Structure Learning for Markov Logic Networks with Many Descriptive Attributes

Hassan Khosravi and **Oliver Schulte*** and **Tong Man** and **Xiaoyuan Xu** and **Bahareh Bina**
hkhosrav@cs.sfu.ca, oschulte@cs.sfu.ca, mantong01@gmail.com,
xiaoyuanxu08@gmail.com, bba18@cs.sfu.ca
School of Computing Science
Simon Fraser University
Vancouver-Burnaby, Canada

Abstract

Many machine learning applications that involve relational databases incorporate first-order logic and probability. Markov Logic Networks (MLNs) are a prominent statistical relational model that consist of weighted first order clauses. Many of the current state-of-the-art algorithms for learning MLNs have focused on relatively small datasets with few descriptive attributes, where predicates are mostly binary and the main task is usually prediction of links between entities. This paper addresses what is in a sense a complementary problem: learning the structure of an MLN that models the distribution of discrete descriptive attributes on medium to large datasets, given the links between entities in a relational database. Descriptive attributes are usually nonbinary and can be very informative, but they increase the search space of possible candidate clauses. We present an efficient new algorithm for learning a directed relational model (parametrized Bayes net), which produces an MLN structure via a standard moralization procedure for converting directed models to undirected models. Learning MLN structure in this way is 200-1000 times faster and scores substantially higher in predictive accuracy than benchmark algorithms on three relational databases.

Introduction

The field of statistical-relational learning (SRL) has developed a number of new statistical models for relational databases (Getoor and Tasker 2007). Markov Logic Networks (MLNs) form one of the most prominent SRL model classes; they generalize both first-order logic and Markov network models (Domingos and Richardson 2007). MLNs have achieved impressive performance on a variety of SRL tasks. Because they are based on undirected graphical models, they avoid the difficulties with cycles that arise in directed SRL models (Neville and Jensen 2007; Domingos and Richardson 2007; Taskar, Abbeel, and Koller 2002). An open-source benchmark system for MLNs is the Alchemy package (Kok et al. 2009). This paper addresses structure learning for MLNs in relational schemas that feature a significant number of descriptive attributes, compared to the number of relationships.

*Supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada.

Our approach is to learn a (parametrized) Bayes net (BN) that defines connections between *predicates* or *functions* (Poole 2003). Associations between predicates constitute a smaller model space than clauses that can be searched more efficiently (cf. (Kersting and de Raedt 2007, 10.7)). Our algorithm learns a model of the joint distribution of descriptive attributes given the links between entities in a database, which is in a sense the complementary problem to link prediction. Applying a standard moralization technique to the BN produces an MLN that we refer to as a moralized Bayes net (MBN).

The main technical contribution of the paper is a new algorithm for learning the structure of a parametrized BN. The algorithm upgrades a single table BN learner, which can be chosen by the user, by decomposing the learning problem for the entire database into learning problems for smaller tables. The basic idea is to apply the BN learner repeatedly to tables and join tables from the database, and combine the resulting graphs into a single graphical model for the entire database. As the computational cost of the merge step is negligible, the run-time of this learn-and-join algorithm is essentially determined by the cost of applying the BN learner to each (join) table separately. Thus our algorithm leverages the scalability of single-table BN learning to achieve scalability for MLN structure learning.

We evaluated the MBN structures using one synthetic dataset and two public domain datasets (MovieLens and Mutagenesis). The learn-and-join algorithm constructs an MLN in reasonable time in cases where Alchemy produces no result. When both methods terminate, we observe a speedup factor of 200-1000. Using standard prediction metrics for MLNs, we found in empirical tests that the predictive accuracy of the moralized BN structures was substantially greater than that of the MLNs found by Alchemy. The difficulties we observed for current MLN methods are not surprising as it has been noted that these methods are not suitable for clauses that include function terms (Mihalkova and Mooney 2007, Sec.8), which are the most natural representation for descriptive attributes. Our code and datasets are available for anonymous ftp download from ftp://ftp.fas.sfu.ca/pub/cs/oschulte/aaai2010.

Paper Organization. We review related work, then parametrized BNs and MLNs. Then we present the learn-

and-join algorithm for parametrized BNs. We compare the moralized Bayes net approach to standard MLN structure learning methods implemented in the Alchemy system, both in terms of processing speed and in terms of model accuracy.

