

We report measurements on runtime, accuracy, and conditional log-likelihood (CLL). To define accuracy, we apply MLN inference to predict the probability of an attribute value, and score the prediction as correct if the most probable value is the true one. For example, to predict the gender of person Bob, we apply MLN inference to the atoms gender(Bob, male) and gender(Bob, female). The result is correct if the predicted probability of gender(Bob, male) is greater than that of gender(Bob, female). The conditional log-likelihood (CLL) of a ground atom in a database $\mathcal{D}$ given an MLN is its log-probability given the MLN and $\mathcal{D}$ [2]. The CLL directly measures how precise the estimated probabilities are.The values we report are averages over all attribute predicates.

Infernce. We use the MC-SAT inference algorithm [24] implemented in Alchemy to compute a probability estimate for each possible value of a descriptive attribute for a given object or tuple of objects.

We also compare our model with discriminative learning methods from Inductive Logic Programming.

4 Evaluation Results

In the following section we discuss the run time and then the accuracy of the models. We investigate the predictive performance by doing five-fold cross validation on the given datasets. All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM.

4.1 Run Times

Table 2 shows the time taken in seconds for learning the parameters for Markov Logic Networks using the structures generated by MBN and MBN-DT. The time for the conversion methods is basically the same, namely the time required to compute the database statistics for the entries. For the purposes of discussing runtime, we group LOGPROB

Table 2. The time taken in seconds for parameter learning. we group LOGPROB and LOG-LINEAR methods and call it Log/Lin in this table.

Structure Learning	MBN			MBN-DT		
Parameter Learning	Log/Lin	MLN	LSM	Log/Lin	MLN	LSM
UW-CSE	2	5	80	3	3	8
Mondial	3	90	260	3	15	26
MovieLens	8	10800	14300	9	1800	2100
Mutagenesis	3	9000	58000	4	600	1200
Hepatitis	3	23000	34200	5	4000	5000

and LOG-LINEAR methods and call it Log/Lin in this table. The runtime improvements are orders of magnitude faster than previous methods. Results from extending the moralization approach to parameter learning, provide strong evidence that the moralization approach leverages the scalability of relational databases for data management and Bayes nets learning to achieve scalable MLN learning on databases of realistic size for both structure and parameter learning.

4.2 Accuracy

Tables 3 and 4 show the accuracy results using the MBN and MBN-DT respectively. Higher numbers indicate better performance. For the MBN structure, LSM clearly performs worse. The LOGPROB method performs reasonably well and the results are acceptable.The LOG-LINEAR and MLN methods are competitive and out-perform the other two methods. We will call LOG-LINEAR superior to MLN since learning in LOG-LINEAR is approximately two orders of magnitude faster

For the MBN-DT structure, the LOGPROB method performs very poorly as discussed previously. Since the prediction is mainly based on the number of groundings for clauses, the performance is similar to randomly assigning values for predicates. The LOG-LINEAR and MLN methods are competitive, but MLN tends to do slightly better.

4.3 Conditional Log-Likelihood

Tables 5 and 6 show the predicted average log-likelihood of each fact in the database for the MBN and MBN-DT structure. This measure is especially sensitive to the quality of the parameter estimates. Smaller negative numbers indicate better performance.

The performance on CLL is very similar to accuracy. LOG-LINEAR method does much better than LOGPROB on both MBN and MBN-DT structure. Both MLN and LSM method minimizes CLL in their learning procedure which explains their great performance.

4.4 Comparison with Inductive Logic Programming on Mutagenesis

Table 3 and 4 show results on generative learning where averages over all predicates are reported. In this section we compare the performance of the LOGPROB and LOG-LINEAR algorithm for a classification task, discriminative learning, to predict the mutagenicity of chemical compounds. The class attribute is the mutagenicity. Compounds recorded as having positive mutagenicity are labeled active (positive examples) and compounds recoreded as having 0 or negative mutagenicity are labeled inactive (negative examples). The database contains a total of 188 compounds.

This problem has been extensively studied in Inductive Logic Programming (ILP). The purpose of this comparison is to benchmark the predictive performance of the moralization approach, using generative learning, against discriminative learning by methods that are different from Markov Logic Network learners. Table 7 presents the results of Lodhi and Muggleton [25]. For the STILL system, we followed the creators' evaluation methodology of using a randomly chosen training and test set. The other

Table 3. The 5-fold cross-validation estimate using MBN structure learning for the accuracy of predicting the true values of descriptive attributes, averaged over all descriptive attribute instances. Observed standard deviations are shown.

