Fast Learning of Relational Dependency Networks for Large Complex Datasets

Oliver Schulte OSCHULTE@CS.SFU.CA
Arthur E. Kirkpatrick TED@SFU.CA
School of Computing Science, Simon Fraser University
Vancouver-Burnaby, Canada

Yuke Zhu YUKEZ@STANFORD.EDU
Computer Science Department, Stanford University
Stanford, California, United States

Zhensong Qian ZQIAN@SFU.CA
School of Computing Science, Simon Fraser University
Vancouver-Burnaby, Canada

Tianxiang Gao TGAO@CS.UNC.EDU
Department of Computer Science, University of North Carolina at Chapel Hill
Chapel Hill, North Carolina, United States

Abstract

A Relational Dependency Network (RDN) is a directed graphical model widely used for multi-relational data. RDN structures allow cyclic dependencies that represent relational autocorrelations. We describe a new approach for learning both the structure and the parameters of an RDN, given an input relational database. The basic idea is to first learn a Bayesian network (BN), then apply a closed-form transformation to convert the Bayesian network to an RDN. Thus fast Bayes net learning leads to fast RDN learning. The BN-to-RDN transform computes a conditional random field for each RDN node. The conditional random field is defined by a log-linear equation, whose features are computed from the BN structure and whose weights are computed from the BN parameters. For evaluation, we empirically compare our approach to state-of-the art RDN learning methods that use functional gradient boosting. [experimental details]

1. Introduction

Learning graphical models is one of the main approaches to extending machine learning for relational data. Dependency networks (DNs) (?) are one of the major classes of graphical generative models, together with Markov networks and Bayesian networks (BNs) (?). We describe a new approach to learning dependency networks: first learn a Bayesian network, then convert the Bayesian network to a dependency network. This hybrid approach combines the advantages of both model classes: fast scalable learning for Bayesian networks, principled and accurate inference for relational dependency networks. We propose a novel closed-form algorithm for computing the structure and parameters of relational dependency networks from the structure and parameters of Bayesian networks. The hybrid learning algorithm produces dependency networks with accurate predictions for large complex databases, with millions of records and many predicates [how many exactly?].

2. Relational Dependency Networks and Bayesian Networks

We briefly review the definition of dependency networks and their advantages for modelling relational data.

2.1 Dependency networks and Bayesian networks

Like Bayesian networks, the structure of a dependency network is defined by a graph whose nodes are random variables and whose edges are directed. Unlike Bayesian networks, a dependency network graph may contain cycles and 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. The difference lies in the characteristic independence property of dependency networks: each node is independent of all other nodes given an assignment of values to its parents. In contrast, given an assignment of values to its Bayes net parents, there remains a dependence between a node and its children. In graphical model terms, the parents of a node in a dependency network form a Markov blanket: Given an assignment of values to the Markov blanket of a node, the node is independent of all other nodes in the network. A dependency network parameter is therefore equivalent to specifying the probability of a node value given an assignment of values to all other nodes. Such conditional probabilities play an important role in statistical inference, for example in the widely used Gibbs sampling procedure (?). We therefore refer to them as **Gibbs conditional probabilities**, or simply Gibbs probabilities.[1] Table 1 summarizes the differences in the two model classes. Figure 1 illustrates a Bayesian network and Figure 2 a dependency network. An example of a Gibbs conditional probability distribution for the inference graph of Figure 2 is

$$P(gender(anna)|gender(bob), CoffeeDr(anna), Friend(anna, bob), Friend(bob, anna), Friend(anna, anna)).$$

We next discuss the advantages of dependency networks for relational data.

Table 1: Dependency Networks vs. Bayesian Networks

Model Class	Bayesian Networks	Dependency Networks
Structure	Directed Acyclic Graph	Directed Graph - Cycles Allowed
Parameters	Probability of Node Given Parents	Probability of Node Given Markov Blanket
Structure Advantage	Fast Learning	Accommodates relational autocorrelations.
Parameter Advantage	Learning: Closed-Form Estimation	Inference aggregates entire Markov blanket

1. In the terminology of dependency networks (?), Gibbs probabilities are referred to as local probability distributions. The WinBUGS system refers to them as full conditional probabilities (?).

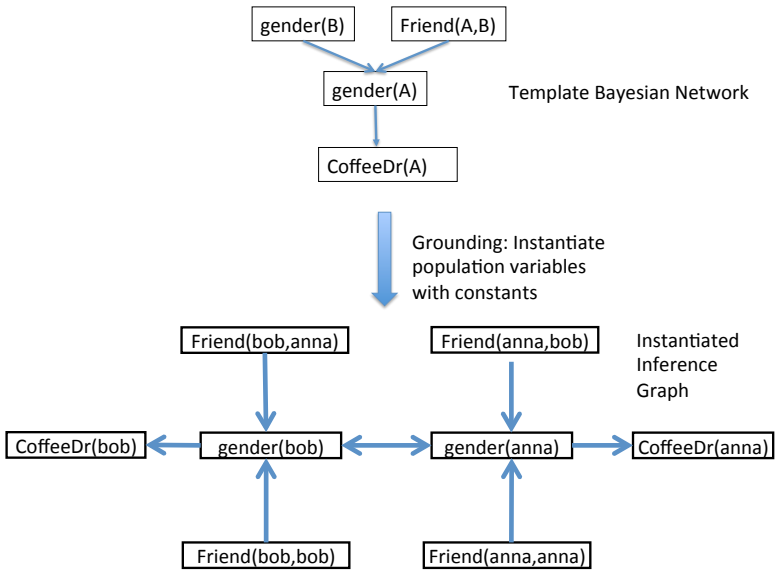

Figure 1: A Bayesian network template model (top) and the instantiated inference graph (bottom), with two individuals Anna and Bob in the domain of the first-order or population variables $\mathbb{A}, \mathbb{B}$. Population variables are to be instantiated with all constants that denote a member of the applicable set. The two instantiations $\mathbb{A} \backslash anna, \mathbb{B} \backslash bob$ and $\mathbb{A} \backslash bob, \mathbb{B} \backslash anna$ produce a cycle involving the two edges $gender(anna) \rightarrow gender(bob)$ and $gender(bob) \rightarrow gender(anna)$. **Add BN parameters as well?**