Related Work

Nonrelational structure learning methods. Schmidt *et al.* (2008) compare and contrast structure learning algorithms in directed and undirected graphical methods for nonrelational data, and evaluate them for learning classifiers. Tillman *et al.* (2008) provide the ION algorithm for merging graph structures learned on different datasets with overlapping variables into a single partially oriented graph. It is similar to the learn-and-join algorithm in that it extends a generic single-table BN learner to produce a BN model for a set of data tables. One difference is that the ION algorithm is not tailored towards relational structures. Another is that the learn-and-join algorithm does not analyze different data tables completely independently and merge the result afterwards. Rather, it recursively constrains the BN search applied to join tables with the adjacencies found in BN search applied to the respective joined tables.

MLN structure learning methods. Current methods (Kok and Domingos 2009; Mihalkova and Mooney 2007; Huynh and Mooney 2008; Biba, Ferilli, and Esposito 2008) successfully learn MLN models for binary predicates (e.g., link prediction), but do not scale well to larger datasets with descriptive attributes that are numerous and/or have a large domain of values. One reason for this is that the main approaches so far have been derived from Inductive Logic Programming techniques that search the space of clauses, which define connections between *atoms*. Descriptive attributes introduce a high number of atoms, one for each combination of attribute and value, and therefore define a large search space of clauses. We utilize BN learning methods that search the space of links between predicates/functions, rather than atoms.

Mihalkova and Mooney (2007) distinguish between top-down approaches, that follow a generate-and-test strategy of evaluating clauses against the data, and bottom-up approaches that use the training data and relational pathfinding to construct candidate conjunctions for clauses. Kok and Domingos (2009) add constant clustering and hypergraph lifting to refine relational path-finding. Biba *et al.* (2008) propose stochastic local search methods for the space of candidate clauses. Our approach applies a single BN learner as a subroutine. In principle, the BN learning module may follow a top-down or a bottom-up approach; in practice, most BN learners use a top-down approach. The BUSL algorithm (Mihalkova and Mooney 2007) employs a single-table Markov network learner as a subroutine. The Markov network learner is applied once after a candidate set of conjunctions and a data matrix has been constructed. In contrast, we apply the single-table BN learner repeatedly, where results from earlier applications constrain results of later applications.

Directed graphical models for SRL. The syntax of other directed SRL models, such as Probabilistic Relational Models (PRMs) (Getoor et al. 2007) and Bayes Logic Programs (BLPs) (Kersting and de Raedt 2007), is similar to that of Parametrized Bayes Nets (Poole 2003). Two key issues for directed SRL models are the following.

(1) The directed model represents generic statistical relationships found in the database. In the terminology of Halpern (1990) and Bacchus (1990), the model represents type 1 probabilities or domain frequencies. For instance, a PBN may encode the probability that a student is highly intelligent given the properties of a single course they haven taken. But the database may contain information about many courses the student has taken, which needs to be combined; we may refer to this as the *combining problem*. To address the combining problem, one needs to use an aggregate function, as in PRMs, or a combining rule as in BLPs, or the log-linear formula of MLNs, as we do in this paper. In PRMs and BLPs, the aggregate functions respectively combining rules add complexity to structure learning.

(2) Directed SRL models face the *cyclicity problem*: there may be cyclic dependencies between the properties of individual entities. For example, if there is generally a correlation between the smoking habits of friends, then we may have a situation where the smoking of Jane predicts the smoking of Jack, which predicts the smoking of Cecile, which predicts the smoking of Jane, where Jack, Jane, and Cecile are all friends with each other. In the presence of such cycles, neither aggregate functions nor combining rules lead to well-defined probabilistic predictions. The difficulty of the cyclicity problem has led Neville and Jensen to conclude that "the acyclicity constraints of directed models severely limit their applicability to relational data" (2007, p.241) (see also (Domingos and Richardson 2007; Taskar, Abbeel, and Koller 2002)). Converting the Bayes net to an undirected model avoids the cyclicity problem. Thus the approach of this paper combines advantages of both directed and undirected SRL models: Efficiency and interpretability from directed models, with the solutions to the combining and cyclicity problems from undirected models.

Background and Notation

Parametrized Bayes nets are a basic SRL model; we follow the original presentation of Poole (2003). A **population** is a set of individuals, corresponding to a domain or type in logic. A parametrized 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. An assignment of the form $f(t_1, \ldots, t_k) = a$, where a is a constant in the range of f is an **atom**; if all terms t_i are constants, the assignment is a **ground** atom. A **parametrized Bayes net structure** consists of

(1) a directed acyclic graph (DAG) whose nodes are parametrized random variables.

