Machine Learning manuscript No.
(will be inserted by the editor)

Learning Compact Markov Logic Networks With Decision Trees

Hassan Khosravi and Oliver Schulte ·
Jianfeng Hu · Tianxiang Gao
{hkhosrav, oschulte, jhfu, tga18}@sfu.ca

Abstract Statistical-relational learning combines logical syntax with probabilistic methods. Markov Logic Networks (MLNs) are a prominent model class that generalizes both first-order logic and undirected graphical models (Markov networks). The qualitative component of an MLN is a set of clauses and the quantitative component is a set of clause weights. Generative MLNs model the joint distribution of relationships and attributes. A state-of-the-art structure learning method is the *moralization approach*: learn a set of directed Horn clauses, then convert them to conjunctions to obtain MLN clauses. The directed clauses are learned using Bayes net methods. The moralization approach takes advantage of the high-quality inference algorithms for MLNs and their ability to handle cyclic dependencies. A weakness of moralization is that it leads to an unnecessarily large number of clauses. In this paper we show that using decision trees to represent conditional probabilities in the Bayes net is an effective remedy that leads to much more compact MLN structures. In experiments on benchmark datasets, the decision trees reduce the number of clauses in the moralized MLN by a factor of 5-25, depending on the dataset. The accuracy of predictions is competitive with the models obtained by standard moralization, and in many cases superior.

1 Introduction: Moralization for Relational Data

As relational data are very common in practice, an important goal is to extend machine learning techniques for them. Several prominent statistical-relational formalisms combine logic programming clauses with the statistical interpretation of graphical models [1–4]. These generative models represent probabilistic patterns over both links/relationships and attributes. To illustrate the connection between graphical and logical formalisms for directed graphical models, consider the conditional probability parameters of a Bayes net (BN), which are of the form $P(\mathit{child\_value}|\mathit{parent\_values}) = p$. These can be translated into Horn clauses of

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

the form *child_value* $\leftarrow$ *parent_values*; p where the head is the assignment of a value to a child node, and the body specifies an assignment of values to the parent nodes. Ngo and Haddawy refer to such clauses as p-sentences [5]. In this view, the qualitative component of a Bayes net is a set of Horn clauses, and the quantitative component is a set of conditional probabilities, one for the head of each clause. For undirected models, the qualitative component of a Markov Logic Network (MLN) is a set of first-order formulas, and the quantitative component is a set of weights, one for each clause. Domingos and Richardson show how an MLN can be interpreted as a template for a Markov random field whose nodes comprise ground atoms that instantiate the first-order formulas [4]. MLNs have achieved impressive performance on a variety of relational learning tasks. An open-source benchmark system for MLNs is the Alchemy package [6].

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, the standard Bayes net product formula no longer provides a valid basis for probabilistic inference. 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" [7, p.241]. Several researchers advocate the use of undirected rather than directed models because cycles do not arise with the former [4, 8].

Structure Learning via Moralization. The recently introduced moralization approach [9] can be seen as a hybrid method that uses directed models for learning and undirected models for inference. This method learns a directed first-order Bayes net model for an input relational database. The Bayes net structure is then converted to an MLN set of clauses using the moralization method, described by Domingos and Richardson [4, 12.5.3]. In graphical terms, moralization connects all co-parents that share a child, then omits edge directions. In logical terms, moralization converts the (probabilistic) Horn clauses defined by a Bayes net to conjunctions of literals. Converting the Bayes net to an undirected model avoids the cyclicity problem. The learn-and-join algorithm of Khosravi *et al* upgrades propositional Bayes net learning to relational data in a very efficient way [9]. Compared to predecessor MLN learning algorithms on several benchmark datasets, structure learning was orders of magnitude faster. Moreover, the predictive performance of the moralized Bayes net models using MLN inference methods was substantially more accurate.

Because there are many attributes (predicates) in the databases, and most of them have three possible values or more, there are many conditional probability parameters in the Bayes net. A disadvantage of the moralization approach is that it adds a clause for each conditional probability parameter, which produces a relatively large number of clauses. 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, parameter estimates are less accurate. This paper presents an extension of the moralization approach that produces significantly smaller MLN structures without sacrificing statistical power.

Decision Trees for Representing Local Independencies. As discussed by Kersting and deRaedt [2, 10.7]), a key factor for efficiently learning a graphical relational model is to search for associations between functions or *predicates*, rather than for associations between function/predicate values or *literals*. For instance, an algorithm may search for an association between the GPA of a student and the difficulty of a course she has taken, rather than an association between the literals (GPA = high) and (difficulty = high). It is well-known that because Bayes net graphs represent associations between random variables, rather than between specific values of these variables, they may fail to capture *local* or *context-sensitive* independencies that hold conditional on specific values of the random variables [10, 11]. In the relational setting, this means that when Bayes net graphs represent associations between functions/predicates, they may not capture local independencies among literals. Thus while model search in the predicate space has efficiency advantages, it has the disadvantage of missing context-sensitive independencies. A common way to represent context-sensitive independencies is by augmenting the Bayes net with decision trees: instead of keeping a conditional probability table for each node of the Bayes net, learn a decision tree that predicts the probability of a child node value given values for its parents [10–12, 3]. Such trees are also called probability estimation trees. The main advantages of decision trees for relational models are as follows. (i) Many methods have been developed for learning decision trees that produce probability estimates [13–16]. (ii) Each tree branch corresponds to a conjunction of literals and is straightforwardly converted to an MLN clause. We refer to this conversion as **context-sensitive moralization**. Figure 1 illustrates the system architecture for context-sensitive moralization.

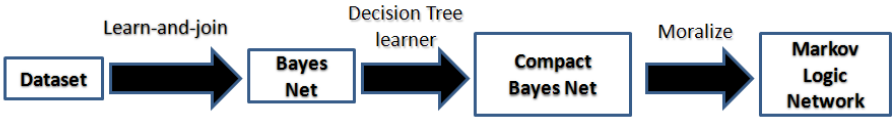

Fig. 1 System Architecture for context-sensitive moralization: learning a compact Markov Logic Network from an input relational database.

Learning Decision Trees for Local Independencies. Given a fixed Bayes net structure, the conditional distribution of a child node v given a value assignment to its parent nodes can be learned from local statistics that involve only the family of v, which comprises v together with its parents. Our approach is to apply a standard propositional decision tree learner, which can be chosen by the user, to a data table that represents the local family statistics from the input observed database. Since propositional decision tree learners are applied "as is", our approach leverages the speed of propositional decision tree learning to achieve fast relational learning. The data table is formed by a relational join that involves only links and attributes that appear in the child node or its parents. In logical terms, the table is constructed from the tuples (groundings) that satisfy an assignment of values to the family of v. The resulting decision tree compactly models the conditional frequency, in the input database, of a child node value given an assignment of values to its parents.