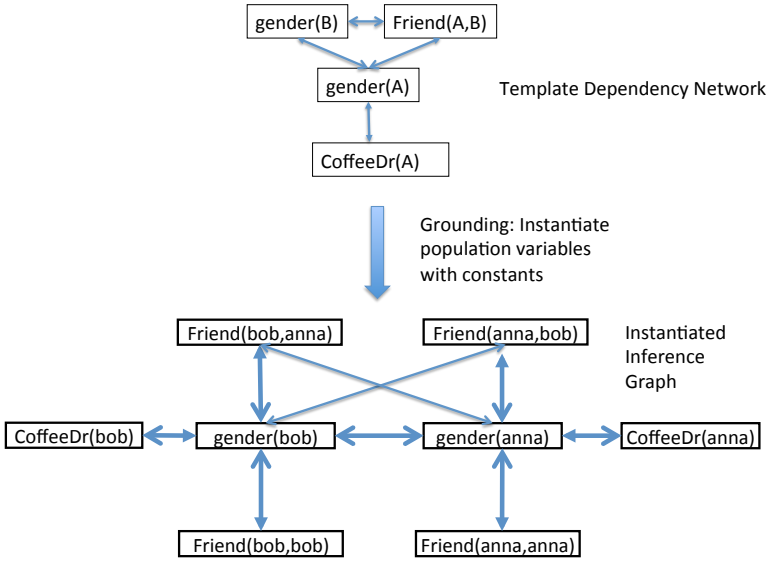

Figure 2: A dependency network template model (top) and the instantiated inference graph (bottom). The Gibbs probability parameters for the inference graph specify the conditional probability of a node given its neighbors (Markov blanket).

2.2 Relational Dependency Networks

Relational Dependency Networks have received a significant amount of recent development from researchers (?, ?). Like our paper, this work follows the traditional knowledge-based model construction approach (KBMC). In the KBMC approach, a template graphical model is instantiated for a specific domain of individuals to produce an *instantiated* or *ground* graphical model, called the inference graph by Neville and Jensen (?). Figures 1 and 2 illustrate Bayesian/dependency network templates and their groundings.

The advantage of dependency network structure for relational data is that they allow cyclic dependencies. Necessary and sufficient conditions for when a ground Bayesian network template contains cycles are given in (?). Typically these occur in the common case that a dataset features auto-correlations, where the value of an attribute for an individual depends on the values of the same attribute for related individuals (?). Figure 1 provides an example. In contrast, dependency networks allow cycles, so the grounding of a dependency network template is guaranteed to lead to a valid dependency network structure. The class of cyclic graphs is closed under the operation of grounding, whereas the class of acyclic graphs is not. This fact has been a knotty problem for defining a joint distribution using a template semantics for Bayesian networks (?, ?, ?).

The advantage of the dependency network parametrization for relational data is that conditional probabilities are defined with respect to a larger set of specified nodes. One of the key differences for prediction between relational and non-relational iid data is that *different predictors may be instantiated multiple times* (?). In the social network example

of Figure 1, suppose we want to predict the value of the ground node *gender*(*anna*). Each gender of a friend of *anna* adds one relevant predictor. The number of predictors is therefore not fixed a priori by the model, but depends on the size of the relational neighborhood, that is, the number of individuals related to the target individual. Relational prediction therefore requires aggregating information from different linked individuals. Two common approaches are using (i) combining rules (?) and (ii) aggregation functions (?). In a dependency network model, the aggregation encompasses the entire Markov blanket of a target node. In a Bayesian network model, the aggregation encompasses only its parents, not the relevant information from its children. We discuss this point further below in Section **??**.

3. Learning Dependency Networks for Relational Data via Bayesian Networks

Our aim in this paper is combine the fast learning of Bayesian networks with the inference advantages of dependency networks. We utilize state-of-the-art relational learning methods for Bayesian networks. Our novel contribution is a closed-form transformation of a Bayesian network to a relational dependency network. Converting a Bayesian network structure to a dependency network structure is simple: for each node, we add an edge pointing to the node from each member of its Bayes net Markov blanket (?). The BN Markov blanket of a node consists of its parents, children, and co-parents. The DN structure of Figure 2 results from the BN structure of Figure 1 in this manner. This is equivalent to the standard moralization method for converting a BN to an undirected model (?, 12.5.3), (?), except that the dependency network contains bi-directed edges instead of undirected edges.

For propositional iid data, converting Bayesian network parameters to dependency network parameters requires solving for the Gibbs conditional probabilities given Bayesian network parameters, which is a standard exercise (?, Ch.14.5.2). The result is the *BN log-linear equation* for each node. A discriminative log-linear equation models a conditional probability for a target node value, given an assignment of values to other variables, called the input variables. In the case of a Gibbs probability, the input variables comprise all variables other than the target node. A log-linear model requires defining a set of features, and for each feature a feature function that returns a number for that feature and for a given conditional probability to be computed (?). The parameters of the model are feature weights, one for each feature. The conditional probability of the target node value given values for the input variables is proportional to the exponentiated weighted sum of the feature functions. For propositional data, the set of features in the BN equation is the set of joint value assignments to a child and its parents in the BN structure. The weights are the log-conditional probabilities of the child value given its parent values. For propositional data, the input variables specify a unique set of values for a given child-parent configuration. But with relational data, multiple value assignments for a child-parent configuration can be instantiated multiple times. Our proposal in this paper is to accommodate multiple instantiations by using the *proportion of instantiations* that satisfy a child-parent value assignment as the feature function for the child-parent. Proportions have the desirable consequence that all feature functions are normalized to the [0,1] range. Without this normalization, features with more instantiations carry exponentially more weight. Figure 5 shows the program flow for computing a Gibbs probability using the log-linear equation.