(2) a population for each first-order variable.

(3) an assignment of a range to each functor.

The functor syntax is rich enough to represent an entity-relation schema (Ullman 1982) via the following translation: Entity sets correspond to populations, descriptive attributes

$Student(\underline{student\_id}, intelligence, ranking)$
$Course(\underline{course\_id}, difficulty, rating)$
$Professor\ (\underline{professor\_id}, teaching\_ability, popularity)$
$Registered\ (\underline{student\_id}, \underline{course\_id}, grade, satisfaction)$
$RA\ (\underline{student\_id}, \underline{prof\_id}, salary, capability)$

Table 1: A relational schema for a university domain. Key fields are underlined.

to function symbols, relationship tables to predicate symbols, and foreign key constraints to type constraints on the first-order variables. Table 1 shows a university relational schema and Figure 1 a parametrized Bayes net structure for this schema. A **table join** of two or more tables contains the rows in the Cartesian products of the tables whose values match on common fields.

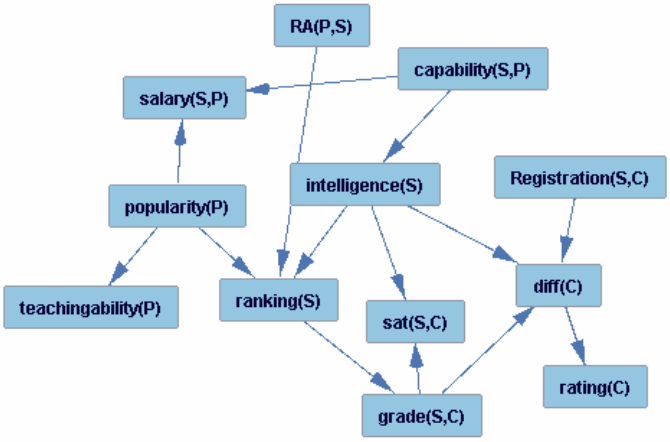

Figure 1: A parametrized BN graph for the relational schema of Table 1.

Markov Logic Networks are presented in detail by Domingos and Richardson (2007). The set of first-order formulas is defined by closing the set of atoms under Boolean combinations and quantification. A **grounding** of a formula p is a substitution of each variable X in p with a constant from the population of X. A database instance $\mathcal{D}$ determines the truth value of each ground formula. The ratio of the number of groundings that satisfy p, over the number of possible groundings, is called the **database frequency** of p.

The qualitative component or structure of an MLN is a finite set of formulas or clauses $\{p_i\}$, and its quantitative component is a set of weights $\{w_i\}$, one for each formula. Given an MLN and a database $\mathcal{D}$, let n_i be the number of groundings that satisfy p_i in $\mathcal{D}$. An MLN assigns a log-likelihood to a database according to the equation

$$ln(P(\mathcal{D})) = \sum_i w_i n_i - ln(Z) \qquad (1)$$

where Z is a normalization constant.

Bayes net DAGs can be converted into MLN structures through the standard **moralization** method (Domingos and Richardson 2007, 12.5.3): connect all spouses that share a common child, and make all edges in the resulting graph undirected. In the moralized BN, a child forms a clique with its parents. For each assignment of values to a child and its parents, add a formula to the MLN.

The Learn-And-Join Method for PBN Structure Learning

We describe our learning algorithm for parametrized Bayes net structures that takes as input a database $\mathcal{D}$. The algorithm is based on a general schema for upgrading a propositional BN learner to a statistical relational learner. By "upgrading" we mean that the propositional learner is used as a function call or module in the body of our algorithm. We require that the propositional learner takes as input, in addition to a single table of cases, also a set of *edge constraints* that specify required and forbidden directed edges. Our approach is to "learn and join": we apply the BN learner to single tables and combine the results successively into larger graphs corresponding to larger table joins. Algorithm 1 gives pseudocode.