Evaluation. We compared our learning algorithms with several state-of-the-art methods using public domain datasets (MovieLens, Mutagenesis, Mondial, Hepatitis). Decision tree pruning is fast and very effective in reducing the number of MLN clauses, by a factor of 5-25 depending on the dataset. The comparison with the unpruned moralized models and with LSM [17] and LHL [18],state-of-the-art MLN structure learning methods, indicates that predictive accuracy with decision trees is competitive and in many cases superior.

Limitations. We outline limitations of our current system. These limitations are not fundamental to context-sensitive moralization; we leave extensions that address them for future work. The main limitation of our current algorithm is that it does not find associations between links, for instance that if a professor advises a student, then they are likely to be coauthors. In the terminology of Probabilistic Relational Models [1], our model addresses attribute uncertainty, but not existence uncertainty (concerning the existence of links). A related limitation is that the algorithm does not consider associations between attributes conditional on the *absence* of a relationship.

Another limitation is that we do not propose a new weight learning method, so we use standard Markov Logic Network methods for parameter learning after the structure has been learned. While these methods find good parameter settings, they are slow.

In this paper we first learn a Bayes net structure and then use decision tree learning to obtain a more compact parametrization of the fixed net structure. Friedman and Goldszmidt [11] provide evidence that integrating decision tree learning with the Bayes net graph search leads to more accurate Bayes net models than applying decision tree learning after model search.

Contributions. The main contribution of the paper is to show that decision tree learning algorithms can be combined with Bayes nets to learn a compact set of clauses for relational data. The method is compared to other Markov Logic Network structure learning methods on 4 benchmark databases.

Paper Organization. We review related work, then background and notation. We show how the learn-and-join moralization algorithm can be combined with probability estimation trees. Different structure learning algorithms are compared with and without decision tree pruning on four relational databases, in terms of processing speed, model complexity, and model accuracy.

2 Additional Related Work

Bayes nets and Decision Trees. For nonrelational data, the use of decision trees has been long established to reduce the number of parameters after a Bayes net structure has been learned [10], [11, Sec.1]. Friedman and Goldszmidt use minimum description length as an objective function for Bayes net+decision tree learning [11, 19], which is motivated by the goal of obtaining a compact representation of conditional probabilities. In their work on propositional data, as in ours on relational data, the important feature of decision trees is their capacity for information compression, rather than their discriminatory power for classification.

We use Parametrized Bayes Nets [20] 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, several previous statistical-relational formalisms use decision trees for compact representation of conditional probabilities. Getoor, Taskar and Koller used decision trees to augment Statistical-Relational Models [12]. The join-based syntax and semantics of Statistical-Relational Models are different from the logic-based syntax and template grounding semantics of Parametrized Bayes nets and MLNs. Logical Bayesian Networks [3] use decision trees to represent conditional probability parameters. The main difference with our use of decision trees is that the decision tree branches are interpreted as existentially quantified conjunctions of literals as in Tilde [21], which is different from the grounding semantics of MLN formulas.

Relational Dependency Networks. Dependency Networks (DNs) were introduced by Heckerman *et al.* as a graphical model that combines aspects of both Bayes nets and Markov nets [22]. Dependency networks approximate a joint distribution as the product of conditional probabilities of each node given its Markov blanket (which renders the node conditionally independent of all others). In contrast to DNs, the parameters of Bayes nets are conditional probabilities for a node given its parents, which do *not* render a node independent of all others (except for nodes without children). Another difference between BNs and DNs is that the acyclicity constraint of BNs implies the existence of a topological ordering of the nodes, whereas nodes in a DN are unordered. For further discussion of the relationships between BNs, MNs and DNs see [22].

As a solution to the cyclicity problem, Neville and Jensen proposed upgrading Dependency Networks for relational data [23]. The differences between BNs and DNs in the propositional case carry over to Parametrized Bayes nets and Relational Dependency Networks [24]. As a consequence, propositional Bayes net learning algorithms cannot be applied for learning Relational Dependency Networks. Instead, learning algorithms for Relational Dependency Networks have been based on learning independent conditional probability models for each node. In particular, Neville and Jensen use relational probability trees for learning conditional probabilities in Relational Dependency Networks [23]; these decision trees require the specification of aggregate functions. Natarajan et al. propose the use of functional gradient boosting to learn such trees [25]. Functional gradient boosting has also been used to learn relational regression trees that are converted to MLN clauses [26], similar to our moralization approach. We compare relational regression tree learning with Bayes net+decision tree learning in Section 5.3 below, after we have presented the details of our algorithm.

Kok and Domingos [17] emphasize the importance of learning long clauses for relational models. In principle, the moralization approach can learn arbitrarily long clauses. In our simulations, the moralization method produces substantially longer clauses than the MLN comparison learners.

3 Background Concepts

Our work combines concepts from relational databases, graphical models, and Markov Logic networks. As much as possible, we use standard notation from these

different areas. We begin with logical and relational concepts, then add terminology from graphical models.

3.1 Functors and Relational Schemas

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 functor whose range is $\{T, F\}$ is a **predicate**, usually written with uppercase letters like P, R. A **parametrized random variable** is of the form $f(\tau_1, \ldots, \tau_k)$ where f is a functor and each term τ_i is a first-order variable or a constant. We also refer to parametrized random variables as **functor nodes**, or for short **fnodes**.[1] Unless the functor structure matters, we refer to a functor node simply as a node. An assignment of the form $f(\tau_1, \ldots, \tau_k) = a$, where a is a constant in the range of f, is a **literal** [28]. In predicate notation a literal is written as $f(\tau_1, \ldots, \tau_k, a)$. A **formula** is formed by Boolean combinations of literals; we refer to a conjunction of literals simply as a **conjunction**. Thus conjunctions are equivalent to assignments of values to functor nodes. We follow the notation of Prolog and of statistics and write $l_1, l_2, \ldots, l_n$ for the conjunction $l_1 \wedge l_2 \wedge \cdots \wedge l_n$ of literals.

A **population** is a set of individuals, corresponding to a domain or type in logic. Each first-order variable X is associated with a population. An **instantiation** or **grounding** for a set of variables $X_1, \ldots, X_k$ assigns a constant c_i from the population of X_i to each variable X_i. The **database frequency** of a formula in a relational database $\mathcal{D}$ is the number of instantiations of the population variables in the functor nodes that satisfy the assignment in the database, divided by the number of all possible instantiations. We denote this empirical joint distribution by $P_{\mathcal{D}}$. The conditional frequency $P_{\mathcal{D}}(\phi_1|\phi_2)$, where ϕ_1, ϕ_2 are formulas, is defined in terms of joint conjunction frequencies in the usual way as $P_{\mathcal{D}}(\phi_1|\phi_2) \equiv P_{\mathcal{D}}(\phi_1 \wedge \phi_2)/P_{\mathcal{D}}(\phi_2)$ [29].