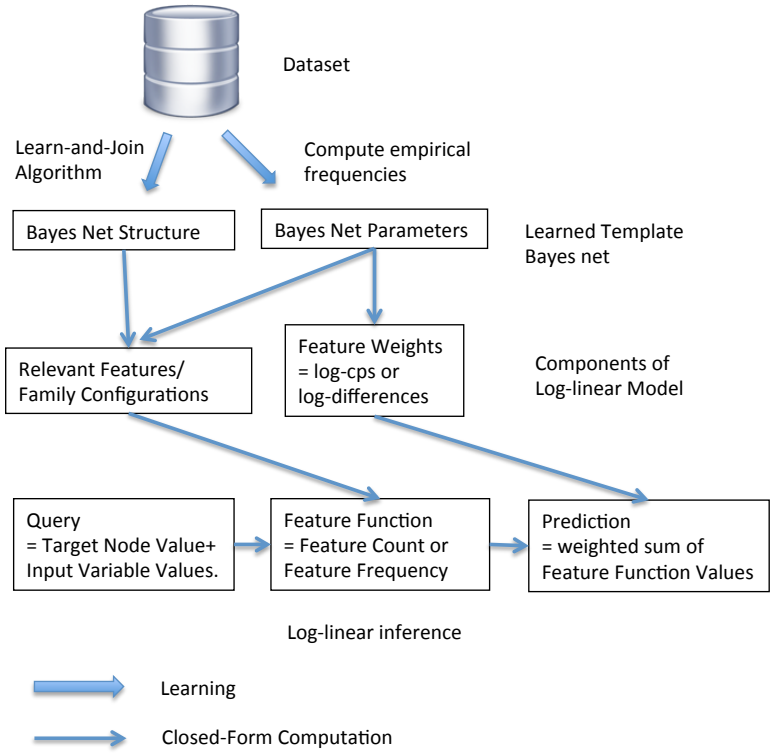

Figure 3: The program flow for computing relational Gibbs probabilities with a template Bayes net. Features and weights are computed from the Bayes net. Feature function values are computed for each query.

[Example needs fixing. Do we want one?: To illustrate, suppose we wish to compute the probability that individual $\mathbb{A} = sam$ is male, using the template BN of Figure 1 (b). The input variables specify whether Sam is a coffee drinker, which other persons are his friends, and the gender of other people. There are two parents of the gender node in this template model. One feature in the log-linear model assigns these parents the values $Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}$ and $gender(\mathbb{B}) = \mathrm{M}$. The feature function returns the proportion of Sam's friends that are male, given the values specified in the input variables. The weight associated with this feature is the log-ratio of two quantities: (i) the conditional probability that $gender(\mathbb{A}) = \mathrm{M}$, given the assignment of values to its parents, which is specified as .63 in the BN. (ii) The BN marginal or unconditional probability that $gender(\mathbb{A}) = \mathrm{M}$, which can be computed via BN inference as .55. So the weight of this feature is $w = ln(0.63/0.55) \approx 0.136$. The positive weight indicates that according to the BN template model, having a male friend raises the probability of being male.]

3.1 Evaluation

Using five standard databases, we evaluate the predictive accuracy of dependency networks obtained from Bayesian networks. The parameters of the Bayesian networks are set to be the maximum likelihood estimates. We show that Bayes net learning is fast and scales to databases with millions of records and complex schemas that contain multiple relationships and attributes. We compare predictive accuracy using counts vs. proportions as feature functions. Proportions yield substantially more accurate predictions, which indicates the importance of normalizing feature functions to a common range. The tested predictions involve different types of target predicates, describing features of individuals, the existence or absence of links/relationships between individuals, and features of links. To benchmark our approach, we compare BN-to-RDN learning with RDNBoost, the state-of-the art learning method for relational dependency networks. [Results summary]

3.2 Contributions

Our main contributions are as follows; Figure 4 provides a visual summary.

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

Previous approaches to learning dependency networks take a completely different approach, based on learning a discriminative model for each node that defines Gibbs probabilities directly. The new ideas in this paper can be combined with such approaches in various ways. (1) Functional gradient boosting is also based on a log-linear model. Current approaches use instantiation counts as feature functions. Our results suggest that using using gradient boosting with proportions would achieve better results. (2) Fast Bayesian network learning methods can be used to select features. Discriminative learning methods should work much faster restricted to the BN Markov blanket of a target node. (3) The Bayesian network can provide an initial dependency network for the boosting procedure; gradient boosting can then be used for fine-tuning.

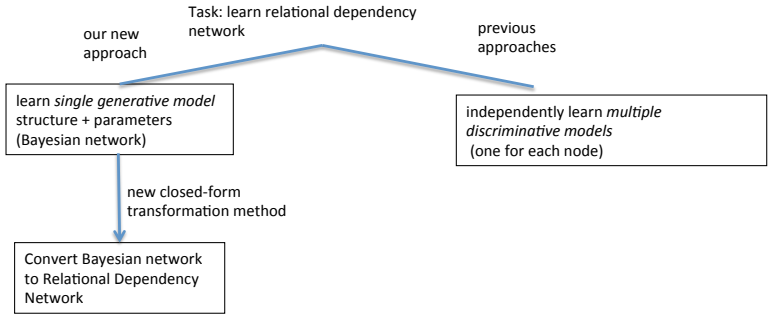

Figure 4: High-level comparison of our new approach with previous work on learning relational dependency networks.