We assume that a parametrized BN contains a default set of nodes as follows: (1) one node for each descriptive attribute, of both entities and relationships, (2) one Boolean node for each relationship table. For each type of entity, we introduce one first-order variable. There are data patterns that require more than one variable per entity type to express, notably *relational autocorrelation*, where the value of an attribute may depend on other related instances of the same attribute. For instance, there may be a correlation between the smoking of a person and that of her friends. We model such cases by introducing an additional entity variable for the entity type with corresponding functors for the descriptive attributes, with the following constraints. (1) One of the entity variables is selected as the main variable; the others are auxilliary variables. (2) A functor whose argument is an auxilliary variable has no edges pointing into it. Thus the auxilliary variable serves only to model recursive relationships. For instance the smoking example can be modelled with a JBN structure

$$Friend(X, Y) \rightarrow Smokes(X) \leftarrow Smokes(Y)$$

where X is the main variable of type person and Y is an auxilliary variable.

We list the phases of the algorithm, including two final phases to convert the BN into an MLN. Figure 2 illustrates the increasingly large joins built up in Phases 1–3.

(1) *Analyze single tables.* Learn a BN structure for the descriptive attributes of each entity table E of the database separately (with primary keys removed).

(2) *Analyze single join tables.* Each relationship table R is considered. The input table for the BN learner is the join of R with the entity tables linked by a foreign key constraint (with primary keys removed). Edges between attributes from the same entity table E are constrained to agree with the structure learned for E in phase (1).

3) *Analyze double join tables.* The input tables from the second phase are joined in pairs (if they share a common foreign key) to form the input tables for the BN learner. Edges

between variables considered in phases (1) and (2) are constrained to agree with the structures previously learned.

(4) *Satisfy slot chain constraints.* For each link $A \rightarrow B$ in G, where A and B are functors that correspond to attributes from different tables, arrows from Boolean relationship variables into B are added if required to satisfy the following constraints: (1) A and B share a variable among their arguments, or (2) the parents of B contain a chain of foreign key links connecting A and B.

(5) *Moralize to obtain an MLN structure.*

(6) *Parameter estimation.* Apply an MLN weight learning procedure.

In phases (2) and (3), if there is more than one first-order variable for a given entity type (autocorrelation), then the single-table BN learner is constrained such that functor nodes whose arguments are auxilliary variables have no edges pointing into them.

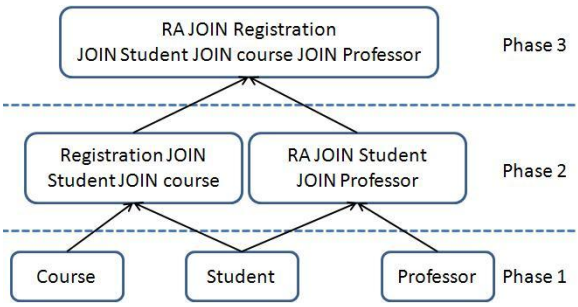

Figure 2: To illustrate the learn-and-join algorithm. A given BN learner is applied to each table and join tables. The presence or absence of edges at lower levels is inherited at higher levels.

Discussion. For fast computation, Algorithm 1 considers foreign key chains of length up to 2. In principle, it can process longer chains. As the algorithm does not assume a bound on the degree of the BN graph, it may produce MLN clauses that are arbitrarily long.

An iterative deepening approach that prioritizes dependencies among attributes from more closely related tables is a common design in SRL. The new feature of the learn-and-join algorithm is the constraint that a BN for a table join must respect the links found for the joined tables. This edge-inheritance constraint reduces the search complexity considerably. For instance, suppose that two entity tables contain k descriptive attributes each. Then in an unconstrained join with a relationship table, the search space of possible adjacencies has size $\binom{2k}{2}$, whereas with the constraint, the search space size is $k^2/2$, which is smaller than $\binom{2k}{2}$ because the quadratic k^2 factor has a smaller coefficient. For example, with $k = 6$, we have $\binom{2k}{2} = 66$ and $k^2/2 = 18$. For the learn-and-join algorithm, the main computational challenge in scaling to larger table joins is therefore not the increasing number of columns (attributes) in the join, but only the increasing number of rows (tuples).

