Fast Learning of Relational Dependency Networks

Oliver Schulte, Zhensong Qian, Arthur E. Kirkpatrick
Xiaoqian Yin, Yan Sun

School of Computing Science, Simon Fraser University, Canada
{oschulte,zqian,ted,xiaoqian_yin,sunyans}@sfu.ca

Abstract. A Relational Dependency Network (RDN) is a directed graphical model widely used for multi-relational data. These networks allow cyclic dependencies, necessary to represent relational autocorrelations. We describe an approach for learning both the RDN's structure and its parameters, given an input relational database: First learn a Bayesian network (BN), then transform the Bayesian network to an RDN. Thus fast Bayesian network learning translates into fast RDN learning. The BN-to-RDN transform comprises a simple, local adjustment of the Bayesian network structure and a closed-form transform of the Bayesian network parameters. This method can learn an RDN for a dataset with a million tuples in minutes. We empirically compare our approach to state-of-the-art RDN learning methods that use functional gradient boosting, on six benchmark datasets. Learning RDNs via BNs scales much better to large datasets than learning RDNs with boosting, and provides competitive accuracy in predictions.

1 Introduction

Learning graphical models is one of the main approaches to extending machine learning for relational data. Two of the major classes of graphical models are dependency networks (DNs) [9] and Bayesian networks (BNs) [18]. We describe a new approach to learning dependency networks: first learn a Bayesian network, then convert that network to a dependency network. This hybrid approach combines the advantages of learning Bayesian networks with the advantages of inference from dependency networks. Our experiments show that the hybrid learning algorithm can produce dependency networks for large and complex databases, up to one million records and 19 predicates. The predictive accuracy of the resulting networks is competitive with those from state-of-the-art function gradient boosting methods but scales substantially better than the boosting methods. We make three contributions:

1. A faster approach for learning relational dependency networks: first learn a Bayesian network, then convert it to a dependency network.
2. A closed-form log-linear discriminative model for computing the relational dependency network parameters from Bayesian network structure and parameters.

3. Necessary and sufficient conditions for the resulting network to be **consistent**, defined as the existence of a single joint distribution that induces all the conditional distributions defined by the dependency network [9].

2 Relational Dependency Networks and Bayesian Networks

We review dependency networks and their advantages for modelling relational data. We assume familiarity with the basic concepts of Bayesian networks [18].

2.1 Dependency networks and Bayesian networks

The structures of Bayesian networks and dependency networks are defined by a directed graph whose nodes are random variables. Unlike Bayesian networks, a dependency network graph may contain cycles, including the special case of bi-directed edges. As with Bayesian networks, the parameters of dependency networks are conditional distributions over the value of a child node given its parents. However, the independence property of dependency networks is simpler: each node is independent of all other nodes given an assignment of values to only its parents. By contrast a node in a Bayesian network is only independent given an assignment of values to its parents, its children, and the co-parents of its children. In graphical model terms, the **Markov blanket** of a node in a dependency network, the minimal set of nodes such that assigning them values will make this node independent of the rest of the network, is its parents.

Consequently, a parameter in a dependency network effectively specifies the probability of a node value given an assignment of values to *all* other nodes. Because Gibbs sampling can derive a joint distribution from these parameters [9, 17], we refer to them as **Gibbs conditional probabilities**, or simply **Gibbs probabilities**.[1] This is the counterpart to the formula deriving a joint distribution from the product of a Bayesian network's conditional probabilities.

2.2 Relational Dependency Networks

There are various notations for defining random variables in relational structures, of equivalent expressive power. We adopt a functor-based notation from a logic for graphical-relational models [21, 20]. A functor is a symbol denoting a function or predicate. Each functor has a set of values (constants) called the **domain** of the functor. We consider only functors with finite domains. An expression $f(\tau_1, \ldots, \tau_k)$, where f is a functor and each τ_i is a first-order variable or a constant, is a **Parametrized Random Variable** (PRV). A directed acyclic graph whose nodes are PRVs is a **parametrized Bayesian network structure**, while a general directed graph whose nodes are PRVs is a **relational dependency network structure** (RDN). RDNs extend dependency networks

[1] The terminology of DNs [9] calls these "local probability distributions".

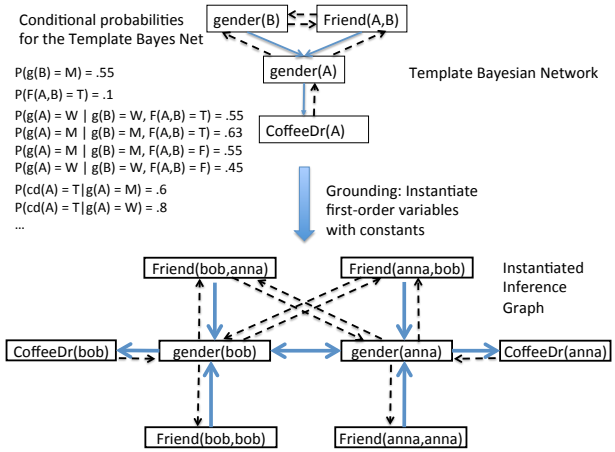

Fig. 1. A Bayesian/dependency template network (top) and the instantiated inference graphs (bottom). By convention, predicates (Boolean functors) are capitalized. Edges from the BN template are solid blue, while edges added by the BN-to-DN transformation are dashed black. The edge set in the DN comprises both solid and dashed arrows. Note that although the template BN is acyclic, its instantiation features a bi-directed edge between $gender(bob)$ and $gender(anna)$.

for relational data via knowledge-based model construction [17]: The first-order variables in a template RDN graph are instantiated for a specific domain of individuals to produce an *instantiated* or *ground* propositional DN graph, the **inference graph**. Figure 1 gives a dependency network template and its inference graph. Given an edge in the template RDN, instantiating both the parent and the child of the edge with the same grounding produces an edge in the inference graph. An example Gibbs probability for the graph in Figure 1 (abbreviating functors) is

$$P(g(anna)|g(bob), CD(anna), F(anna, bob), F(bob, anna), F(anna, anna)).$$

Both the structure and the parameter space of RDN models offer special advantages for relational data [17, 15]:

1. Dependency network structures are well-adapted for relational data because they allow cyclic dependencies, so grounding a dependency network template is guaranteed to produce a valid dependency network (whereas grounding a Bayesian net may introduce cycles, making the instantiation non-Bayesian).
2. Relational prediction requires aggregating information from different linked individuals [16]. In a dependency network parameter, the aggregation encompasses the entire Markov blanket of a target node, whereas for Bayesian network parameters, the aggregation encompasses only part of the blanket.