Paper Organization. We begin (Section 5) with background and define the notation for relational Bayes net models. Section 6 presents the new log-linear relational regression model, which we evaluate on five benchmark databases in Sections 8 and 9. In Section **??** we use the same databases to compare our approach to RDN-Boost (?). We end with related work in Section 4, and conclude with suggestions for future work in Section 11.

4. Related Work

[Ted: I moved this up for easier review. We can move it back to the end.]

Dependency Network Learning Dependency networks were introduced by (?). The traditional approach to learning dependency networks is to learn a collection of discriminative models, one for each node in the network. One motivation for dependency networks is that for nonrelational data, learning multiple discriminative model is faster than learning a single generative model (e.g., a BN). The motivation for our approach is that for large relational data, the reverse is true. The fundamental reason for this is that in relational data, model evaluation is much more computationally expensive relative to model generation. Parameter estimation and model evaluation in Bayes nets can be done in closed-form using maximum likelihood, rather than through a local optimization procedure as with many discriminative models. Such local optimization procedures require many data accesses to evaluate the model at each optimization step. Relational model evaluation requires extensive counting computations for finding how often a predictive pattern is instantiated in a given relational neighborhood. In propositional data represented in a single table of cases, instantiation counts cover a subset of already existing table rows. In relational data that are represented in multiple tables, it is necessary to combine information from different tables which involves expensive cross-product. For further discussion of scalability and model evaluation cost, please see (?).

Relational dependency networks were introduced in (?). Their learning algorithm used aggregate functions to combine different instantiations of a target node's Markov blanket.

Function gradient boosting is the state-of-the-art learning methods for relational dependency networks (?). That work uses a log-linear model with counts as feature functions, not proportions.

We briefly discuss work on Bayesian networks and Markov networks for relational data. (?) *et al.* compare Bayes, Markov and dependency nets in detail for nonrelational data.

Bayesian networks There are several proposals for defining directed relational template models, based on graphs with directed edges or rules in clausal format (?, ?, ?, ?). In order to define the probability of a child node conditional on multiple instantiations of a parent set, template semantics for Bayesian networks requires the addition of combining rules (?) or aggregation functions (usually with extra parameters) (?). As described by (?), aggregate functions can be added to a Parametrized Bayesian network by including functor nodes with aggregates. Combining rules such as the arithmetic mean (?) 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.[2] To our knowledge, the geometric mean has not been used before as a combining rule for relational data. Another difference with template Bayesian networks is that the geometric mean is applied to the entire Markov blanket of the target node, whereas usually a combining rule applies only to the parents of the target node.

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 (?). This is also the standard Bayes net-to-MLN transformation recommended by the Alchemy system (?). A discriminative model can be derived from any MLN. The Gibbs conditional probabilities defined by an MLN obtained from converting a Bayesian network are the same as those defined by our log-linear equation, when *counts* are used as feature function (?). There is no MLN whose discriminative model is equivalent to our log-linear equation with *proportions* as feature functions.

5. Background: Bayes Nets for Relational Data

[Question: can we get away with having only the notation table + examples?] [Suggestion: we should probably use constrained instantiation count because of the similarity to Poole's parfactors.]

We adopt function-based notation from logic for combining Bayes nets with relational concepts (?, ?, ?, ?). Different communities in statistics and logic use different terms for similar concepts, and similar terms for different concepts. We strive for notation that is as broadly accessible as possible. Table 2 summarizes our notation for relational concepts.

5.1 Relational Concepts.

A **population** $\mathcal{P}$ is a set of individuals. Individuals are denoted by lower case identifiers (e.g., *sam*). Identifiers representing individuals are called **constants**.

2. The geometric mean of a list of numbers $x_1, \ldots, x_n$ is $(\prod_i x_i)^{1/n}$. The logarithm of the geometric mean is therefore $1/n \sum_i \ln x_i$. Thus geometric mean = exp(average (logs)).

Table 2: Summary of Notation for Relational Concepts

Notation	Explanation	Example
$\mathbb{A}, \mathbb{B}$	**Population Variables**	$\mathbb{A}$
T, U, V	**Terms**. A term consists of a **functor** and its arguments. The arguments may be any combination of **population variables** and **constants**.	$gender(\mathbb{A})$
$\vec{T}, \vec{U}, \vec{V}$	**Term Tuples**. A list of terms. A given term can occur at most once. Order is significant.	$gender(\mathbb{B}); Friend(\mathbb{A}, \mathbb{B})$
$\mathrm{Ra}(T), \mathrm{Ra}(\vec{T})$	The **range of a term** and the **range of a term tuple**. The range of a **term** is the range of its functor. The range of a tuple is the Cartesian product of the ranges of the constituent **terms**.	$\mathrm{Ra}(gender(\mathbb{B}), Friend(\mathbb{A}, \mathbb{B})) = \{\mathrm{W}, \mathrm{M}\} \times \{\mathrm{T}, \mathrm{F}\}$
t, u, v	**Values** from the ranges of T, U, V.	$\mathrm{M}; \mathrm{T}$
$\vec{t}, \vec{u}, \vec{v}$	**Tuples of values** from the ranges of $\vec{T}, \vec{U}, \vec{V}$.	(M, T)
$T = t$	**Literal**. A term bound to a value from its range.	$Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}$
$\vec{T} = \vec{t}$	**Literal conjunction**. A conjunction bound to a tuple of values from its **range**. Every **term** in a literal conjunction is bound to its corresponding value.	$Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}, gender(\mathbb{A}) = \mathrm{W}$
γ	**Instantiation**. An instantiation maps zero or more **population variables** to appropriate **constants**.	$\gamma = \{\mathbb{A} \backslash anna, \mathbb{B} \backslash bob\}$
$T^*, \vec{T}^*$	**Fully ground term/tuple**. All arguments to all functors are constants. Equivalently, the term/tuple does not contain any population variables.	$gender(anna), Friend(anna, bob)$
Λ^*	Complete **literal conjunction** comprising **fully grounded** terms. The conjunction binds *every* ground term to a value.	See Table 3.
$\mathrm{n}\left[\vec{T} = \vec{t}; \Lambda^*\right]$	**Instantiation Count.** Counts the number of groundings of $\vec{T} = \vec{t}$ that evaluate as true in Λ^*. A **grounding** instantiates *all* population variables in $\vec{T}$.	See Table 3.