Getoor and Grant discuss the applications of function concepts for statistical-relational modelling in detail [30]. The functor formalism is rich enough to represent the constraints of an entity-relationship (ER) schema [31] via the following translation: Entity sets correspond to populations, descriptive attributes to functions, relationship tables to predicates, and foreign key constraints to type constraints on the arguments of relationship predicates. A **table join** of two or more tables contains the rows in the Cartesian products of the tables whose values match on common fields. A table join corresponds to a conjunction [31].

3.2 Graphical Models for Relational Data: Parametrized Bayes Nets and Markov Logic Networks.

A **Bayes net structure** [32] is a directed acyclic graph G, whose nodes comprise a set of random variables denoted by V. The **family** of a node in a Bayes net comprises the node together with its parents. In this paper we consider only discrete finite random variables. When discussing a Bayes net, we refer interchangeably

[1] The term "functor" is used as in Prolog [27]. Functor nodes go by different names in different contexts: In Prolog, the equivalent of a functor node is called a "structure", in Bayes Logic Programs, a "Bayesian atom" [2], and simply "atom" by Chiang and Poole [28].

to its nodes or its variables. A Bayes net (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. Often the conditional probabilities are specified in a **conditional probability table**. Parametrized Bayes Nets form a basic SRL model class introduced by Poole [20]. A **Parametrized Bayes Net** (PBN) is a Bayes net whose nodes are functor nodes.

The qualitative component or structure of a **Markov Logic Network** (MLN) is a finite set of first-order formulas or clauses $\{\phi_i\}$, and its quantitative component is a set of weights $\{w_i\}$, one for each clause [4]. Below we follow the Alchemy system and write MLN formulas using predicate notation rather than functor notation. MLN inference uses a log-linear model to assign a probability to each possible database (interpretation). The log-likelihood of a database is the weighted sum of the number of satisfying groundings for each clause, plus a database-independent normalization constant.

Moralization converts a directed acyclic graph into an undirected model. To convert a PBN into an MLN using moralization, add a clause to the MLN for each assignment of values to a child and its parents [4, Sec. 12.5.3]. The MLN for a moralized Bayes net B thus contains a clause for *each* conditional probability in B [4].

3.3 Examples

Table 1 shows a university relational schema. Figure 2 shows a small relational database instance for this schema for illustration.

Student(*Name*, *intelligence*, *ranking*)
Course(*Number*, *difficulty*, *rating*)
Professor (*Name*, *teaching_ability*, *popularity*)
Registration (*S.name*, *C.number*, *grade*, *satisfaction*)
RA (*S.name*, *P.name*, *salary*, *capability*)

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

In the schema of Table 1 the attribute *ranking* of the *Student* table can be represented as a functor node $ranking(S)$ where S is a first-order variable ranging over the population of students. The range of this functor node comprises 3 values, 1, 2, 3 that represent different ranking levels. The *Registration* relationship can be represented as a Boolean functor node, or predicate, $Registered(S, C)$, whose range is is $\{T, F\}$. The functor node $grade(S, C)$ represents the grade of a student in a course, which is a descriptive attribute associated with a registration link. Table 2 shows the database frequencies of various conjunctions. An example of a database conditional probability is

$$P_{\mathcal{D}}(ranking(S) = 1 | RA(S, P) = T, popularity(P) = 3) = \frac{2/9}{2/9} = 1.$$

Course			
Number	Prof	rating	difficulty
101	Oliver	3	1
102	David	2	2
103	Oliver	3	2

Student		
Name	intelligence	ranking
Jack	3	1
Kim	2	1
Paul	1	2

Professor		
Name	popularity	teaching Ability
Oliver	3	1
Jim	2	1

Registration			
S.name	C.number	grade	satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Kim	103	A	1
Paul	101	B	1
Paul	102	C	2

RA			
S.Name	P.Name	salary	capability
Jack	Oliver	High	High
Kim	Oliver	Med	Med
Kim	Jim	Low	Med

Fig. 2 A simple relational database instance for the relational schema of Table 1.

Table 2 To illustrate the computation of the frequency of a conjunction formula in the database example of Figure 2.

Formula	#groundings	Frequency $P_{\mathcal{D}}$
$ranking(S) = 1$	2	$2/3$
$ranking(S) = 1, intelligence(S) = 2$	1	$1/3$
$RA(S,P) = T$	3	$3/(3 \cdot 3)$
$RA(S,P) = T, popularity(P) = 3$	2	$2/9$

Figure 3 shows a Parametrized Bayes net for the schema of Table 1, where each descriptive attribute and each relationship table (link type) is represented as a functor node. The Bayes net was learned from an expanded version of the database in Figure 2 using the learn-and-join algorithm [9] (see Section 5.1). Figure 4 illustrates the moralization process.

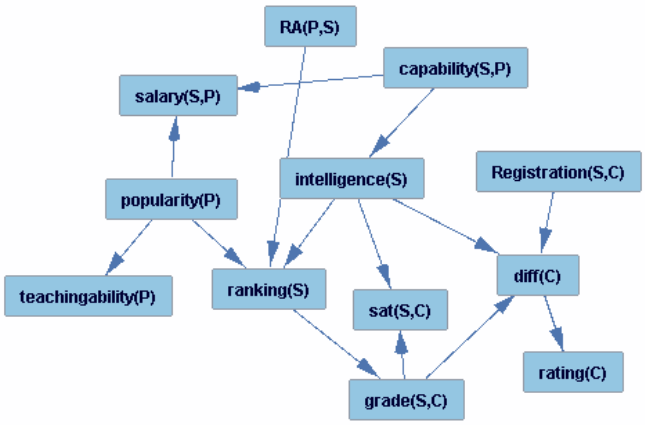

Fig. 3 A Parametrized Bayes net graph for the relational schema of Table 1.

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

$w_{1,1}$	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. 4 The figure on the left shows a conditional probability table for the functor node $ranking(S)$ in the Parametrized Bayes net of Figure 3. We use obvious abbreviations for functors. The range of $popularity, intelligence, ranking$ is $\{1, 2, 3\}$ and the range of $RA = \{True, False\}$. A tabular representation therefore requires a total of $3 \times 3 \times 2 \times 3 = 54$ conditional probability parameters. The figure on the right illustrates the corresponding 54 clauses, one for each row in the conditional probability table.

4 Augmenting Bayes Nets with Decision Trees

We first discuss decision trees for representing conditional probabilities in Parametrized Bayes nets, then how to convert the decision tree branches to Markov Logic Network clauses.