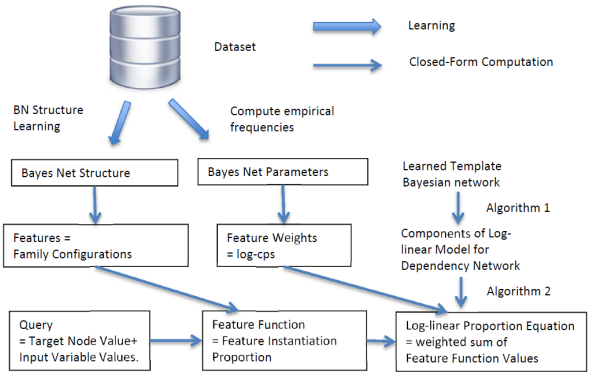

Fig. 2. The program flow for computing Gibbs probabilities from a template Bayesian network. Features and weights are computed from the Bayesian network. Feature function values are computed for each query.

3 Learning Relational Dependency Networks via Bayesian Networks

Our algorithm for rapidly learning relational dependency networks (Figure 2) begins with any relational learning algorithm for Bayesian networks. We then apply a simple, fast transformation to the resulting Bayesian network, obtaining a relational dependency template. Finally we apply a closed-form computation to derive the dependency network parameters from the Bayesian structure and parameters.

BN-to-DN structure conversion. Converting a Bayesian network structure to a dependency network structure is simple: for each node, add an edge pointing to the node from each member of its BN Markov blanket [9]. The result contains bidirectional links between each node, its children, and its co-parents (nodes that share a child with this one). This is equivalent to the standard moralization method for converting a BN to an undirected model [3], except that the dependency network contains bi-directed edges instead of undirected edges. Bidirected edges have the advantage that they permit assignment of different parameters to each direction, whereas undirected edges have only one parameter.

BN-to-DN parameter conversion. Converting Bayesian network parameters to dependency network parameters is simple for propositional i.i.d. data: solve for the Gibbs conditional probabilities given Bayesian network parameters. The propositional result is as follows. A **family** comprises a node and its parents. A **family configuration** specifies a value for a child node and each of its parents.

For example in the Bayesian network of Figure 1, a family configuration is

$$gender(\mathbb{A}) = \mathrm{M}, Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}, gender(\mathbb{B}) = \mathrm{M}.$$

For propositional data, an assignment of values to the Markov blanket of a target node assigns a unique configuration for each family whose child is the target node or one of its children. Hence the Markov blanket induces a *unique* log-conditional probability for each such family configuration. The probability of a target node value given an assignment of values to the Markov blanket is then proportional to *the exponentiated sum of these log-conditional probabilites* [21, Ch.14.5.2].

With relational data, different family configurations such as the one above can be simultaneously instantiated, *multiple times.* We adapt the propositional log-linear equation for relational data by replacing the unique log-conditional probability with the *expected* log-conditional probability that results from selecting an instantiation of the family configuration uniformly at random. The probability of a target node value given an assignment of values to the Markov blanket is then proportional to the exponentiated sum of the expected log-conditional probabilites. We describe the resulting closed-form equation in the next section.

Algorithm 1: Computing Features and Weights for Template Dependency Network.

Input: Template Bayesian Network B (Structure and Parameters)
Output: A List of Relevant Features; a Weight for each Feature

for each target node T **do**
  initialize $Feature\_Weight\_List(T)$ as the empty list
  for each U in $\{T \cup \mathrm{Ch}(T)\}$ **do**
    for each value u of the child node U **do**
      for each vector of parent values $\boldsymbol{u}_{pa}$ **do**
        Feature $F := (U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$
        FeatureWeight $w := \ln \theta(U = u | \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$
        if the Feature F does not contain a false relationship other than T **then**
          add (F, w) to $Feature\_Weight\_List(T)$
        end if
      end for
    end for
  end for
end for
return $Feature\_Weight\_List(T)$

4 The Log-linear Proportion Equation

We propose a log-linear equation, the **log-linear proportion equation**, for computing a Gibbs conditional probability for a ground target node, T^*, given

Algorithm 2: Computing Gibbs conditional probabilities, the parameters of the Inference Dependency Network.

Input: Feature-Weight List of Dependency Network, Query $P(T^* = t|\Lambda^*) =?$. T is a template node, $T^* = T\gamma$ is the target grounding.
Output: Normalized log-linear score

```
initialize score(T* = t) := 0
for each Feature F = (U = u, Pa(U) = u_pa) in Feature_Weight_List(T) do
    Let w be the weight listed for feature F
    {Next compute feature function.}
    RelFamCnt(F) := n^r [γ; U = u, Pa(U) = u_pa; T* = t, Λ*]
    TotalRelFamCnt(U) := Σ_{u', u'_pa} n^r [γ; U = u', Pa(U) = u'_pa; T* = t, Λ*]
    FamilyProportion p^r(F) := RelFamCnt(F) / TotalRelFamCnt(U)
    score(T* = t) += p^r · w
end for
return Normalized scores for target node.
```

(i) a target value t for the target node, (ii) a complete set of values Λ^* for all ground terms other than the target node, and (iii) a template Bayesian network. The template structure is represented by functions that return the set of parent nodes of U, $\mathrm{Pa}(U)$, and the set of child nodes of U, $\mathrm{Ch}(U)$. The parameters of the template are represented by the conditional probabilities of a node U having a value u conditional on the values of its parents, $\theta(U = u|\mathrm{Pa}(U) = \boldsymbol{u}_{pa})$. A grounding γ substitutes a constant for each member of a list of first-order variables. A grounding is therefore equivalent to an equality constraint $\{\mathbb{A}_1 = a_1, \ldots, A_k = a_k\}$. Applying a grounding to a template node defines a fully ground target node. For instance, we may have $gender(\mathbb{A})\{\mathbb{A} = sam\} = gender(sam)$. These are combined in the following log-linear equation:

Definition 1 (The Log-Linear Proportion Equation).

$$P(T^* = t|\Lambda^*) \propto$$

$$\sum_{U} \sum_{u, \boldsymbol{u}_{pa}} [\ln \theta(U = u|\mathrm{Pa}(U) = \boldsymbol{u}_{pa})] \cdot \mathrm{p^r}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]$$

where

U varies over $\{T\} \cup \mathrm{Ch}(T)$,
the singleton value u varies over *the range of* U;
the vector of values $\boldsymbol{u}_{pa}$ varies over *the product of the ranges of* U*'s parents*;
$T^* = T\gamma$ is *is the target grounding of template node* T;
and $\mathrm{p^r}$ is *the proportion feature function.*

The feature function $\mathrm{p^r}$ specifies the proportion of instantiations that satisfy a given family configuration, relative to all family configurations with positive links only.