5.1.1 Functor Notation

A **functor** $f : \mathcal{P}_1, \ldots, \mathcal{P}_a \rightarrow V_f$ maps a list of individuals to a functor value, where V_f is the output type or **range** of the functor. In this paper we consider only functors with a finite range. If $V_f = \{\mathrm{T}, \mathrm{F}\}$, the functor f is a (Boolean) **predicate**; other functors are called **attributes**. A predicate with more than one argument is called a **relationship**. Predicates typically represent the presence of a binary property or a relationship, while attributes typically represent properties that can take multiple values. We use lowercase for attribute functors and uppercase for predicates. We define two kinds of variables, with distinct domains: (i) A **population variable** varies over a population domain. The same population may be associated with more than one population variable. We use an outline Latin font for population variable ($\mathbb{A}, \mathbb{B}, \ldots$). (ii) A **term** is an expression of the form $f(\boldsymbol{\tau}) \equiv f(\tau_1, \ldots, \tau_k)$, where each τ_i is a population variable or a constant of the appropriate argument type for that functor.[3] A term can be assigned values in the range of its functor.

In a context where the functor and arguments of terms are not important, we denote them by uppercase Latin letters ($T, U, \ldots$).

A **literal** is an assignment to a term, denoted generically as $T = t$, where t is in the range of the functor for T. Literals can be combined to generate **formulas**. In this paper we consider only formulas that are **conjunctions** of literals, denoted by the Prolog-style comma notation, e.g., $T = t, \ldots, U = u$, for which we also use the vector notation $\vec{T} = \vec{t}$.

5.1.2 Groundings and Relational Structures

An term is **ground** if it contains no population variables. We mark fully ground terms and formulas by an asterisk, T^*. An **instantiation** for a term assigns a constant to a set of the population variables in the term. Formally, an instantiation γ is a set $\{\mathbb{A}_i \backslash a_1, \ldots, \mathbb{A}_j \backslash a_j\}$ that assigns a constant $\gamma(\mathbb{A}_i)$ to each variable $\mathbb{A}_i$ from its population. The expression T_γ denotes the term that results from applying the substituition γ to all variables in T. If the instantiation γ specifies a constant for all the population variables that occur in the term, the resulting term is ground, and γ is called a **grounding**. An instantiation is applied to a formula by applying it to all terms in the formula. Thus a grounding of a formula is an instantiation that grounds of all its terms.

A **relational structure** is a (model, interpretation) pair that assigns a unique value to each ground term (?). Assuming a finite list of functors, and that all populations are finite, a relational structure is equivalent to a **complete conjunction** of ground literals $\Lambda^* \equiv (\vec{T}^* = \vec{t})$. The ground literals in the relational structure are called **facts**. A conjunction of ground literals evaluates as true in a relational structure if each of its conjuncts is a fact. A well-studied operation (?) for statistical learning is to count, for a given nonground formula $\vec{T} = \vec{t}$ and complete conjunction Λ^*, the number of groundings of the formula that evaluate as true in the complete conjunction. Note that only groundings that instantiate exactly the population variables that appear in $\vec{T}$. We refer to this quantity as the formula's

3. The traditional term in first-order logic for a term variable is "function term". In statistical-relational learning, alternative terms include "parametrized random variable" (?), "atom" (?), "Bayesian atom" (?), and "functor random variable" (?).

instantiation count, denoted by

$$\mathrm{n}\left[\vec{T} = \vec{t}; \Lambda^*\right].$$

5.1.3 Examples.

Figure **??**(a) shows the facts in a relational structure, represented as database tables. For this example only, let us assume that the tables represent a relational structure completely, and hence specifies a complete conjunction Λ^*. Table 3 shows examples of instantiation counts for this complete conjunction.

Table 3: Instantiation counts in the database of Figure **??**. For the sake of the example, we treat the database as specifying a complete conjunction $\Lambda^* = \{gender(anna) = W,\ gender(bob) = M,\ Friend(anna, bob) = T,\ Friend(bob, anna) = T,\ Friend(anna, anna) = F,\ Friend(bob, bob) = F\}$.

Formula	Instantiation Count
$gender(\mathbb{A}) = \mathrm{M}$	1
$Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}$	2
$Friend(bob, \mathbb{B}) = \mathrm{T}$	1
$gender(\mathbb{B}) = \mathrm{M}, Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}$	1
$gender(anna) = \mathrm{M}, Friend(\mathbb{A}, anna) = \mathrm{T}$	0

5.2 Bayes Nets for Relational Data.

A Bayes net (BN) is a pair $\langle G, \boldsymbol{\theta}_G \rangle$, where G is a directed acyclic graph and $\boldsymbol{\theta}_G$ is a set of parameters that specify the probability distributions of children conditional on instantiations of their parents. A **Template Bayes Net** (TBN) is a Bayes net whose nodes are nonground terms (?, ?). That is, every node in the Bayes net is a term containing one or more population variables. When describing Bayes nets, we use "term" and "node" interchangeably. Table 4 summarizes our notation for Bayes nets and Figure **??**(b) shows an example net.