Algorithm 1 Pseudocode for MBN structure learning

Input: Database $\mathcal{D}$ with $E_1, ..E_e$ entity tables, $R_1, ...R_r$ Relationship tables,
Output: MLN for $\mathcal{D}$
Calls: PBN: Any propositional Bayes net learner that accepts edge constraints and a single table of cases as input. WL: Weight learner in MLN
Notation: PBN$(T, \text{Econstraints})$ denotes the output DAG of PBN. Get-Constraints(G) specifies a new set of edge constraints, namely that all edges in G are required, and edges missing between variables in G are forbidden.

```
Add descriptive attributes of all entity and relationship tables as variables to G. Add a boolean indicator for each relationship table to G.
Econstraints = ∅ [Required and Forbidden edges]
for m=1 to e do
    Econstraints += Get-Constraints(PBN(E_m , ∅))
end for
for m=1 to r do
    N_m := join of R_m and entity tables linked to R_m
    Econstraints += Get-Constraints(PBN(N_m, Econstraints))
end for
for all N_i and N_j with a foreign key in common do
    K_ij := join of N_i and N_j
    Econstraints += Get-Constraints(PBN(K_ij, Econstraints))
end for
for all possible combination of values of a node and its parents do
    Add a clause with predicates to MLN input file
end for
Run WL on the MLN file
```

In addition to efficiency, a statistical motivation for the edge-inheritance constraint is that the marginal distribution of descriptive attributes may be different in an entity table than in a relationship table. For instance, if a highly intelligent student s has taken 10 courses, there will be at least ten satisfying groundings of the conjunction $Registered(S, C), intelligence(S) = hi$. If highly intelligent students tend to take more courses than less intelligent ones, then in the $Registered$ table, the frequency of tuples with intelligent students is higher than in the general student population. In terms of the Halpern-Bacchus database domain frequencies, the distribution of database frequencies conditional on a relationship being true may be different from its unconditional distribution (Halpern 1990; Bacchus 1990). The fact that marginal statistics about an entity type E may differ between the entity table for E and a relationship table (or a join table) is another reason why the learn-and-join algorithm constrains edges between attributes of E to be determined only by the result of applying the BN learner to the E table. This ensures that the subgraph of the final parametrized Bayes net whose nodes correspond to the attributes of the E table is exactly the same as the graph that the single-table BN learner constructs for the E table.

After moralization the log-linear formalism of MLNs can be used to derive predictions about the properties of individual entities from a parametrized Bayes net. The next section evaluates the accuracy of these predictions on three

Dataset	#tuples	#Ground atoms
University	171	513
Movielens	82623	170143
MovieLens1 (subsample)	1468	3485
MovieLens2 (subsample)	12714	27134
Mutagenesis	15218	35973
Mutagenesis1 (subsample)	3375	5868
Mutagenesis2 (subsample)	5675	9058

Table 2: Size of datasets in total number of table tuples and ground atoms. Each descriptive attribute is represented as a separate function, so the number of ground atoms is larger than that of tuples.

databases.

Evaluation

All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. For single table BN search we used GES search (Chickering 2003) with the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; (The Tetrad Group 2008)). Our code and datasets are available for anonymous ftp download from ftp://ftp.fas.sfu.ca/pub/cs/oschulte/aaai2010.

Datasets

We used one synthetic and two benchmark real-world datasets. Because the Alchemy systems returned no result on the real-world datasets, we formed two subdatabases for each by randomly selecting entities for each dataset. We restricted the relationship tuples in each subdatabase to those that involve only the selected entities. Table 2 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.

University Database. We manually created a small dataset, based on the schema given in Table 1. The entity tables contain 38 students, 10 courses, and 6 Professors. The $Registered$ table has 92 rows and the RA table has 25 rows.

MovieLens Database. The second dataset is the MovieLens dataset from the UC Irvine machine learning repository. It contains two entity tables: $User$ with 941 tuples and $Item$ with 1,682 tuples, and one relationship table $Rated$ with 80,000 ratings. The $User$ table has 3 descriptive attributes $age, gender, occupation$. We discretized the attribute age into three bins with equal frequency. The table $Item$ represents information about the movies. It has 17 Boolean attributes that indicate the genres of a given movie. We performed a preliminary data analysis and omitted genres that have only weak correlations with the rating or user attributes, leaving a total of three genres.

Mutagenesis Database. This dataset is widely used in ILP research (Srinivasan et al. 1996). Mutagenesis has two entity tables, $Atom$ with 3 descriptive attributes, and $Mole$, with 5 descriptive attributes, including two attributes that are discretized into ten values each (logp and lumo). It features two relationships $MoleAtom$ indicating which atoms are parts of which molecules, and $Bond$ which relates two atoms and has 1 descriptive attribute. Representing a relationship between entities from the same table in a parametrized BN requires using two or more variables associated with the same population (e.g., $Bond(A_1, A_2)$). We also tested our method on the Financial dataset with similar results, but omit a discussion due to space constraints.