Example. Table 1 illustrates the computation of our log-linear model for predicting the gender of a new test instance (sam).

Table 1. Applying the log-linear proportion equation with the Bayesian network of Figure 1 to compute $P(gender(sam) = \mathrm{W}|\Lambda^*)$ and $P(gender(sam) = \mathrm{M}|\Lambda^*)$. Each row represents a feature/family configuration. For the sake of the example we suppose that the conjunction Λ^* specifies that Sam is a coffee drinker, has 60 male friends, and 40 female friends. CP refers to the conditional probability BN parameter of Figure 1. For the feature weights $w \equiv \ln(CP)$.

Child Value u	Parent State $\boldsymbol{u}_{pa}$	CP	w	p^{r}	$w \times \mathrm{p}^{\mathrm{r}}$
$g(sam) = \mathrm{W}$	$g(B) = \mathrm{W}$, $F(sam, B) = \mathrm{T}$	0.55	−0.60	0.4	−0.24
$g(sam) = \mathrm{W}$	$g(B) = \mathrm{M}$, $F(sam, B) = \mathrm{T}$	0.37	−0.99	0.6	−0.60
$CD(sam) = \mathrm{T}$	$g(sam) = \mathrm{W}$	0.80	−0.22	1.0	−0.22
$CD(sam) = \mathrm{F}$	$g(sam) = \mathrm{W}$	0.20	−1.61	0.0	0.00
Sum ($\exp(Sum) \propto P(gender(sam) = \mathrm{W}\|\Lambda^*)$)					−1.06
$g(sam) = \mathrm{M}$	$g(B) = \mathrm{W}$, $F(sam, B) = \mathrm{T}$	0.45	−0.80	0.4	−0.32
$g(sam) = \mathrm{M}$	$g(B) = \mathrm{M}$, $F(sam, B) = \mathrm{T}$	0.63	−0.46	0.6	−0.28
$CD(sam) = \mathrm{T}$	$g(sam) = \mathrm{M}$	0.60	−0.51	1.0	−0.51
$CD(sam) = \mathrm{F}$	$g(sam) = \mathrm{M}$	0.40	−0.92	0.0	0.00
Sum ($\exp(Sum) \propto P(gender(sam) = \mathrm{M}\|\Lambda^*)$)					−1.11

It is common in statistical-relational models to restrict predictors to existing relationships only [6, 21]. The inner sum of Formula 1 computes the expected log-conditional probability for a family with child node U, when we randomly select a relevant grounding of the first-order variables in the family.

Definition 1 has the form of a log-linear model [26]: The features of the model are the family configurations $(U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$ where the child node is either the target node or one of its chldren. The feature weights are the log-conditional BN probabilities defined for the family configuration. The input variables are the values of the ground nodes other than the target nodes, specified by the conjunction Λ^*. The family count specifies how many times the feature is instantiated in the input variables (plus the target node value). The family proportion is the feature function, which maps a feature to a real value given the input variables. Proportions have the desirable consequence that all feature functions are normalized to the [0,1] range. Feature instantiation proportions are computed as follows.

1. For a given family configuration $(U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$, let the **family count**
$$\mathrm{n}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]$$

be the number of instantiations that (a) satisfy the family configuration and the ground node values specified by $T^* = t, \Lambda^*$, and (b) are consistent with the equality constraint defined by γ. This notation is consistent with the parfactor notation of [20].
2. The **relevant family count** n^r is 0 if the family configuration contains a false relationship (other than the target node), else equals the feature count.
3. The **family proportion** is the relevant family count, divided by the total sum of all relevant family counts for the given family. In symbols:

$$\mathrm{p}^{\mathrm{r}}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right] = \frac{\mathrm{n}^{\mathrm{r}}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]}{\sum_{u', \boldsymbol{u}'_{pa}} \mathrm{n}^{\mathrm{r}}\left[\gamma; U = u', \mathrm{Pa}(U) = \boldsymbol{u}'_{pa}; T^* = t, \Lambda^*\right]}$$

Table 1 illustrates the computation of these quantities. Algorithm 1 shows pseudocode for the closed-form transformation of Bayesian network structure and parameters into features and weights for the dependency network. Algorithm 2 shows pseudocode for computing the scores defined by the log-linear Equation 1, given a list of weighted features and a target query.

5 Theoretical Analysis

We analyze the computational complexity of the BN-to-DN conversion and of computing a classification score using the log-linear equation 1. A key theoretical question for a DN is whether the network is *consistent*. In the theory of dependency networks, consistency refers to the existence of a single joint probability distribution that induces the various local conditional probability distributions for each node [9]. We prove that under mild assumptions, a relational dependency network constructed from a Bayesian network is consistent if and only if the relevant family counts are the same for each ground node.

5.1 Complexity Analysis

The loops of Algorithm 1 enumerate every family configuration in the template Bayesian network exactly once. Therefore *computing features and weights takes time linear in the number of parameters of the Bayesian network.*

Evaluating the log-linear equation as shown in Algorithm 2, requires finding the number of instantiations that satisfy a conjunctive family formula, given a grounding. This is an instance of the general problem of computing the number of instantiations of a formula in a relational structure. Computing this number is a well-studied problem with highly efficient solutions [27, 24]. The complexity of this problem is discussed in [24]. A key parameter is the number m of first-order variables that appear in the formula. A loose upper bound on the complexity of counting instantiations is d^m, where d is the maximum size of the domain of the first-order variables. Thus counting instantiations has parametrized polynomial complexity [5], meaning that if m can be treated as a constant, then counting instantiations requires only polynomially many operations in the size of the relational structure (i.e., the size of $T^* = t, \Lambda^*$ in Equation 1). For general m, the problem of computing the number of formula instantiations is #P-complete [4, Prop.12.4].