A **family** comprises a node and its parents. A **family configuration** specifies a value for a child node and each of its parents. Since we consider functors with discrete ranges only, there are only finitely many family configurations. Using the notation in Table 4, a family configuration is equivalent to the conjunction $T = t, \mathrm{Pa}(T) = \vec{t}_{pa}$. For each family configuration, a **Bayes net parameter**

$$\theta(T = t | \mathrm{Pa}(T) = \vec{t}_{pa})$$

specifies the probability of the child value t given the parent values $\vec{t}_{pa}$.[4] Given a complete conjunction Λ^*, the number of family configurations is[5]

$$\mathrm{n}\left[T = t, \mathrm{Pa}(T) = \vec{t}_{pa}; \Lambda^*\right].$$

4. For i.i.d. data, a commonly used notation for a BN parameter is θ_{ijk} (?).
5. For i.i.d. data, a commonly used notation for this quantity is n_{ijk} (?).

Table 4: Summary of Notation for Template Bayes Nets

Notation	Explanation	Example
$\mathrm{Pa}(T) : T \to \vec{T}$	**Parents**. The **terms** associated with the parent nodes of the unique Bayes net node associated with T. The parent terms will be distinct.	$\mathrm{Pa}(\mathit{gender}(\mathbb{A})) = \langle \mathit{gender}(\mathbb{B}), \mathit{Friend}(\mathbb{A},\mathbb{B})\rangle$
$\mathrm{Ch}(T) : T \to \vec{T}$	**Children**. The **terms** associated with the child nodes of the unique Bayes net node associated with T.	$\mathrm{Ch}(\mathit{gender}(\mathbb{A})) = \mathit{CoffeeDr}(\mathbb{A})$
$\vec{t}_{pa}$	**Value in the parent range**. Tuple of values from $\mathrm{Ra}(\mathrm{Pa}(T))$.	$\langle \mathrm{M}, \mathrm{T}\rangle$
$T = t, \mathrm{Pa}(T) = \vec{t}_{pa}$	**Family Configuration** that specifies values for node T and its parent nodes $\mathrm{Pa}(T)$.	$\mathit{gender}(\mathbb{A}) = \mathrm{M}$, $\mathit{gender}(\mathbb{B}) = \mathrm{W}$, $\mathit{Friend}(\mathbb{A},\mathbb{B}) = \mathrm{T}$
$\theta(T = t \mid \mathrm{Pa}(T) = \vec{t}_{pa})$	**Conditional probability** of a node value given a parent configuration.	$\theta(\mathit{gender}(\mathbb{A}) = \mathrm{W} \mid \mathit{gender}(\mathbb{B}) = \mathrm{W}, \mathit{Friend}(\mathbb{A},\mathbb{B}) = \mathrm{T}) = 0.55$
$\theta(T = t)$	**Marginal probability** of a node value entailed by the Bayes net.	$P(\mathit{gender}(\mathbb{A}) = \mathrm{M}) = 0.55$

5.3 Bayes Net Gibbs Probabilities.

We review the equation for Gibbs conditional probabilities for a Bayes net that specifies a joint distribution over its nodes via the standard product formula (?). Inferring a Gibbs conditional probability can be represented as a probabilistic query

$$P(T = t | \vec{V} = \vec{v}) = ?$$

where T is the target node, t is a value for the target node, and $\vec{v} = \vec{V}$ specifies a value for every other node. The product formula for the Bayes net joint distribution entails that the Gibbs probability is proportional to a product of conditional probabilities for the target node and its children (?, Ch.14.5.2):

$$P(T = t | \vec{V} = \vec{v}) \propto \prod_{U \in \{T\} \cup \mathrm{Ch}(T)} \theta(U = u | \mathrm{Pa}(U) = \vec{u}_{pa}) \tag{1}$$

where u and $\vec{u}_{pa}$ are specified by the values in $t, \vec{v}$ for the corresponding nodes. The product is a number between 0 and 1 that needs to be normalized to obtain the Gibbs conditional probability.

Example. For the Bayes net of Figure **??**, consider the query

$$P(\mathit{gender}(\mathbb{A}) = \mathrm{W} | \mathit{gender}(\mathbb{B}) = \mathrm{W}, \mathit{Friend}(\mathbb{A},\mathbb{B}) = \mathrm{T}, \mathit{CoffeeDr}(\mathbb{A}) = \mathrm{T}).$$

By Equation 1, this probability is proportional to

$$\begin{aligned} \theta(\mathit{gender}(\mathbb{A}) = \mathrm{W} | \mathit{gender}(\mathbb{B}) = \mathrm{W}, \mathit{Friend}(\mathbb{A},\mathbb{B}) = \mathrm{T}) \quad &\cdot \quad \theta(\mathit{CoffeeDr}(\mathbb{A}) = \mathrm{T} | \mathit{gender}(\mathbb{A}) = \mathrm{W}) \\ = 0.55 \quad &\cdot \quad 0.8 = 0.44. \end{aligned}$$

5.4 Log-Linear Models

Equation 1 can be seen as an instance of a standard log-linear schema as follows. The general equation form that defines a **discriminative log-linear model** (?, Sec.4.2.2.1) is

$$P(T = t|\vec{V} = \vec{v}) \propto exp(w_t + \sum_{i=1}^{K} w_i f_i(t, \vec{V})), \tag{2}$$

where T is the target or output variable, and $\vec{v} = \vec{V}$ represents an assignment of values to the input variables. The model is based on a finite set of K features, and for each feature there is a real-valued weight parameter w_i. The term w_t is a bias weight that may depend on the target node value t but not the input variables. The functions $f_1, \ldots, f_k$ are **feature functions**, such that f_i returns a real number for feature i given values for both the target and input variables. A log-linear equation defines a log-linear model; we use the term "equation" to emphasize the mathematical form, and the term "model" to emphasize the parameter space.

Rewriting Equation 1 as

$$P(T = t|\vec{V} = \vec{v}) \propto exp(\sum_{U \in \{T\} \cup \mathrm{Ch}(T)} \ln \theta(U = u|\mathrm{Pa}(U) = \vec{u}_{pa})) \tag{3}$$

shows that a BN Gibbs probability follows a log-linear model with the following specifications. (1) The features are all family configurations whose child node is either the target node or a child of the target node. (2) The feature weights are the log-conditional probabilities associated with a family configuration; the bias weight is 0. (3) The feature function for each a family configuration returns 1 if the family configuration is specified by the conjunction $(t, \vec{v})$ of input and output variables, 0 otherwise.