4.1 Decision Trees for Conditional Probabilities.

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. Suppose that a node X has three binary parents U, V, W. It may be the case that $P(x|u, V, W)$ is equal to some constant p_1 regardless of the values taken by V and W. Then the Bayes net requires only 5 parameters rather than 8 as in a tabular representation. A **decision tree** can take advantage of local independencies to compactly represent conditional probabilities [10]. The nodes in a decision tree for a Parametrized random variable *class* are parametrized random variables. An edge that originates in a PRV $f(t_1, \ldots, t_k)$ is labelled with one of the possible values in the range of f. The leaves are labelled with probabilities for the different possible values of the *class* variable. Figure 5 shows a tree with $class = ranking(S)$. In decision tree research, it is common to term such trees *probability estimation trees* to distinguish them from classification trees that select a definite class label at the leaves. In this paper, we follow the usage in Bayes net research and statistical-relational learning and refer to probability estimation trees simply as decision trees.

4.2 Context-sensitive Moralization: Converting Decision Trees to MLN Clauses

A Parametrized Bayes net structure with decision trees can be converted to an MLN by adding a clause for each complete branch in the tree that is the conjunction of the literals along the branch. Figure 5 illustrates a decision tree representation of the conditional probabilities for the child node $ranking(S)$ and the

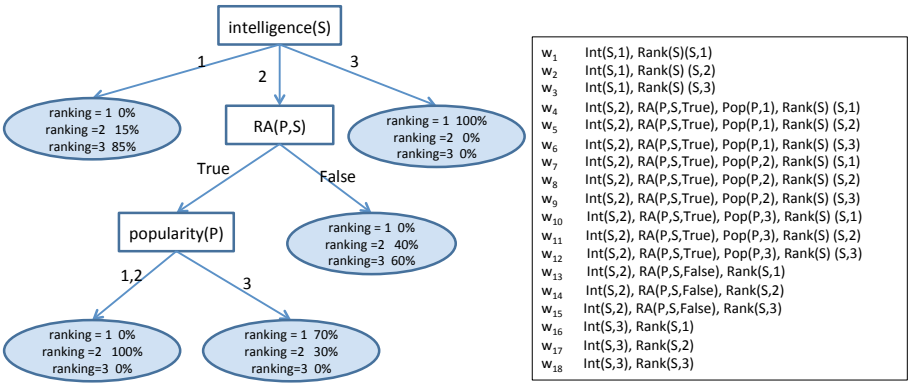

Fig. 5 A decision tree that specifies conditional probabilities for the *ranking*(S) node in Figure 3 and the corresponding MLN clauses generated from the decision tree . The number of clauses has been reduced from 54 to 18

clauses corresponding to the complete branches. Comparing Figure 5 with Figure 4 illustrates how the decision tree representation of the conditional probability parameters produces fewer clauses than standard moralization shown. In terms of MLN clauses, pruning decision tree nodes corresponds to merging clauses. A decision tree model may have branches of different sizes, so the clauses that are extracted for one child node may vary in the number of predicates.

5 Learning Decision Trees for a Bayes Net Structure

We discuss how the decision tree representation can be combined with a directed model relational learning method. First we briefly review the learn-and-join algorithm from previous work [9]. The learn-and-join algorithm is the state-of-the-art structure learning algorithm for Parametrized Bayes nets. We use it to find an initial Bayes net structure from a relational database. The Bayes net structure is then augmented with decision trees. The approach of this paper works with any structure learning algorithm for Parametrized Bayes nets, not just the learn-and-join algorithm.

5.1 Bayes Net Structure Learning Review: The Learn-and-Join Algorithm

Khosravi *et al.* present the learn-and-join structure learning algorithm. The algorithm upgrades a single-table Bayes net learner for relational learning. It learns dependencies among descriptive attributes conditional on the existence of a relationship, or a chain of relationships, between them. For details and pseudocode please see [9]. The key idea of the algorithm can be explained in terms of the *lattice of relationship table joins*. The learn-and-join algorithm builds a Parametrized Bayes net for the entire database $\mathcal{D}$ by level-wise search through the relationship join lattice. The user chooses a single-table Bayes net learner. The learner is applied

to entity tables that involve 0 relationships. Then the learner is applied to relationship joins that involve $s = 1, 2, \ldots$ relationship tables linked by foreign key pointers (matching types). The learned edges from smaller join tables are propagated to larger join tables. In the example database of Figure 2, the Bayes net learner is applied to the entity tables *Student*, *Professor*, *Course*. Then the Bayes net learner is applied to the *Registration* table (joined with attribute information from the *Student* and *Course* tables), to the *RA* table (joined with attribute information from the *Student* and *Professor* tables), and finally to the join of *Registration* and *RA* (joined with attribute information from the *Student*, *Course*, *Professor* tables). Schulte [24] provides a theoretical foundation: the learn-and-join algorithm optimizes a pseudo-likelihood function that measures the fit of a Parametrized Bayes Nets to a given input database. The measure is the expected log-likelihood of a random instantiation of the first-order variables in the Parameterized Bayes Net.

5.2 Learning Decision Trees for Conditional Probabilities.

For a fixed Parametrized Bayes net structure, we learn a set of decision trees that represent the conditional probabilities of each node given an assignment of values to its parents. Our system design is modular and can use any propositional decision tree learner that estimates class probabilities at the leaves. As the learn-and-join algorithm applies a Bayes net learner to join tables, we apply the decision tree learner to the same type of join tables as follows, for each node v in the Bayes net.

1. Form a **family join table** that combines all functor nodes in its family. This table is constructed from the set of all groundings that satisfy all literals that can be formed from the functor nodes in the family, that is, all possible assignments of values to nodes in the family.
2. Omit the ids of entities from the family join table, and apply the decision tree learner to the remaining columns. The result is a tree that represents, for each possible assignment of values to the parents of node v, a corresponding conditional probability in its leaf.

In a generic data table T, the conditional probability $P_T(\mathit{child} = \mathit{value} | \mathit{parents} = \mathbf{pa})$ is the number of rows with $\mathit{child} = \mathit{value}$ and $\mathit{parents} = \mathbf{pa}$, divided by the number of rows with $\mathit{parents} = \mathbf{pa}$. The family join table is constructed such that the conditional probability in the table is the number of groundings in the database that satisfy $\mathit{child} = \mathit{value}$ and $\mathit{parents} = \mathbf{pa}$, divided by the number of groundings that satisfy $\mathit{parents} = \mathbf{pa}$. Therefore a decision tree learner applied to the family data table learns a model of the conditional probabilities $P_{\mathcal{D}}(\mathit{child} = \mathit{value} | \mathit{parents} = \mathbf{pa})$ defined by the database distribution $P_{\mathcal{D}}$.

After augmenting the Bayes net structure with decision trees, we convert the decision tree branches to Markov Logic Network clauses to obtain an MLN structure. Algorithm 1 summarizes the MLN structure learning algorithm in pseudo code.