5.2 Consistency

A basic question in the theory of dependency networks is the *consistency* of the local conditional Gibbs probabilities. This means that there is a single joint distribution P over all nodes that agrees with the local probabilities. In symbols, this entails that

$$P(T^* = t|\Lambda^*) = P(T^* = t, \Lambda^*)$$

for all target nodes T^* and query conjunctions Λ^*. Dependency networks learned from data are almost always inconsistent but nonetheless provide accurate predictions [9, 17]. Heckermann *et al.* show that a dependency network is consistent if and only if there is a Markov network with the same graphical structure that agrees with the local conditional distributions [9]. A sufficient condition for the consistency of an RDN derived from a BN is therefore that all relevant family counts are the same for all members of all ground families.

On the other hand, without an equality constraint on relevant family counts, an RDN derived from a BN is generally not consistent. A sufficient and necessary condition for when different relevant counts may occur with different ground nodes is that a parent and a child contain different population variables.

Theorem 1. *Assume that a template BN contains at least one edge such that the parent and child do not contain the same set of population variables. Then there exist a query conjunction Λ^* and two ground nodes T_1^* and T_2^* such that the conditional distributions $\theta(T_1^*|\Lambda^*)$ and $\theta(T_2^*|\Lambda^*)$ defined by Equation 1 are mutually inconsistent.*

The proof of this result is complex, therefore we present it in an appendix. Intuitively, in a joint distribution, the correlation or potential of an edge is a single fixed quantity, whereas in Equation 1, the correlation is adjusted by the size of the relational neighbourhood of the parent resp. child node. If the relational neighborhoods are of different sizes, so are the relevant family counts.

6 Empirical Evaluation

We compare learning RDNs via Bayesian networks with learning via boosted functional gradient methods. Boosting methods follow the traditional approach to learning dependency networks, which is to learn a collection of separate discriminative models, one for each node in the network [9]. Boosted functional gradient methods have been shown to perform well on small datasets [12, 15]; our experiments extend these results to medium–large datasets.

6.1 Experimental Conditions and Metrics

All experiments were done on a machine with 8 GB of RAM and a single Intel Core 2 QUAD Processor Q6700 with a clock speed of 2.66 GHz (there is no hyper-threading on this chip), running Linux Centos 2.6.32. Code was written in Java, JRE 1.7.0. All code and datasets are available [11].

Datasets We used six benchmark real-world databases. For more details please see the references in [23]. Summary statistics are given in Table 2.

MovieLens MovieLens is a commonly-used rating dataset.[2] It contains two entity sets, Users and Movies. For each user and movie that appears in the database, all available ratings are included. MovieLens(1M) contains 1 M ratings, 3,883 Movies, and 6,039 Users. MovieLens(0.1M) contains about 0.1 M ratings, 1,682 Movies, and 941 Users. We did not use the binary genre predicates because they are easily learned with exclusion rules.

Mutagenesis This dataset is widely used in inductive logic programming research. It contains information on Atoms, Molecules, and Bonds between them. We use the discretization of Schulte and Khosravi [23].

Hepatitis 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 Data from multiple geographical Web data sources.

UW-CSE This dataset lists facts about the Department of Computer Science and Engineering at the University of Washington, such as entities (e.g., $Person$, $Course$) and their relationships (e.g., $AdvisedBy$).

IMDb The largest dataset in terms of number of total tuples (more than 1.3M) and schema complexity. It combines MovieLens with data from the Internet Movie Database (IMDb)[3] [19].

Methods Compared Functional gradient boosting is a state-of-the-art method for applying discriminative learning to build a generative graphical model. The local discriminative models are ensembles of relational regression trees [12]. Functional gradient boosting for relational data is implemented in the Boostr system [13]. For functors with more than two possible values, we followed [12] and converted each such functor to a set of binary predicates by introducing a predicate for each possible value. We compared the following methods:

RDN_Bayes Our method: Learn a Bayesian network, then convert it to a relational dependency network.

RDN_Boost The RDN learning mode of the Boostr system. Information from ground nodes linked to the target is aggregated with functions $count, max, average$ and existential quantification [15].

MLN_Boost The MLN learning mode of the Boostr system. It takes a list of target predicates for analysis. We provide each binary predicate in turn as a single target predicate, which amounts to using MLN learning to construct an RDN. This RDN uses a log-linear model for Gibbs conditional probabilities that is derived from Markov Logic Networks.

We used the default Boostr settings. We experimented with alternative settings but they did not improve the performance of the boosting methods.

[2] www.grouplens.org
[3] www.imdb.com, July 2013

To obtain the BN structure for RDN_Bayes, the learn-and-join algorithm [23] was applied to each benchmark database. The BN parameters can be estimated by applying the maximum likelihood principle, using the conditional frequencies observed in a relational database [22, 24]. These were computed using previously-published algorithms for multi-relational data [24].

Prediction Metrics We follow [12] and evaluate the algorithms using conditional log likelihood (CLL) and area under the precision-recall curve (AUC-PR). AUC-PR is appropriate when the target predicates features a skewed distribution as is typically the case with relationship predicates. For each fact $T^* = t$ in the test dataset, we evaluate the accuracy of the predicted Gibbs probability $P(T^* = t|\Lambda^*)$, where Λ^* is a complete conjunction for all ground terms other than T^*. Thus Λ^* represents the values of the input variables as specified by the test dataset. CLL is the average of the logarithm of the Gibbs probability for each ground truth fact in the test dataset. For the gradient boosting method, we used the AUC-PR and likelihood scoring routines included in Boostr.

Both metrics are reported as means and standard deviations over all binary predicates. The learning methods were evaluated using 5-fold cross-validation. Each database was split into 5 folds by randomly selecting entities from each entity table, and restricting the relationship tuples in each fold to those involving only the selected entities (i.e., subgraph sampling [23]). The models were trained on 4 of the 5 folds, then tested on the remaining one.