This log-linear equation specifies the Gibbs probability for the nonground terms/nodes in the template Bayes net. Our goal in this paper is to define inference for queries whose target are ground terms. We therefore want to generalize Equation (3) for ground terms. The log-linear equations for ground terms that we consider are also based on products of the Bayes net parameters, combined with transformations.

6. The Log-Difference Frequency Equation

We propose a log-linear equation for computing a Gibbs conditional probability in closed form, given (i) a ground target node (the output variable), (ii) a target value for the target node, (iii) a complete set of values for all ground terms other than the term of the target node (the input variables), and (iv) a template Bayes net. We refer to our proposal as the **log-difference frequency equation**. Our notation is summarized in Table 5. For now we assume that component (iv), the template BN, is fixed (Section 6.3 discusses structure learning), and consider in this section components (i)–(iii). Figure 5 shows the program flow for computing a Gibbs probability using the log-difference frequency equation.

6.1 Conditional Queries

Conditional queries comprise the following elements:

Table 5: Summary of Notation for Relational Gibbs Probabilities

Notation	Explanation	Example
T^*_γ	Fully Ground Target Node	$[gender(\mathbb{A})]_{\{\mathbb{A}\setminus sam\}} = gender(sam)$
t	Target Node Value	M
Δ^*	**Query Conjunction.** A fully ground literal conjunction that binds every term to a value except for the target node	See Table 3
$P(\mathsf{T}^*_\gamma = t \mid \Delta^*)$	**Gibbs conditional probability**, or a **Query**	$P(gender(sam) = \mathrm{M} \mid \Delta^*)$
$\mathsf{U}_\gamma = u, \mathrm{Pa}(\mathsf{U})_\gamma = \vec{u}_{pa}$	Partially ground query family configuration	$gender(sam) = \mathrm{M}$, $gender(\mathbb{B}) = \mathrm{W}$, $Friend(sam, \mathbb{B}) = \mathrm{T}$
$\mathrm{n}^{\mathrm{r}}[\mathsf{U}_\gamma = u, \mathrm{Pa}(\mathsf{U})_\gamma = \vec{u}_{pa}; \Delta^*, \mathsf{T}_\gamma = t]$	**Relevant Instantiation Count** of a query family configuration.	See Section 6.3.
$\mathrm{p}^{\mathrm{r}}[\mathsf{U}_\gamma = u, \mathrm{Pa}(\mathsf{U})_\gamma = \vec{u}_{pa}; \Delta^*, \mathsf{T}_\gamma = t]$	**Relevant Instantiation Frequency** of a query family configuration.	See Section 6.3.

1. A **target literal** $\mathsf{T}^*_\gamma = t$, where T^*_γ denotes a ground term that results from applying the grounding γ to the term T.

2. A conjunction Λ^* that specifies a value for each ground term other than T^*_γ. The conjunction $(\mathsf{T}^*_\gamma = t, \Delta^*)$ specifies the value of *every* ground term; we refer to it as the **query conjunction**.

A Gibbs conditional probability corresponds to a probabilistic **query**

$$P(\mathsf{T}^*_\gamma = t|\Lambda^*) =?$$

For example, a target node may be $gender(sam)$, which results from the grounding $gender(\mathbb{A})\{\mathbb{A}\backslash sam\}$.[6] The conjunction Λ^* specifies a value for all ground terms other than $gender(sam)$.

6.2 Features and Feature Weights

The features are all family configurations whose child node is either the target node or a child of the target node, as with nontemplate BNs. Thus the set of features equals the set of **query family configurations**

$$QFC \equiv \{\mathsf{U} = u, \mathrm{Pa}(\mathsf{U}) = \vec{u}_{pa} : U \in \{T\} \cup \mathrm{Ch}(T), u \in \mathrm{Ra}(U), \vec{u}_{pa} \in \mathrm{Ra}(\mathrm{Pa}(U))\}.$$

For each query family configuration there is an associated weight

$$w \equiv \ln \theta(U = u|\mathrm{Pa}(U) = \vec{u}_{pa}).$$

If $\theta(U = u|\mathrm{Pa}(U) = \vec{u}_{pa}) = \ln \theta(U = u)$, then the parent configuration is probabilistically independent of the child condition (according to the template BN). In that case we say that the feature defined by the family configuration is **irrelevant**, otherwise **relevant**. In the log-difference equation, irrelevant features receive weight 0, which is equivalent to eliminating them from the model. Table 6 illustrates relevant and irrelevant features.
Discussion. It is well-known that eliminating irrelevant features is important for predictive accuracy in statistical-relational learning (?, ?, ?, ?). For example, individuals whose every relationship with the target individual has value F are often irrelevant to predicting features of the target. In the example above, the gender of nonfriends (all $\mathbb{B}$ such that $Friend(sam, \mathbb{B}) = \mathrm{F}$) is probabilistically independent of the gender of the target. In a realistic social network, where 99% or more of the users are *not* friends with a given individual, this would entail that the vast majority of groundings are irrelevant to predicting an individual's gender. A common approach to eliminating irrelevant predictors is to stipulate a logical condition that must be met for the predictor to be included (?, ?, ?, ?). The log-difference model instead defines irrelevant features in terms of the Bayes net parameters, and eliminates them by assigning 0 weight.

6. The node $gender(sam)$ can also be defined by the grounding $gender(\mathbb{B})\{\mathbb{B}\backslash sam\}$. To make our definition unambiguous, we assume that the BN is in main functor format and that the template node T is the main functor node for its functor; see (?).

Table 6: Relevant and Irrelevant Features, or Family Configurations, for the Bayes net of Figure **??**. Marginal probabilities are computed using standard Bayes net inference.