Comparison Systems and Performance Metrics

We refer to the output of our method—Algorithm 1—as MBNs. Weight learning is carried out with Alchemy's default procedure, which currently uses the method of Kok and Domingos (2005). We compared MBN learning with 4 previous MLN structure learning methods implemented in different versions of Alchemy.

1. MSL is the structure learning method described by Kok and Domingos (2005).
2. MSLC is the MSL algorithm applied to a reformated input file that adds a predicate for each constant of each descriptive attribute to the .db input file.
3. LHL is the state-of-the-art lifted hypergraph learning algorithm of Kok and Domingos (2009).
4. LHLC is the LHL algorithm applied to the reformated input file.

Data reformatting was used by Kok and Domingos (2007). To illustrate, for instance the predicate $Salary$ with values(high, medium, low), is represented with a single binary predicate $Salary(Student, Salary\_Type)$ in the standard Alchemy input format. The reformatted file contains instead three unary predicates $Salary_{high}(Student)$, $Salary_{med}(Student)$, $Salary_{low}(Student)$. The effect is that the arguments to all predicates are primary keys, which tends to yield better results with Alchemy. To evaluate predictions, the MLNs learned with the reformatted data were converted to the format of the original database.

We use 4 main performance metrics: Runtime, Accuracy(ACC), Conditional log likelihood(CLL), and Area under Curve(AUC). ACC is the percentage of the descriptive attribute values that are correctly predicted by the MLN. 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 $\mathcal{D}$. The CLL directly measures how precise the estimated probabilities are. The AUC curves were computed by changing the CLL threshold above which a ground atom is predicted true (10 tresholds were used). The AUC is insensitive to a large number of true negatives. For ACC and CLL the values we report are averages over all predicates. For AUC, it is the average over all predicates that correspond to descriptive attributes with binary values (e.g. gender). CLL and AUC have been used in previous studies of MLN learning (Mihalkova and Mooney 2007; Kok and Domingos 2009). As in previous studies, we used the MC-SAT inference algorithm (Poon and Domingos 2006) to compute a probability estimate for each possible value of a descriptive attribute for a given object or tuple of objects.

Dataset	MBN	MSL	MSLC	LHL	LHLC
University	0.03 + 0.032	5.02	11.44	3.54	19.29
MovieLens	1.2 +120	NT	NT	NT	NT
MovieLens1	0.05 + 0.33	44	121.5	34.52	126.16
MovieLens2	0.12 + 5.10	2760	1286	3349	NT
Mutagenesis	0.5 +NT	NT	NT	NT	NT
Mutagenesis1	0.1 + 5	3360	900	3960	1233
Mutagenesis2	0.2 +12	NT	3120	NT	NT

Table 3: Runtime to produce a parametrized MLN, in minutes. The MBN column shows structure learning time + weight learning time.

Runtime Comparison

Table 3 shows the time taken in minutes for learning in each dataset. The Alchemy times include both structure and parameter learning. For the MBN approach, we report both the BN structure learning time and the time required for the subsequent parameter learning carried out by Alchemy.

Structure Learning. The learn-and-join algorithm returns an MLN structure very quickly (under 2 minutes). This includes single-table BN parameter estimation as a subroutine.

Structure and Weight Learning. The total runtime for the MBN approach is dominated by the time required by Alchemy to find a parametrization for the moralized BN. On the smaller databases, this takes between 5-12 minutes. On MovieLens, parametrization takes two hours, and on Mutagenesis, it does not terminate. While finding optimal parameters for the MBN structures remains challenging, the combined structure+weight learning system is much faster than the overall structure + weight learning time of the Alchemy methods: They do not scale to the complete datasets, and for the subdatabases, the MBN approach is faster by a factor ranging from 200 to 1000. As reported by Kok and Domingos (2009), the runtimes reported on other smaller databases for other MLN learners are also much larger (e.g., BUSL takes about 13 hr on the UW-CSE dataset with 2,112 ground atoms). These results are strong evidence that the MBN approach leverages the scalability of Bayes net learning to achieve scalable MLN learning on databases of realistic size.