Example. The family join table for the node $\mathit{ranking}(S)$ is the join of the tables *RA*, *Student*, *Professor*, followed by projecting (selecting) the attributes *ranking*,*intelligence*, and *popularity*. Figure 6 illustrates the join data table for the example database

Algorithm 1: Pseudocode for compact MLN structure learning using the learn-and-join structure learning algorithm with decision trees.

Input: Database instance $\mathcal{D}$
Output: MLN for $\mathcal{D}$
Calls: $LearnAndJoin(\mathcal{D})$: Outputs a DAG G for an input database $\mathcal{D}$.
Calls: $Join(V)$: Takes in a set of nodes from $\mathcal{D}$ and outputs the data join table for the nodes in V.
Calls: $DecisionTree(T, child)$: A Decision Tree learner that outputs conditional class probabilities for *child* given data table T.

```
G = LearnAndJoin(D)
for all nodes v in G do
    v_family = v + Parents(v)
    T = Join(v_family)
    Tree_v = DecisionTree(T, v)
    for all leaf node entries of Tree_v do
        Add to MLN M the conjunction that corresponds to the decision tree branch of the leaf node.
    end for
end for
Return MLN M
```

of Figure 2. This data table provides the satisfying groundings for all literals that involve the four functor nodes in the family of $ranking(S)$, conditional on the existence of an RA relationship, that is conditional on $RA(S, P) = T$.

S.name(S)	P.name(P)	ranking(S)	intelligence(S)	popularity(P)
Jack	Oliver	3	1	1
Kim	Oliver	2	1	1
Kim	Jim	2	1	2

Fig. 6 The join data table for learning a decision tree that represents the conditional probabilities of $ranking(S)$ given its parents $intelligence(S), popularity(P), RA(S, P)$, where $RA(S, P) = T$. The tuples shown are the ones from the RA table, joined with the applicable information from the *Professor* and *Student* tables. The last three columns are given as input data to a propositional decision tree learner, with $ranking(S)$ designated as the class label.

A decision tree learner applied to such a data table might produce the decision tree shown in Figure 7(left). Since the popularity of a random professor is independent of that of a random student, the association between the functor nodes $popularity(P)$ and $ranking(S)$ depends on the existence of an RA link between them (i.e., the popularity of a professor predicts the ranking of a student only if the student is an RA for the professor). To indicate this dependence, we add the functor node $RA(S, P)$ as a parent of $popularity(P)$ in the decision tree for $ranking(S)$. Like most statistical-relational systems [1,33], and like the learn-and-join algorithm, we consider associations between attributes of two entities only conditional of the existence of a link between the entities. Therefore there is no

branch corresponding to $RA(S, P) = F$ in the decision tree of Figure 7(right). We leave as a project for future work learning associations conditional on the *absence* of a link between two entities (e.g., an association between the ranking of a student and the popularity of a professor given that the student is *not* an RA for the professor).

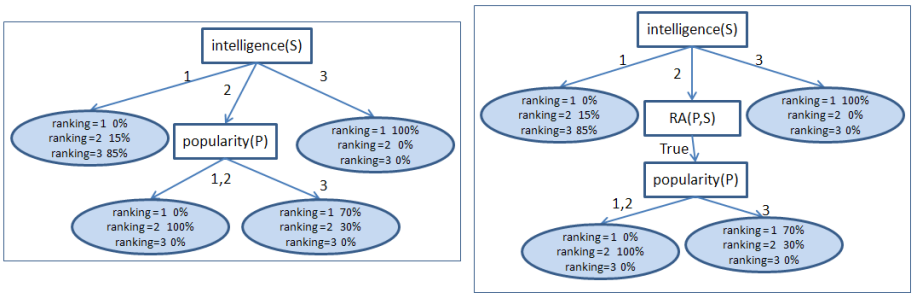

Fig. 7 Left: A decision tree learner applied to an input table like that shown in Figure 6 may produce this decision tree. Right: The split on the node *popularity*(P) depends implicitly on the existence of an RA link between a professor and a student. The output of the decision tree learner is augmented with a functor node $RA(S, P)$ to indicate this dependence.

5.3 Discussion.

Two bodies of related work are relevant: how to learn probability estimation trees for a single table, and how to upgrade a propositional decision tree learner for relational data. Most work on upgrading decision tree learning for relational data has been on learning classifiers rather than probability estimation trees.

Learning Probability Estimation Trees. In a seminal paper, Provost and Domingos observed that algorithms that build decision tree classifiers may not lead to good class probability estimates, mainly because trees for classification may be too small [13]. A number of improvements for probability estimation have been suggested, including the use of local probability models at the leaves of a tree [13–16]. Our focus in this paper is on whether the decision tree *representation* is sufficient in principle to produce more compact Markov Logic Networks; we leave exploring different tree learners for future work.

Upgrading Propositional Tree Learners. The tree learning approach in this paper can be viewed as a form of *lifted learning*, in analogy to lifted probabilistic inference [20]. Lifted inference uses as much as possible frequency information defined at the class level in terms of first-order variables, rather than facts about specific individuals. Likewise, our approach uses frequency information defined in terms of first-order variables, namely the number of satisfying groundings of a first-order

formula, which is provided by the family join table. Applying a propositional learner to the family join table can be justified theoretically using the random instantiation pseudo-likelihood measure [24]. Some ILP systems for discriminative learning, such as FOIL and Linus [34], are also based on the number of groundings of various clauses, which is similar to the join tables constructed by our algorithm.

Propositionalization approaches use aggregate functions to "flatten" relational data into a single table. Inductive Logic Programming (ILP) systems learn clauses that classify an example as positive by logical entailment [34, 21]. Typically this involves the use of existential quantification as an aggregation mechanism. Relational probability trees employ a range of aggregate functions as features for predicting class probabilities [23]. While Markov Logic networks can be extended with aggregate functions, the basic log-linear prediction model of MLNs is different from approaches that use aggregate features for classification.

Gradient Boosting for Relational Regression Trees. A recent method for learning Markov Logic Networks, developed independently of our work on decision trees, is functional gradient boosting of regression trees [26]. The basic similarity is that paths in the regression tree are converted to MLN formulas in a similar manner to our moralization method. The main differences to our Bayes net approach are as follows.

Model Class. (i) The leaves of a relational regression tree contain general weights for the MLN clauses. Hence the tree cannot be interpreted as specifying a conditional probability. (ii) Boosting produces an ensemble of trees for each node (target predicate), rather than a single tree as in our system.