6.2 Results

Learning Times. Table 2 shows learning times for the methods. The Bayesian network learning simultaneously learns a joint model for all predicates. For the boosting method, we added together the learning times for each target predicate. On MovieLens(1M), the boosting methods take over 2 days to learn a classifier for the relationship B_U2Base, so we do not include learning time for this predicate for any boosting method. On the largest database, IMDb, the boosting methods cannot learn a classifier for the three relationship predicates with our system resources, so we only report learning time for descriptive attributes by the boosting methods. Likewise, our accuracy results in Tables 3 and 4 include only measurements for descriptive attributes on the datasets IMDb and MovieLens(1M).

Table 2 shows that RDN_Bayes scales very well with the number of data tuples: even the MovieLens dataset with 1 M records can be analyzed in seconds. This is because it provides closed-form parameter estimation and hence closed-form model scoring. RDN_Bayes is less scalable with the number of predicate, since it learns a joint model over all predicates simultaneously, although the time remains feasible (1–3 hours for 17–19 predicates; see also [23]). By contrast, the boosting methods that learn separate discriminative models scale well with the number of predicates, consistent with findings from propositional learning [9]. However, model evaluation itself is quite expensive for the boosting methods.

Table 2. Learning Time. The total learning time for constructing a relational dependency network from an input database. Only partial boosting learning times are reported for the larger databases MovieLens(1M) and IMDb—see text for details. Spread is reported as coefficient of variation (standard deviation / mean).

			RDN_Bayes		RDN_Boost		MLN_Boost	
Dataset	kTuple	Preds.	(s)	CV	(s)	CV	(s)	CV
UW	0.6	14	**14**	0.00	237	0.06	329	0.16
Mondial	0.9	18	1836	0.07	**369**	0.06	717	0.05
Hepatitis	11.3	19	5434	0.01	6648	0.02	**3197**	0.04
Mutagenesis	24.3	11	**11**	0.00	1342	0.04	1040	0.02
MovieLens(0.1M)	83.4	7	**8**	0.07	3019	0.04	3292	0.01
MovieLens(1M)	1010.1	7/6	**8**	0.09	32230	0.04	25528	0.04
IMDb	15538.4	17/13	**9346**	0.22	78129	0.04	29704	0.03

Table 3. Conditional Log-Likelihood: Mean (top), Std. Dev. (bottom)

					MovieLens		
Method	UW	Mond.	Hepa.	Muta.	(0.1M)	(1M)	IMDb
RDN_Boost	-0.30	-0.48	-0.48	-0.36	-0.50	**-0.22**	**-0.49**
MLN_Boost	-0.14	-0.40	-0.49	-0.23	-0.50	-0.23	**-0.49**
RDN_Bayes	**-0.01**	**-0.25**	**-0.39**	**-0.22**	**-0.30**	-0.28	-0.51
RDN_Boost	0.02	0.03	0.01	0.02	0.01	0.00	0.00
MLN_Boost	0.01	0.05	0.01	0.02	0.01	0.00	0.00
RDN_Bayes	0.00	0.06	0.10	0.07	0.00	0.00	0.00

Unlike propositional i.i.d. data, relational data are represented in multiple tables, so evaluation requires expensive combining of information from different tables [17]. Consequently, learning Bayesian networks explores a more complex model space than the boosting approaches to learning RDNs, but is typically much faster due to its more efficient model evaluation.

Accuracy. Whereas learning times are evaluated on all predicates, we evaluate accuracy on binary predicates (e.g., *gender*, *Borders*) because the boosting methods are based on binary classification. By the likelihood metric (Table 3), the Bayesian network method performs best on four datasets, comparably to

Table 4. Area Under Precision-Recall Curve: Mean (top), Std. Dev. (bottom).

					MovieLens		
Method	UW	Mond.	Hepa.	Muta.	(0.1M)	(1M)	IMDb
RDN_Boost	0.42	0.27	0.55	0.71	0.50	0.88	0.63
MLN_Boost	0.68	0.44	0.55	**0.86**	0.50	0.88	0.63
RDN_Bayes	**0.89**	**0.79**	0.55	0.50	**0.65**	**1.00**	**0.85**
RDN_Boost	0.00	0.00	0.01	0.02	0.01	0.00	0.01
MLN_Boost	0.01	0.04	0.01	0.04	0.01	0.00	0.01
RDN_Bayes	0.00	0.07	0.11	0.10	0.02	0.00	0.00

MLN_Boost on Mutagenesis, and slightly worse than both boosting methods on the two largest datasets. By the precision-recall metric (Table 4), the Bayesian network method performs substantially better on four datasets, identically on Hepatitis, and substantially worse on Mutagenesis.

Combining these results, for most of our datasets the Bayesian network method has comparable accuracy and much faster learning. This is satisfactory because boosting is a powerful method that achieves accurate predictions by producing a tailored discriminative model for each target predicate—a computationally expensive process. By contrast, Bayesian network learning uses the simpler process of simultaneously constructing a joint model for all predicates, and uses simple maximum likelihood estimation for parameter values. We conclude that *Bayesian network learning scales much better to large datasets, and provides competitive accuracy in predictions.*

In addition to scalability, Bayesian networks offer two more advantages. First, learning easily extends to attributes with more than two possible values. Second, the parameters and the predictions derived from them are much easier to interpret than an ensemble of regression trees [15].

6.3 Comparison of Model Structures

Boosting is known to lead to very accurate classification models [2]. For propositional data, a Bayesian network classifier with maximum likelihood estimation for parameter values is a reasonable baseline method [8], but we would expect less accuracy than from a boosted ensemble of regression trees. Therefore the predictive performance of our RDN models is not due to the Bayesian network equation, but due to the more powerful features that Bayesian network learning finds in relational datasets. Table 5 reports results that quantitatively confirm this analysis.

For each database, we selected the target predicate where RDN-Bayes shows the greatest predictive advantage over RDN-Boost (shown as Δ CLL and Δ AUC-PR). We then compute how many more predicates the RDN-Bayes model uses to predict the target predicate than the RDN-Boost model, shown as Δ Predicates. This number can be as high as 11 more predicates (for Mondial). We also compare how many more population variables are contained in the Markov blanket of the RDN-Bayes model, shown as Δ Variables. In terms of database tables, the number of population variables measures how many related tables are used for prediction in addition to the target table. This number can be as high as 2 (for IMDb and Hepatitis). To illustrate Figure 3 shows the parents (Markov blanket) of target node $gender(\mathbb{U})$ from IMDb in the RDN-Boost and RDN-Bayes models. The RDN-Bayes model introduces 4 more parents and 2 more variables, *Movie* and *Actor*. These two variables correspond to a relationship chain of length 2. Thus BN learning discovers that the gender of a user can be predicted by the gender of actors that appear in movies that the user has rated.