Predictive Accuracy and Data Fit

Previous work on MLN evaluation has used a "leave-one-out" approach that learns MLNs for a number of subdatabases with a small subset omitted (Mihalkova and Mooney 2007). This is not feasible in our setting because even on a training set of size about 15% of the original, finding an MLN structure using Alchemy is barely possible. Given these computational constraints, we investigated the predictive performance by learning an MLN on one randomly selected 2/3 of the subdatabases as a training set, testing predictions on the remaining 1/3. While this does not provide strong evidence about the generalization performance in absolute terms, it gives information about the relative performance of the methods. Table 4 reports the average ACC, CLL, and AUC of each dataset. Higher numbers indicate better performance and NT indicates that the system was not able to return an MLN for the dataset, either crashing or timing out after 4 days of running. MBN achieved substantially better predictions on all test sets, in the range of 10-20% for accuracy.

Where the learning methods return a result on a database, we also measured the predictions of the different MLN models for the facts in the training database. This indicates how well the MLN summarizes the statistical patterns in the data. These measurements test the power of using the log-linear equation (1) to derive instance-level type 2 predictions from a BN model of generic type 1 frequencies. While a small improvement in predictive accuracy may be due to overfitting, the very large improvements we observe are evidence that the MLN models produced by the Alchemy methods underfit and fail to represent statistically significant dependencies in the data. The ground atom predictions of the MBNs are always better, e.g. by at least 20% for accuracy.

Conclusion and Future Work

This paper considered the task of building a statistical-relational model for databases with many descriptive attributes. We combined Bayes net learning, one of the most successful machine learning techniques, with Markov Logic networks, one of the most successful statistical-relational formalisms. The main algorithmic contribution is an efficient new structure learning algorithm for a parametrized Bayes net that models the joint frequency distribution over attributes in the database, given the links between entities. Moralizing the BN leads to an MLN structure. Our evaluation on two benchmark databases with descriptive attributes shows that compared to current MLN structure learning methods, the approach using moralization improves the scalability and run-time performance by orders of magnitude. With standard parameter estimation algorithms and prediction metrics, the moralized MLN structures make substantially more accurate predictions.

An important direction for future work are pruning methods that take advantage of local independencies. Each CP-table row in the parametrized Bayes net becomes a formula in the moralized Bayes net. Often there are local independencies that allow CP tables to be represented more compactly. SRL research has used decision trees as an effective compact representation for CP-tables (Getoor, Taskar, and Koller 2001). This suggests that combining decision trees with parametrized Bayes nets will lead to more parsimonious MLN structures. Another possibility is to use MLN weight estimation procedures to prune uninformative clauses. Huynh and Mooney (2008) show that L1-regularization can be an effective method for pruning a large set of MLN clauses.

References

Bacchus, F. 1990. *Representing and reasoning with probabilistic knowledge: a logical approach to probabilities*. Cambridge, MA, USA: MIT Press.

Biba, M.; Ferilli, S.; and Esposito, F. 2008. Structure learning of Markov logic networks through iterated local search. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N. M., eds., *ECAI*, 361–365.

Dataset	Accuracy					CLL					AUC				
	MBN	MSL	MSLc	LHL	LHLc	MBN	MSL	MSLc	LHL	LHLc	MBN	MSL	MSLc	LHL	LHLc
Movielens11	**0.63**	0.39	0.45	0.42	0.50	**-0.99**	-3.97	-3.55	-4.14	-3.22	**0.64**	0.46	0.60	0.49	0.55
Movielens12	**0.59**	0.42	0.46	0.41	0.47	**-1.15**	-3.69	- 3.56	-3.68	-3.08	**0.62**	0.47	0.54	0.50	0.55
Mutagenesis1	**0.60**	0.34	0.47	0.33	0.45	**-2.44**	-4.97	- 3.59	-4.02	-3.12	**0.69**	0.56	0.56	0.50	0.53
Mutagenesis2	**0.68**	NT	0.53	NT	NT	**-2.36**	NT	- 3.65	NT	NT	**0.73**	NT	0.59	NT	NT

Table 4: The table shows predictive performance for our MBN method and structure learning methods implemented in Alchemy. We trained on 2/3 of the database and tested on the other 1/3.