Learning. (i) Two-class boosting is used, so the learning method can be applied to Boolean predicates (e.g., relationship literals), but not immediately to multi-valued attributes, which are the focus of our evaluation.[2] (ii) Gradient boosting performs simultaneously MLN structure and parameter learning, because the regression trees specify both the Markov blanket of a node and the weights of clauses. In contrast, we apply decision tree learning only to obtain a more compact structure, and we apply it only to the parents of a target node, not to its entire Markov blanket (which in a Bayes net includes children and co-parents). This means that decision trees for different nodes are learned independently of each other. By contrast, in the gradient boosting approach, trees learned for one node v produce clauses that are applied to learning trees for other nodes (namely, those in the Markov blanket of v).

Since functional gradient boosting is a powerful and very general framework for regression, a promising topic for future work is to apply gradient boosting for augmenting a Bayes net with decision trees, by learning a set of decision trees for a target node v conditional on its parents. While the learned set of decision trees may not be as easy to interpret as a single one [25, Sec.3.5], the Markov Logic Network that results from converting the trees is a set of clauses in either case.

[2] It is possible to change the data representation so that all attributes are binary, see [35, 9, 25].

6 Experimental Design

We first discuss the datasets used, then the systems compared, finally the comparison metrics.

Datasets. We used 4 benchmark real-world databases. For more details please see the references in [9] and on-line sources such as [36].

MovieLens Database. This is a standard dataset from the UC Irvine machine learning repository.

Mutagenesis Database. This dataset is widely used in ILP research. It contains information on Atoms, Molecules, and Bonds between them. We use the discretization of [9].

Hepatitis Database. This data is a modified version of the PKDD02 Discovery Challenge database. 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 followed the modification of [37], and used a subset of the tables and features for fast inference.

Table 3 lists the resulting full database sizes in terms of total number of tuples and number of ground atoms, which is the input format for Alchemy. We also show the average values for the functor nodes in the database.

Dataset	**#tuples**	**#Ground atoms**	**#Values (Average)**
Movielens	82623	170143	2.7
Mutagenesis	15218	35973	3.9
Hepatitis	12447	71597	4.4
Mondial	814	3366	4.3

Table 3 Size of full 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. The average number of values of the descriptive attributes in each database is shown. Binary descriptive attributes (e.g., gender) are rare.

Comparison Systems. All simulations were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. Our code and datasets are available on the world-wide web [36]. We made use of the following existing implementations.

Single Table Bayes Net Search GES search [38] with the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; [39]).

Single Table Decision Tree Learning The J48 program of the Weka package [40], which implements the C4.5 decision tree algorithm. We used the probability estimation setting, which turns off pruning and applies the Laplace correction, as recommended by Provost and Domingos [13].

MLN Parameter Learning The default weight training procedure [41] of the Alchemy package [6], Version 30.

MLN Inference The MC-SAT inference algorithm [42] to compute a probability estimate for each possible value of a descriptive attribute for a given object or tuple of objects.

We use the Alchemy parameter learning and the state-of-the art MC-SAT inference method for compatibility with previous studies [43,18,17]. We also carried out simulations with an exact evaluation of the MLN classification formula given by Domingos and Richardson [4]. The relative improvement of context-sensitive vs. standard moralization is similar, but the absolute accuracies are lower, which is consistent with the findings of previous studies of MLN inference.

Algorithms. We compared four MLN structure learning algorithms.

MBN The structure is learned using the learn-and-join algorithm (Section 5.1). The weights of clauses are learned using Alchemy. This method is called MBN for "Moralized Bayes Net" by Khosravi *et al.* [9]. The learn-and-join algorithm produces clauses with positive relationship literals only.

MBN + DT The structure is first learned using the learn-and-join algorithm and then augmented with decision trees using Algorithm 1. As illustrated in Figure 5, this algorithm produces clauses with positive relationship literals only. The weights of clauses are learned using Alchemy.

LHL Lifted Hypergraph Learning [18] uses relational path finding to induce a more compact representation of data, in the form of a hypergraph over clusters of constants. Clauses represent associations among the clusters.

LSM Learning Structural Motifs [17] uses random walks to identify densely connected objects in data, and groups them and their associated relations into a motif.

The first two methods compare variants of the moralization method, whereas the last two are reference methods. We chose LSM and LHL because they are the most recent MLN structure learning methods that are based on the Alchemy system.

Performance Metrics. We use 4 performance metrics: Number of Clauses or Parameters, learning time, Accuracy (ACC), and Conditional log likelihood (CLL). ACC and CLL have been used in previous studies of MLN learning [18]. 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 that represent descriptive attributes. We do not use Area under Curve (AUC), as it mainly applies to binary values, and most of the attributes in our dataset are nonbinary. We evaluate the learning methods using two different schemes.

5-fold cross-validation. We formed 5 subdatabases for each using standard subgraph subsampling [44,9], which selects entities from each entity table uniformly at random and restricts the relationship tuples in each subdatabase to those that involve only the selected entities. The models were trained on 4 of the 5 subdatabases, then tested on the remaining fold. We report the average over the 5 runs, one for each fold.

	MBN + DT	MBN	LSM	LHL
MovieLens	39	327	10	NT
Mondial	102	2470	20	25
Mutagen	50	880	13	NT
Hepatitis	120	793	23	27

Table 4 5-fold cross-validation estimate of the number of parameters in learned model.

Learning Curve. To study the learning behavior at different sample sizes, we performed a set of experiments that train the model on N% of the data, where N ranges from 10 to 100 in step sizes of 10. Results for each sample size are averages over 10 runs.

7 Evaluation Results

As our aim is to learn more compact structures, we first examine the number of parameters or clauses learned. Our results indicate that the decision tree representation leads to substantially more compact models. Learning time measurements show a significant speed-up in weight learning with the smaller models. Predictive performance is competitive with standard moralization based on conditional probability tables, in many cases even superior.

7.1 Number of Parameters/Clauses

Table 4 shows the average number of clauses produced by the MLN models using 5-fold cross-validation (so the average is over 5 different measurements). *Adding decision trees to the learn-and-join algorithm leads to much more compact models*, with improvement ratios in the range of 5-25. The LSM algorithm and the LHL algorithm learn a very small number of clauses most of which are unit or short clauses. Figure 8 shows the number of parameters learned respectively by LSM, LHL, and MBN + DT using a Learning Curve scheme. MBN + DT exploits increasing data to learn a more complex model. LSM and LHL learn almost the same very small number of clauses independent of the data size. Inspection of the learned clauses by LSM and LHL shows that the rules are mostly just the unit clauses that model marginal probabilities (e.g., $\mathit{intelligence}(S, I)$) [4]. This indicates underfitting the data, as our measurements of ACC and CLL confirm (Tables 6, 7 below).