Table 5. Difference in Markov blankets between RDN_Bayes and RDN_Boost. Δx = (x for RDN_Bayes - x for RDN_Boost). RDN_Bayes predicts a target more successfully because it uses more predicates and those predicates contain more first-order variables.

Database	Target	Δ Preds.	Δ Vars.	Δ CLL	Δ AUC-PR
Mondial	religion	11	1	0.58	0.30
IMDb	gender	4	2	0.30	0.68
UW-CSE	student	4	1	0.50	0.55
Hepatitis	sex	4	2	0.20	0.25
Mutagenesis	ind1	5	1	0.56	0.22
MovieLens	gender	1	1	0.26	0.26

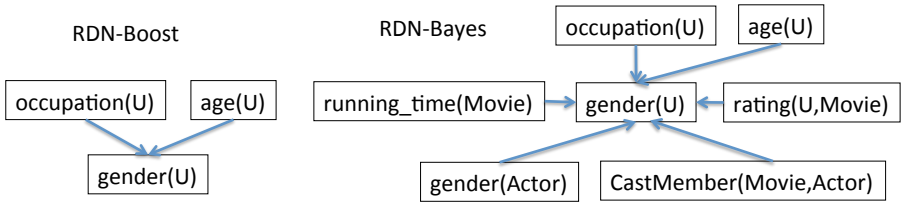

Fig. 3. The parents of target *gender*($\mathbb{U}$) in the models discovered by RDN_Boost (left) and RDN_Bayes (right).

7 Related Work

Dependency networks were introduced by Heckerman et al. [9] and extended to relational data by Neville and Jensen [17]. Heckerman et al. compare Bayesian, Markov and dependency networks for nonrelational data [9].

Bayesian networks. There are several proposals for defining directed relational template models, based on graphs with directed edges or rules in clausal format [10, 6]. Defining the probability of a child node conditional on multiple instantiations of a parent set requires the addition of combining rules [10] or aggregation functions [6]. Combining rules such as the arithmetic mean [16] combine global parameters with a local scaling factor, as does our log-linear model. In terms of combining rules, our model uses the *geometric mean* rather than the arithmetic mean.[4] To our knowledge, the geometric mean has not been used before as a combining rule for relational data.

Markov Networks. Markov Logic Networks (MLNs) provide a logical template language for undirected graphical models. Richardson and Domingos propose transforming a Bayesian network to a Markov Logic network using moralization, with log-conditional probabilities as weights [3]. This is also the standard BN-to-MLN transformation recommended by the Alchemy system [1]. A discriminative

[4] The geometric mean of a list of numbers $x_1, \ldots, x_n$ is $(\prod_i x_i)^{1/n}$. Thus geometric mean = exp(average (logs)).

model can be derived from any MLN [3]. The structure transformation was used in previous work [23], where MLN parameters were learned, not computed in closed-form from BN parameters. The Gibbs conditional probabilities derived from an MLN obtained from converting a Bayesian network are the same as those defined by our log-linear Formula 1, *if* counts replace proportions as feature functions [22]. There is no MLN whose discriminative model is equivalent to our log-linear equation with proportions as feature functions.[5]

8 Conclusion and Future Work

Relational dependency networks offer important advantages for modelling relational data. We proposed a novel approach to learning dependency networks: first learn a Bayesian network, then perform a closed-form transformation of the Bayesian network to a dependency network. The key question is how to transform BN parameters to DN parameters. We introduced a new relational adaptation of the standard BN log-linear equation for the probability of a target node conditional on an assignment of values to its Markov blanket. The new log-linear equation uses a sum of expected values of BN log-conditional probabilities, with respect to a random instantiation of first-order variables. This is equivalent to using feature instantiation proportions as feature functions. We compared our approach to state-of-the-art functional gradient boosting methods on six benchmark datasets. Learning RDNs via BNs scales much better to large datasets than with boosting, and provides competitive accuracy in predictions.

The boosting approach to constructing a dependency network by learning a collection of discriminative models is very different from learning a Bayesian network. There are various options for hybrid approaches that combine the strengths of both. (1) Fast Bayesian network learning can be used to select features. Discriminative learning methods should work faster restricted to the BN Markov blanket of a target node. (2) The Bayesian network can provide an initial dependency network structure. Gradient boosting can then be used to fine-tune a discriminative model of a child node given parent nodes, replacing a flat conditional probability table.

Appendix: Proof of Consistency Characterization

We show that for a given template BN, there are two ground target nodes and query conjunction Λ^* such that the conditional distributions of the ground target nodes given Λ^* do not agree with any joint distribution over the ground target nodes given Λ^*. We begin by establishing some properties of the template BN and the query conjunction that are needed in the second part of the proof. The second part proves the inconsistency by showing that consistency entails a constraint that is violated by the template BN for the constructed query conjunction Λ^*.

[5] A preliminary version of this paper was presented at the StarAI 2012 workshop. A second version of this paper was presented at ILP 2014 but not published in the conference proceedings.

8.1 Properties of the template BN and the input query Λ^*

The inconsistency of the BN networks arises when a parent and a child ground node have different relevant family counts. The next lemma shows that this is possible exactly when the template BN is properly relational, meaning it relates parents and children from different populations.

Lemma 1. *The following conditions are equivalent for a template edge $T_1 \to T_2$.*

1. *The parent and child do not contain the same population variables.*
2. *It is possible to find a grounding γ for both parent and child, and an assignment Λ^* to all other nodes, such that the relevant family count for the T_2 family differs for $T_1^* = \gamma T_1$ and $T_2^* = \gamma T_2$.*

Proof. If the parent and child contain the same population variables, then there is a 1-1 correspondence between groundings of the child and groundings of the parents. Hence the count of relevant family groundings is the same for each, no matter how parents and child are instantiated. If the parent and child do not contain the same population variables, suppose without loss of generality that the child contains a population variable $\mathbb{A}$ not contained in the parent. Choose a common grounding γ for the parents and child node. For the ground child node, γT_2, let γ be the only family grounding that is relevant, so the relevant count is 1. For the ground parent node, there is at least one other grounding of the child node T_2' different from γT_2 since T_2 contains another population variables. Thus it is possible to add another relevant family grounding for γT_1, which means that the relevant count is at least 2.