Dataset	Accuracy					CLL					AUC				
	MBN	MSL	MSLc	LHL	LHLc	MBN	MSL	MSLc	LHL	LHLc	MBN	MSL	MSLc	LHL	LHLc
University	**0.85**	0.37	0.51	0.37	0.55	**-0.4**	-5.79	-3.24	-5.91	-2.66	**0.88**	0.45	0.68	0.52	0.70
Movielens1	**0.67**	0.43	0.43	0.42	0.49	**-0.75**	-4.09	-2.83	-4.09	-3.42	**0.70**	0.46	0.53	0.50	0.51
Movielens2	**0.65**	0.42	0.42	0.43	0.48	**-1**	-3.55	- 3.94	-3.38	-3.82	**0.69**	0.49	0.51	0.50	0.52
Movielens	0.69	NT	NT	NT	NT	- 0.70	NT	NT	NT	NT	0.73	NT	NT	NT	NT
Mutagenesis1	**0.81**	0.36	0.55	0.33	0.52	**- 0.6**	-4.70	- 3.38	-4.33	-3.03	**0.90**	0.56	0.60	0.52	0.53
Mutagenesis2	**0.81**	NT	0.35	NT	NT	**-0.60**	NT	- 4.65	NT	NT	**0.90**	NT	0.56	NT	NT

Table 5: The table shows predictive performance represent the training error measured on the database our MBN method and two Alchemy structure where inference is performed over the training dataset.

Chickering, D. 2003. Optimal structure identification with greedy search. *Journal of Machine Learning Research* 3:507–554.

Domingos, P., and Richardson, M. 2007. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* (2007).

Getoor, L., and Tasker, B. 2007. *Introduction to statistical relational learning*. MIT Press.

Getoor, L.; Friedman, N.; Koller, D.; Pfeffer, A.; and Taskar, B. 2007. Probabilistic relational models. In *Introduction to Statistical Relational Learning* (2007). chapter 5, 129–173.

Getoor, L.; Taskar, B.; and Koller, D. 2001. Selectivity estimation using probabilistic models. *ACM SIGMOD Record* 30(2):461–472.

Halpern, J. Y. 1990. An analysis of first-order logics of probability. *Artificial Intelligence* 46(3):311–350.

Huynh, T. N., and Mooney, R. J. 2008. Discriminative structure and parameter learning for markov logic networks. In Cohen, W. W.; McCallum, A.; and Roweis, S. T., eds., *ICML*, 416–423. ACM.

Kersting, K., and de Raedt, L. 2007. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* (2007). chapter 10, 291–318.

Kok, S., and Domingos, P. 2005. Learning the structure of Markov logic networks. In Raedt, L. D., and Wrobel, S., eds., *ICML*, 441–448. ACM.

Kok, S., and Domingos, P. 2007. Statistical predicate invention. In *ICML*, 433–440. ACM.

Kok, S., and Domingos, P. 2009. Learning markov logic network structure via hypergraph lifting. In Danyluk, A. P.; Bottou, L.; and Littman, M. L., eds., *ICML*, 64–71. ACM.

Kok, S.; Summer, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Wang, J.; and Domingos, P. 2009. The Alchemy system for statistical relational AI. Technical report, University of Washington.

Mihalkova, L., and Mooney, R. J. 2007. Bottom-up learning of Markov logic network structure. In *ICML*, 625–632. ACM.

Neville, J., and Jensen, D. 2007. Relational dependency networks. In *An Introduction to Statistical Relational Learning* (2007). chapter 8.

Poole, D. 2003. First-order probabilistic inference. In Gottlob, G., and Walsh, T., eds., *IJCAI*, 985–991. Morgan Kaufmann.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*. AAAI Press.

Schmidt, M.; Murphy, K.; Fung, G.; and Rosales, R. 2008. Structure learning in random fields for heart motion abnormality detection. In *CVPR*. IEEE Computer Society.

Srinivasan, A.; Muggleton, S.; Sternberg, M.; and King, R. 1996. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence* 85(1-2):277–299.

Taskar, B.; Abbeel, P.; and Koller, D. 2002. Discriminative probabilistic models for relational data. In Darwiche, A., and Friedman, N., eds., *UAI*, 485–492. Morgan Kaufmann.

The Tetrad Group, Department of Philosophy, C. 2008. The Tetrad project: Causal models and statistical data. http://www.phil.cmu.edu/projects/tetrad/.

Tillman, R. E.; Danks, D.; and Glymour, C. 2008. Integrating locally learned causal structures with overlapping variables. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *NIPS*, 1665–1672. MIT Press.

Ullman, J. D. 1982. *Principles of database systems*. 2. Computer Science Press.