7.2 Learning times

Table 5 shows average times for learning using 5-fold cross-validation. LHL fails to terminate on two of the datasets and is very slow on the other two datasets. The learn-and-join structure learning method scales well, even with decision tree learning added. The computational bottleneck for the two moralization methods is the weight optimization that uses the relatively slow Alchemy routines. *Since decision*

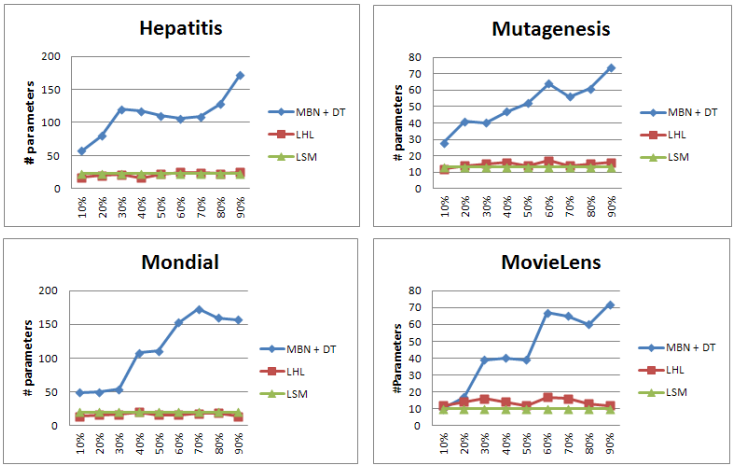

Fig. 8 MBN + DT learns more clauses when more data is available. LSM and LHL produce a very small number of clauses that is essentially independent of the data size. The experiments train the model on N% of the data, where N ranges from 10 to 100 in step sizes of 10.

	MBN + DT	MBN	LSM	LHL
MovieLens	22 + 345	15 + 3401	34.03	NT
Mondial	9 + 18	4+ 1168	29.0	11524
Mutagen	18 + 274	12 + 4425	26.47	NT
Hepatitis	21 + 813	15 + 6219	10.94	72452

Table 5 5-fold cross-validation estimate for Average learning times in seconds. Learning times for the moralization methods are given as (structure learning time + weight learning time).

trees reduce the number of model parameters, they speed up weight optimization by a factor of about 10. So a small increase in the complexity of structure learning achieves a very significant decrease in the complexity of parameter learning.

To further examine the scalability of the moralization algorithms, we use a Learning Curve design. Figure 9 indicates that the weight learning time for MBN increases exponentially with the size of the dataset, but adding decision trees leads to much better scaling. The LSM method (not shown) is very fast for all dataset sizes, because it produces a small set of short rules that is essentially independent of the dataset size. This is mostly due to under-fitting as indicated by the number of parameters in LSM models, and by the predictive accuracy results, which we report next.

7.3 Predictive Performance

We first discuss accuracy, then conditional log-likelihood.

Accuracy. The average accuracy of the two learn-and-join methods is quite similar, and *about 10-15% higher than that of LSM and LHL.* The accuracy numbers are

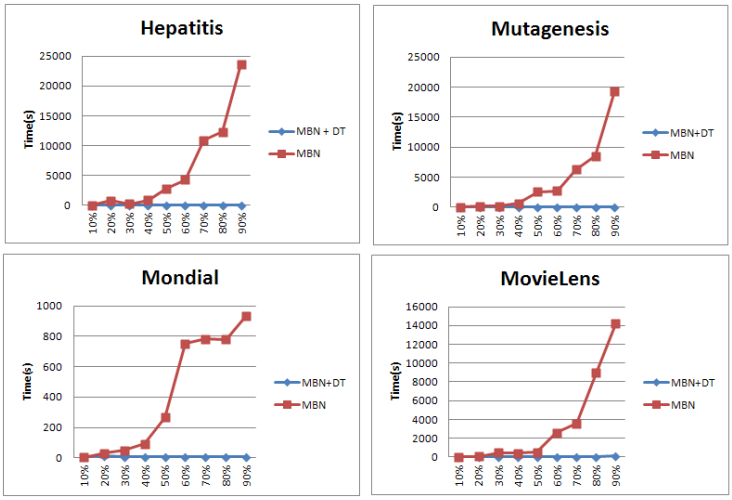

Fig. 9 While structure learning is fast using the learn-and-join algorithm, the weight learning time for standard moralization method MBN increases exponentially with the size of the database. Context-sensitive moralization with decision trees scales much better (MBN+DT). The experiments train the model on N% of the data, where N ranges from 10 to 100 in step sizes of 10.

	MBN + DT	MBN	LSM	LHL
MovieLens	0.55 ± 0.04	**0.56 ± 0.04**	0.39 ± 0.04	NT
Mondial	**0.41 ± 0.054**	**0.41 ± 0.055**	0.26 ± 0.018	0.25± 0.03
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 6 5-fold cross-validation estimate for the accuracy of predicting the true values of descriptive attributes, averaged over all descriptive attribute instances. Observed standard deviations are shown.

fairly low overall because many of the descriptive attributes have many possible values (e.g. 9 for Lumo in Mutagenesis); see Table 3. The LSM accuracy variance is low, which together with poor average accuracy is consistent with the hypothesis that LSM underfits the data.

The moralization method performs generative learning over all attributes, a significantly more difficult task than discriminative learning. While it is usual in Markov Logic Network evaluation to report an average over all predicates in a database, we observed that there is considerable variance among the predictive accuracies for different predicates. For example in the Mutagenesis dataset, the accuracy of the learned MBN model for predicting positive mutagenicity is 87% on 10-fold cross-validation, which is in the $86\% - 88\%$ range of accuracies reported for discriminative methods [45–47].

Conditional Log-likelihood. This measure is especially sensitive to the quality of the parameter estimates. Without decision trees, the MBN method performs clearly worse on 3 of the 4 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. Because the average accuracy is low, this means that when mistaken predictions are made, they are not made with great confidence.

	MBN + DT	MBN	LSM	LHL
MovieLens	-0.8 ± 0.25	-0.79 ± 0.12	**-0.65 ± 0.10**	NT
Mondial	**-1.36 ± 0.12**	−1.76 ± 0.37	-1.43 ± 0.027	-1.98 ± 0.035
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 7 5-fold cross-validation estimate for the conditional log-likelihood assigned to the true values of descriptive attributes, averaged over all descriptive attribute instances. Observed standard deviations are shown.

8 Conclusion and Future Work

Augmenting Bayes net learning with decision tree learning leads to a compact set of Horn clauses that represent generative statistical patterns in a relational database. In our simulations on four benchmark relational databases, decision trees significantly reduced the number of Bayes net parameters, by factors ranging from 5-25. The pattern of average predictive performance and its variance is consistent with the hypothesis that the decision tree method strikes an attractive balance: It avoids the overfitting tendencies of the basic Bayes net moralization algorithm, and it avoids the underfitting tendencies of the Markov Logic learners (LSM and LHL). After converting the Bayes net Horn clauses to Markov Logic Networks, MLN inference can be used to evaluate the predictive accuracy of the resulting models. In our empirical evaluation, the predictive performance of the pruned models is competitive with or superior to the unpruned models.