The proof proceeds in the most simple manner if we focus on template edges that different populations and have no common children.

Definition 2. *An template edge $T_1 \to T_2$ is **suitable** if*

1. *The parent and child do not contain the same population variables.*
2. *The parent and child have no common edge.*

The next lemma shows that focusing on suitable edges incurs no loss of generality.

Lemma 2. *Suppose that a template BN contains an edge such that the parent and child do not contain the same population variables. Then the template BN contains a suitable edge.*

Proof. Suppose that there is an edge satisfying the population variable condition. Suppose that the parent and child share a common child. Since the edge satisfies the condition, the set of population variables in the common child differs from at least one of T_1, T_2. Therefore there is another edge from one of $T_1 \to T_2$ as parent to a new child that satisfies the population variable condition. If this edge is not suitable, there must be another shared child. Repeating this argument, we eventually arrive at an edge satisfying the population variable condition where the child node is a sink node without children. This edge is suitable.

Consider a suitable template edge $T_1 \rightarrow T_2$ that produces a bidirected ground edge $T_1^* \leftrightarrow T_2^*$. For simplicity we assume that T_1 and T_2 are binary variables with domain $\{\mathrm{T}, \mathrm{F}\}$. (This incurs no loss of generality as we can choose a database Λ^* in which only two values occur.) Let $\mathrm{Pa}(T_2)$ be the parents of T_2 other than T_1. Since the template edge is not redundant [18], there is a parent value setting $\mathrm{Pa}(T_2) = \mathbf{pa}$ such that T_1 and T_2 are conditionally dependent given $\mathrm{Pa}(T_2) = \mathbf{pa}$. This implies that the conditional distribution of T_1 is different for each of the two possible values of T_2:

$$\frac{\theta(T_2 = \mathrm{F}|T_1 = \mathrm{F}, \mathbf{pa})}{\theta(T_2 = \mathrm{T}|T_1 = \mathrm{F}, \mathbf{pa})} \neq \frac{\theta(T_2 = \mathrm{F}|T_1 = \mathrm{T}, \mathbf{pa})}{\theta(T_2 = \mathrm{T}|T_1 = \mathrm{T}, \mathbf{pa})}. \tag{1}$$

Let Λ^* denote an assignment of values to all ground nodes other than the target nodes T_1^* and T_2^*. We assume that the input query Λ^* assigns different relevant family counts N_1 to T_1^* and N_2 to T_2^*. This is possible according to Lemma 1.

8.2 Lowd's Equation and Relevant Family Counts

The log-linear equation 1, specifies the conditional distribution of each target node given Λ^* and a value for the other target node. We keep the assignment Λ^* fixed throughout, so for more compact notation, we abbreviate the conditional distributions as

$$p({T_1}^* = t_1|{T_2}^* = t_2) \equiv P({T_1}^* = t_1|{T_2}^* = t_2, \Lambda^*)$$

and similarly for $P({T_1}^* = t_1|{T_2}^* = t_2, \Lambda^*)$.

On the assumption that the dependency network is consistent, there is a joint distribution over the target nodes conditional on the assignment that agrees with the conditional distribution:

$$\frac{p({T_1}^* = t_1, {T_2}^* = t_2)}{p({T_2}^* = t_2)} = p({T_1}^* = t_1|T_2)^*$$

and also with the conditional $p({T_2}^* = t_2|{T_1}^* = t_1)$.

Lowd [14] pointed out that this joint distribution satisfies the equations

$$\frac{p(\mathrm{F},\mathrm{F})}{p(\mathrm{T},\mathrm{F})} \cdot \frac{p(\mathrm{T},\mathrm{F})}{p(\mathrm{T},\mathrm{T})} = \frac{p(\mathrm{F},\mathrm{F})}{p(\mathrm{T},\mathrm{T})} = \frac{p(\mathrm{F},\mathrm{F})}{p(\mathrm{F},\mathrm{T})} \cdot \frac{p(\mathrm{F},\mathrm{T})}{p(\mathrm{T},\mathrm{T})} \tag{2}$$

Since the ratio of joint probabilities is the same as the ratio of conditional probabilities for the same conditioning event, consistency entails the following constraint on conditional probabilities via Equation (2):

$$\frac{p({T_2}^* = \mathrm{F}|{T_1}^* = \mathrm{F})}{p({T_2}^* = \mathrm{T}|{T_1}^* = \mathrm{F})} \cdot \frac{p({T_1}^* = \mathrm{F}|T_2^* = \mathrm{T})}{p({T_1}^* = \mathrm{T}|{T_2}^* = \mathrm{T})} = \frac{p({T_1}^* = \mathrm{F}|{T_2}^* = \mathrm{F})}{p({T_1}^* = \mathrm{T}|{T_2}^* = \mathrm{F})} \cdot \frac{p({T_2}^* = \mathrm{F}|{T_1}^* = \mathrm{T})}{p({T_2}^* = \mathrm{T}|{T_1}^* = \mathrm{T})} \tag{3}$$

We refer to Equation 3 as *Lowd's equation*. The idea of our proof is to show that Lowd's equations are satisfied only if the relevant family counts for the

target nodes are the same. According to the log-linear equation, each conditional probability is proportional to a product of BN parameters. The first step is to show that in Lowd's equation, all BN parameter terms cancel out except for those that are derived from the family that comprises T_1^* and their T_2^* and their common grounding.

Lemma 3. *The conditional probabilities for the target nodes can be written as follows:*

$$P({T_2}^* = t_2 | T_1^* = t_1, \Lambda^*) \propto \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa})^{(N/N_2 + M_{T_2=t_2}/N_2)} \cdot \pi_{T_2=t_2} \quad (4)$$

where $M_{T_2=t_2}$ *and* $\pi_{T_2=t_2}$ *depend only on* t_2 *and not on* t_1 *and*

$$P({T_1}^* = t_1 | T_2^* = t_2, \Lambda^*) \propto \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa})^{(N/N_1 + M_{T_1=t_1}/N_1)} \cdot \pi_{T_1=t_1} \quad (5)$$

where $M_{T_1=t_1}$ *and* $\pi_{T_1=t_1}$ *depend only on* t_1 *and not on* t_2*.*

Proof Outline. This is based on analysing the different types of families that appear in the log-linear equation and their groundings. We omit the full proof due to space reasons; it is available from [25].