	LOGPROB	LOG-LINEAR	MLN	LSM
UW-CSE	0.72 ± 0.083	0.76 ± 0.022	0.75 ± 0.028	0.64 ± 0.086
Mondial	0.40 ± 0.060	0.41 ± 0.045	0.44 ± 0.050	0.32 ± 0.042
Movielens	0.64 ± 0.006	0.64 ± 0.006	0.60 ± 0.029	0.57 ± 0.016
Mutagenesis	0.55 ± 0.139	0.64 ± 0.025	0.61 ± 0.022	0.64 ± 0.029
Hepatitis	0.49 ± 0.033	0.50 ± 0.037	0.51 ± 0.025	0.30 ± 0.028

Table 4. The 5-fold cross-validation estimate using MBN-DT structure learning for the accuracy of predicting the true values of descriptive attributes, averaged over all descriptive attribute instances. Observed standard deviations are shown.

	LOGPROB	LOG-LINEAR	MLN	LSM
UW-CSE	0.06 ± 0.088	0.73 ± 0.166	0.75 ± 0.086	0.65 ± 0.076
Mondial	0.18 ± 0.036	0.43 ± 0.027	0.44 ± 0.033	0.31 ± 0.024
Movielens	0.26 ± 0.017	0.62 ± 0.026	0.62 ± 0.023	0.59 ± 0.051
Mutagenesis	0.21 ± 0.021	0.61 ± 0.023	0.60 ± 0.027	0.61 ± 0.025
Hepatitis	0.19 ± 0.024	0.48 ± 0.032	0.50 ± 0.021	0.40 ± 0.032

Table 5. The 5-fold cross-validation estimate using MBN for the conditional log-likelihood assigned to the true values of descriptive attributes, averaged over all descriptive attribute instances. Observed standard deviations are shown.

	LOGPROB	LOG-LINEAR	MLN	LSM
UW-CSE	-0.45 ± 0.122	-0.37 ± 0.090	-0.40 ± 0.151	-0.42 ± 0.058
Mondial	-2.01 ± 0.721	-1.53 ± 0.287	-1.28 ± 0.149	-1.30 ± 0.055
Movielens	-2.10 ± 0.205	-2.10 ± 0.205	-0.79 ± 0.338	-0.76 ± 0.124
Mutagenesis	-0.87 ± 0.040	-0.83 ± 0.047	-0.92 ± 0.125	-0.92 ± 0.039
Hepatitis	-2.11 ± 1.335	-2.03 ± 1.401	-1.31 ± 0.053	-1.26 ± 0.215

Table 6. The 5-fold cross-validation estimate using MBN-DT for the conditional log-likelihood assigned to the true values of descriptive attributes, averaged over all descriptive attribute instances. Observed standard deviations are shown.

	LOGPROB	LOG-LINEAR	MLN	LSM
UW-CSE	-1.24 ± 0.035	-0.76 ± 0.124	-0.46 ± 0.104	-0.54 ± 0.046
Mondial	-1.43 ± 0.071	-1.19 ± 0.101	-1.24 ± 0.083	-1.29 ± 0.071
Movielens	-2.63 ± 0.027	-1.27 ± 0.096	-0.85 ± 0.030	-0.93 ± 0.071
Mutagenesis	-1.26 ± 0.098	-0.95 ± 0.046	-0.93 ± 0.050	-0.92 ± 0.130
Hepatitis	-1.55 ± 0.547	-1.50 ± 0.039	-1.13 ± 0.052	-1.20 ± 0.048

Table 7. A comparison of the LOG-LINEAR method with standard Inductive Logic Programming systems trained to predict mutagenicity. Although Bayes net learning produces a generative model, its performance is competitive with discriminative learners.

Method	Evaluation	Accuracy	Reference
MBN-LOG-LINEAR	10-fold	0.87	
MBNDT-LOG-LINEAR	10-fold	0.87	
P-progol	10-fold	0.88	[18]
FOIL	10-fold	0.867	[26]
STILL	90%train-10%test	0.936	[27]
MBN-LOG-LINEAR	90%train-10%test	0.944	
MBNDTLOG-LINEAR	90%train-10%test	0.944	

systems are evaluated using 10-fold cross-validation. The table shows that the classification performance of the generative Moralized Bayes net model matches that of the discriminative Inductive Logic Programming models.