A limitation of our system is that we used generic Markov Logic Network algorithms for parameter estimation implemented in the Alchemy package. While the parameter estimation routines of Alchemy run much faster than the structure learning routines, on some datasets we found that parameter estimation can still take a long time. As Markov Logic Networks obtained by moralization have a special structure, it may be possible to design fast parameter estimation routines for them. Domingos and Richardson suggest using the log-conditional probability of a parent-child configuration as the weight for the corresponding MLN clause [4, 12.5.3]; this method is also recommended by the Alchemy Group [48]. In future work, we plan to investigate log-conditional probabilities and other weight learning techniques for context-sensitive moralization. Another important project is an empirical comparison of Markov network learning via the moralization approach with functional gradient boosting [26].

Acknowledgments

The anonymous referees for the ILP conference and the Machine Learning Journal provided helpful comments and pointers to related literature, especially on learning

probability estimation trees. This research was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Getoor, L., Friedman, N., Koller, D., Pfeffer, A., Taskar, B.: Probabilistic relational models. [49] chapter 5 129–173
2. Kersting, K., de Raedt, L.: Bayesian logic programming: Theory and tool. [49] chapter 10 291–318
3. Fierens, D., Blockeel, H., Bruynooghe, M., Ramon, J.: Logical bayesian networks and their relation to other probabilistic logical models. In: ILP. (2005) 121–135
4. Domingos, P., Richardson, M.: Markov logic: A unifying framework for statistical relational learning. [49]
5. Ngo, L., Haddawy, P.: Answering queries from context-sensitive probabilistic knowledge bases. Theor. Comput. Sci. **171**(1-2) (1997) 147–177
6. 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. (2009) Version 30.
7. Neville, J., Jensen, D.: Relational dependency networks. [49] chapter 8
8. Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: UAI. (2002) 485–492
9. Khosravi, H., Schulte, O., Man, T., Xu, X., Bina, B.: Structure learning for Markov logic networks with many descriptive attributes. In: AAAI. (2010) 487–493
10. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in bayesian networks. In: UAI. (1996) 115–123
11. Friedman, N., Goldszmidt, M.: Learning Bayesian networks with local structure. In: NATO ASI on Learning in graphical models. (1998) 421–459
12. Getoor, L., Taskar, B., Koller, D.: Selectivity estimation using probabilistic models. ACM SIGMOD Record **30**(2) (2001) 461–472
13. Provost, F.J., Domingos, P.: Tree induction for probability-based ranking. Machine Learning **52**(3) (2003) 199–215
14. Fierens, D., Ramon, J., Blockeel, H., Bruynooghe, M.: A comparison of pruning criteria for probability trees. Machine Learning **78**(1-2) (2010) 251–285
15. Zhang, H., Su, J.: Conditional independence trees. In: ECML, Springer (2004) 513–524
16. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: KDD. (1996) 202–207
17. Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: ICML. (2010) 551–558
18. Kok, S., Domingos, P.: Learning markov logic network structure via hypergraph lifting. In: ICML. (2009) 64–71
19. Quinlan, J.R., Rivest, R.L.: Inferring decision trees using the minimum description length principle. Inf. Comput. **80**(3) (1989) 227–248
20. Poole, D.: First-order probabilistic inference. In: IJCAI. (2003) 985–991
21. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence **101**(1-2) (1998) 285–297
22. Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., Kadie, C., Kaelbling, P.: Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research **1** (2000) 49–75
23. Neville, J., Jensen, D.: Relational dependency networks. Journal of Machine Learning Research **8** (2007) 653–692
24. Schulte, O.: A tractable pseudo-likelihood function for bayes nets applied to relational data. In: SIAM SDM. (2011) 462–473
25. Natarajan, S., Khot, T., Kersting, K., Gutmann, B., Shavlik, J.W.: Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning **86**(1) (2012) 25–56
26. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.W.: Learning markov logic networks via functional gradient boosting. In: ICDM. (2011) 320–329
27. Bratko, I.: Prolog (3rd ed.): programming for artificial intelligence. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)

28. Chiang, M., Poole, D.: Reference classes and relational learning. Int. J. Approx. Reasoning **53**(3) (2012) 326–346
29. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence **46**(3) (1990) 311–350
30. Getoor, L., Grant, J.: Prl: A probabilistic relational language. Machine Learning **62**(1-2) (2006) 7–31
31. Ullman, J.D.: Principles of database systems. 2. Computer Science Press (1982)
32. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann (1988)
33. Chen, H., Liu, H., Han, J., Yin, X.: Exploring optimization of semantic relationship graph for multi-relational Bayesian classification. Decision Support Systems **48:1** (July 2009) 112–121
34. Dzeroski, S.: Inductive logic programming in a nutshell. [49]
35. Kok, S., Domingos, P.: Statistical predicate invention. In: ICML, ACM (2007) 433–440
36. Khosravi, H., Man, T., Hu, J., Gao, E., Schulte, O.: Learn and join algorithm code. URL = `http://www.cs.sfu.ca/~oschulte/jbn/`.
37. She, R., Wang, K., Xu, Y.: Pushing feature selection ahead of join. In: SIAM SDM. (2005)
38. Chickering, D.: Optimal structure identification with greedy search. Journal of Machine Learning Research **3** (2003) 507–554
39. The Tetrad Group: The Tetrad project (2008) http://www.phil.cmu.edu/projects/tetrad/.
40. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explorations **11**(1) (2009) 10–18
41. Lowd, D., Domingos, P.: Efficient weight learning for Markov logic networks. In: PKDD. (2007) 200–211
42. Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies. In: AAAI. (2006)
43. Mihalkova, L., Mooney, R.J.: Bottom-up learning of Markov logic network structure. In: ICML, ACM (2007) 625–632
44. Frank, O.: Estimation of graph totals. Scandinavian Journal of Statistics **4:2** (1977) 81–89
45. Srinivasan, A., Muggleton, S., Sternberg, M., King, R.: Theories for mutagenicity: A study in first-order and feature-based induction. Artificial Intelligence **85**(1-2) (1996) 277–299
46. Quinlan, J.: Boosting first-order learning. In: Algorithmic Learning Theory, Springer (1996) 143–155
47. Sebag, M., Rouveirol, C.: Tractable induction and classification in first order logic via stochastic matching. In: IJCAI. (1997) 888–893
48. Alchemy Group: Frequently asked questions URL = `http://alchemy.cs.washington.edu/`.
49. Getoor, L., Tasker, B.: Introduction to statistical relational learning. MIT Press (2007)