Lemma 4. *Suppose that conditions* (4) *and* (5) *of Lemma 3 hold. Then Lowd's Equation* (3) *holds if and only if* $N_1 = N_2$*.*

Proof. Observe that in Equation (3), each term on the left has a corresponding term with the same value for the target node assignment and the opposing conditioning assignment. For instance, the term $p({T_2}^* = \mathrm{F} | {T_1}^* = \mathrm{F})$ on the left is matched with the term $p({T_2}^* = \mathrm{F} | {T_1}^* = \mathrm{T})$ on the right. This means that the products in the log-linear expression are the same on both sides of the equation except for those factors that depend on *both* t_1 and t_2. Continuing the example, the factors

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{F}, \mathbf{pa})^{(M_\mathrm{F}/N_2)} \cdot \pi_{T_2=t_2}$$

on the left equal the factors

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{T}, \mathbf{pa})^{(M_{T_1=t_1}/N_2)} \cdot \pi_{T_2=t_2}$$

on the right side of the equation. They therefore cancel out, leaving only the term

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{F}, \mathbf{pa})^{N/N_2}$$

on the left and the term

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{F}, \mathbf{pa})^{N/N_2}$$

on the right. Lowd's equation can therefore be reduced to an equivalent constraint with only such BN parameter terms. For further compactness we abbreviate such terms as follows

$$\theta(t_2 | t_1) \equiv \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa}).$$

With this abbreviation, the conditions of Lemma 3 entail that Lowd's equation 3 reduces to the equivalent expressions.

$$\frac{\theta(\mathrm{F}|\mathrm{F})^{N/N_2}}{\theta(\mathrm{T}|\mathrm{F})^{N/N_2}} \cdot \frac{\theta(\mathrm{T}|\mathrm{F})^{N/N_1}}{\theta(\mathrm{T}|\mathrm{T})^{N/N_1}} = \frac{\theta(\mathrm{F}|\mathrm{F})^{N/N_1}}{\theta(\mathrm{F}|\mathrm{T})^{N/N_1}} \cdot \frac{\theta(\mathrm{F}|\mathrm{T})^{N/N_2}}{\theta(\mathrm{T}|\mathrm{T})^{N/N_2}} \tag{6}$$

$$\left(\frac{\theta(\mathrm{F}|\mathrm{F})}{\theta(\mathrm{T}|\mathrm{F})}\right)^{(N/N_2 - N/N_1)} = \left(\frac{\theta(\mathrm{F}|\mathrm{T})}{\theta(\mathrm{T}|\mathrm{T})}\right)^{(N/N_2 - N/N_1)} \tag{7}$$

By the nonredundancy assumption (1) on the BN parameters, we have

$$\frac{\theta(\mathrm{F}|\mathrm{F})}{\theta(\mathrm{T}|\mathrm{F})} \neq \frac{\theta(\mathrm{F}|\mathrm{T})}{\theta(\mathrm{T}|\mathrm{T})}$$

so Equation 7 implies that

$$N_1 = N_2,$$

which establishes the lemma.

The main theorem now follows as follows: Lemma 1 entails that if the dependency network is consistent, the log-linear equations satisfy Lowd's equation with the bidirected ground edge $T_1^* \leftrightarrow T_2^*$ and the query conjunction Λ^* that satisfies the BN non-redundancy condition. Lemmas 3 and 2 show that if the template BN is relational, it must contain a suitable edge $T_1 \rightarrow T_2$. Lemma 4 together with Lowd's equation entails that the relevant counts for T_1^* and T_2^* must then be the same. But the query conjunction Λ^* was chosen so that the relevant counts are different. This contradiction shows that Lowd's equation is unsatisfiable, and therefore no joint distribution exists that is consistent with the BN conditional distributions specified by the log-linear Equation 1.

References

1. Alchemy Group. Frequently asked questions. URL = `http://alchemy.cs.washington.edu/`.
2. Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
3. Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.
4. Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* [7].
5. Jörg Flum and Martin Grohe. *Parameterized complexity theory*, volume 3. Springer, 2006.
6. Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [7], chapter 5, pages 129–173.
7. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

8. Daniel Grossman and Pedro Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *ICML*, page 46, New York, NY, USA, 2004. ACM.
9. David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, Carl Kadie, and Pack Kaelbling. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1:49–75, 2000.
10. Kristian Kersting and Luc de Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [7], chapter 10, pages 291–318.
11. H. Khosravi, T. Man, J. Hu, E. Gao, and O. Schulte. Learn and join algorithm code. URL = `http://www.cs.sfu.ca/~oschulte/jbn/`.
12. Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, pages 320–329, 2011.
13. Tushar Khot, Jude Shavlik, and Sriraam Natarajan. Boostr, 2013. URL = `http://pages.cs.wisc.edu/~tushar/Boostr/`.
14. D. Lowd. Closed-form learning of Markov networks from dependency networks. In *UAI*, 2012.
15. Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.
16. Sriraam Natarajan, Prasad Tadepalli, Thomas G. Dietterich, and Alan Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artifical Intelligence*, 54(1-3):223–256, 2008.
17. Jennifer Neville and David Jensen. Relational dependency networks. *JMLR*, 8:653–692, 2007.
18. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
19. Veronika Peralta. Extraction and integration of MovieLens and IMDB data. Technical report, Laboratoire PRiSM, Universite de Versailles, 2007.
20. David Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.
21. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.
22. Oliver Schulte. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.
23. Oliver Schulte and Hassan Khosravi. Learning graphical models for relational data via lattice search. *Machine Learning*, 88(3):331–368, 2012.
24. Oliver Schulte, Hassan Khosravi, Arthur Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with bayes nets. *Machine Learning*, 94:105–125, 2014.
25. Oliver Schulte, Zhensong Qian, Arthur E. Kirkpatrick, Xiaoqian Yin, and Yan Sun. Fast learning of relational dependency networks. *CoRR*, abs/1410.7835, 2014.
26. Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning* [7], chapter 4, pages 93–127.
27. Moshe Y. Vardi. On the complexity of bounded-variable queries. In *PODS*, pages 266–276, 1995.