5 Conclusion and Future Work

The moralization approach combines Bayes net learning, one of the most successful machine learning techniques, with Markov Logic networks, one of the most successful statistical-relational formalisms. Previous work applied the moralization method to learning MLN structure; in this paper we extended it to learning MLN parameters. We motivated and empirically investigated a new method for converting Bayes net parameters to MLN weights. For converting a first order Bayes net into an MLN, it was suggested to moralize the Bayes net to obtain the structure of the MLN and then use the log of the conditional probability table entries to calculate the weight of the clauses. This conversion is exact for converting propositional Markov networks to propositional Bayes Nets; however, it fails to perform well for the relational case. We theoretically analyze this conversion and introduce new methods of converting a Bayes net into an MLN.

Our evaluation on five medium-size benchmark databases with descriptive attributes indicates that compared to previous MLN learning methods, the moralization parameter learning approach improves the scalability and run-time performance by at least two orders of magnitude. Predictive accuracy is competitive or even superior.

References

[1] Getoor, L., Tasker, B.: Introduction to statistical relational learning. MIT Press (2007)
[2] Domingos, P., Richardson, M.: Markov logic: A unifying framework for statistical relational learning. In: [1]
[3] Kok, S., Summer, M., Richardson, M., Singla, P., Poon, H., Lowd, D., Wang, J., Domingos, P.: The Alchemy system for statistical relational AI. Technical report, University of Washington, Version 30 (2009)
[4] Schulte, O., Khosravi, H.: Learning graphical models for relational data via lattice search. Machine Learning, 41 pages (2012) (to appear)

[5] Schulte, O., Khosravi, H.: Learning directed relational models with recursive dependencies. Machine Learning (2012) (Forthcoming. Extended Abstract)
[6] Khosravi, H., Schulte, O., Hu, J., Gao, T.: Learning compact markov logic networks with decision trees. Machine Learning (2012) (Forthcoming. Extended Abstract. Acceptance Rate?)
[7] Lowd, D., Domingos, P.: Efficient weight learning for Markov logic networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 200–211. Springer, Heidelberg (2007)
[8] Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: ICML, pp. 551–558 (2010)
[9] Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: UAI, pp. 485–492 (2002)
[10] Khosravi, H., Schulte, O., Man, T., Xu, X., Bina, B.: Structure learning for Markov logic networks with many descriptive attributes. In: AAAI, pp. 487–493 (2010)
[11] Kersting, K., de Raedt, L.: Bayesian logic programming: Theory and tool. In: [1], ch. 10, pp. 291–318
[12] Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: IJCAI, pp. 1300–1309. Springer (1999)
[13] Ramon, J., Croonenborghs, T., Fierens, D., Blockeel, H., Bruynooghe, M.: Generalized ordering-search for learning directed probabilistic logical models. Machine Learning 70, 169–188 (2008)
[14] Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann (1988)
[15] Poole, D.: First-order probabilistic inference. In: IJCAI, pp. 985–991 (2003)
[16] Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in bayesian networks. In: UAI, pp. 115–123 (1996)
[17] Khosravi, H., Schulte, O., Hu, J., Gao, T.: Learning compact markov logic networks with decision trees. In: Muggleton, S.H., Tamaddoni-Nezhad, A., Lisi, F.A. (eds.) ILP 2011. LNCS, vol. 7207, pp. 20–25. Springer, Heidelberg (2012)
[18] Srinivasan, A., Muggleton, S., Sternberg, M., King, R.: Theories for mutagenicity: A study in first-order and feature-based induction. Artificial Intelligence 85, 277–299 (1996)
[19] Frank, R., Moser, F., Ester, M.: A method for multi-relational classification using single and multi-feature aggregation functions. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 430–437. Springer, Heidelberg (2007)
[20] She, R., Wang, K., Xu, Y.: Pushing feature selection ahead of join. In: SIAM SDM (2005)
[21] Chickering, D.: Optimal structure identification with greedy search. Journal of Machine Learning Research 3, 507–554 (2003)
[22] The Tetrad Group: The Tetrad project (2008), `http://www.phil.cmu.edu/projects/tetrad/`
[23] Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for Artificial Intelligence. Morgan and Claypool Publishers (2009)
[24] Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies. In: AAAI (2006)
[25] Lodhi, H., Muggleton, S.: Is mutagenesis still challenging? In: Inductive Logic Programming, pp. 35–40 (2005)
[26] Quinlan, J.: Boosting first-order learning. In: Arikawa, S., Sharma, A.K. (eds.) ALT 1996. LNCS, vol. 1160, pp. 143–155. Springer, Heidelberg (1996)
[27] Sebag, M., Rouveirol, C.: Tractable induction and classification in first order logic via stochastic matching. In: IJCAI, pp. 888–893 (1997)