Child Node Value	Marginal Probability	Parent configuration	Conditional Probability	Relevant?
$CoffeeDr(\mathbb{A}) = \mathrm{T}$	0.70	$gender(\mathbb{A}) = \mathrm{W}$	0.80	yes
$gender(\mathbb{A}) = \mathrm{W}$	0.45	$gender(\mathbb{B}) = \mathrm{W}$, $Friend(\mathbb{A}, \mathbb{B}) = \mathrm{F}$	0.45	no

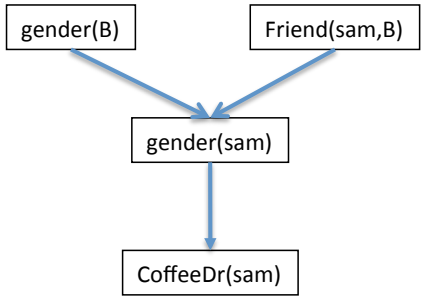

Figure 5: Graphical representation of partial grounding for the target node $gender(sam)$. The graph results from instantiating the population variable $\mathbb{A}$ with the constant sam in the template graph of Figure **??**. Feature counts are computed with respect to the instantiated nodes. Notice that, for some nodes, this instantiation leads to only a *partial* grounding.

6.3 Feature Functions

For each feature, a feature function maps the query conjunction to a real number. A common feature function choice in log-linear models is the number of times that the feature is instantiated in the query conjunction. Our basic proposal is to use, instead, the *frequency* with which each feature is instantiated in the query conjunction. To compute feature frequencies, we first compute feature counts, then normalize. To count all and only instantiations that are related to the grounding of a target node, we apply the grounding to its parents, children, and co-parents, as illustrated in Figure 6.

Normalizing feature counts to obtain feature frequencies must be done with care to take account of irrelevant features. Since irrelevant features receive log-difference weight 0, they are effectively pruned from the model, so their portion of the instantiation counts should be shifted to the relevant features, analogous to conditioning on relevant features. We therefore define the following two feature functions, where a feature is a family configuration.

Relevant Count The relevant count n^{r} is 0 if the family configuration is irrelevant; otherwise it is the instantiation count.

Relevant Frequency The relevant frequency of a feature is its relevant count, divided by the sum of all relevant counts for the same family.

Child Value	Prior Prob.	Parent State	Cond. Prob.	w	p^{r}	$w \times \mathrm{p}^{\mathrm{r}}$	n^{r}	$w \times \mathrm{n}^{\mathrm{r}}$
$cd(sam) = \mathrm{T}$	0.70	$g(sam) = \mathrm{W}$	0.80	0.13	1.0	0.13	1	0.13
$cd(sam) = \mathrm{F}$	0.30	$g(sam) = \mathrm{W}$	0.20	−0.40	0.0	0.00	0	0.00
$g(sam) = \mathrm{W}$	0.45	$g(B) = \mathrm{W}$, $F(sam, B) = \mathrm{T}$	0.55	0.20	0.4	0.08	40	8.02
$g(sam) = \mathrm{W}$	0.45	$g(B) = \mathrm{M}$, $F(sam, B) = \mathrm{T}$	0.37	−0.19	0.6	−0.11	60	−11.74
$g(sam) = \mathrm{W}$	0.45	n/a	n/a	−0.79	1.0	−0.79	1	−0.79
Sum ($\ln P(gender(sam) = \mathrm{W}\|\Delta^*)$)						−0.70		−4.38
$cd(sam) = \mathrm{T}$	0.70	$g(sam) = \mathrm{M}$	0.60	−0.15	1.0	−0.15	1	−0.15
$cd(sam) = \mathrm{F}$	0.30	$g(sam) = \mathrm{M}$	0.40	0.28	0.0	0.00	0	0.00
$g(sam) = \mathrm{M}$	0.55	$g(B) = \mathrm{W}$, $F(sam, B) = \mathrm{T}$	0.45	−0.20	0.4	−0.08	40	−8.02
$g(sam) = \mathrm{M}$	0.55	$g(B) = \mathrm{M}$, $F(sam, B) = \mathrm{T}$	0.63	0.13	0.6	0.08	60	8.14
$g(sam) = \mathrm{M}$	0.55	n/a	n/a	−0.59	1.0	−0.59	1	−0.59
Sum ($\ln P(gender(sam) = \mathrm{M}\|\Delta^*)$)						−0.75		−0.63

Table 7: Applying the log-difference frequency equation with the BN of Figure **??** to compute $P(gender(sam) = \mathrm{W}|\Delta^*)$ and $P(gender(sam) = \mathrm{M}|\Delta^*)$. Each row represents a feature/family configuration. For the sake of the example we suppose that the conjunction Δ^* specifies that Sam is a coffee drinker, has 60 male friends, and 40 female friends. The last two columns show the result of replacing frequencies by counts (the log-difference count equation in Section 7).

Algorithm 1 describes the computation of relevant counts and relevant frequencies. Table 7 gives examples. This completes our definition of the set of features, weights, and feature functions. All told, the resulting log-linear equation is as follows.

[The Log-Difference Frequency Equation]

$$P(\mathsf{T}^*_\gamma = t|\Delta^*) \quad \propto$$
$$\sum_U \sum_{u,\vec{u}_{pa}} \quad [\ln \theta(U = u|\mathrm{Pa}(U) = \vec{u}_{pa})] \quad \cdot \quad \mathrm{p}^{\mathrm{r}}\left[\mathsf{U}_\gamma = u, \mathrm{Pa}(\mathsf{U})_\gamma = \vec{u}_{pa}; \mathsf{T}^*_\gamma = t, \Delta^*\right])$$

where

$$\begin{array}{rlll} U & \text{varies over} & T \cup \mathrm{Ch}(T), & \\ u & \text{varies over} & \mathrm{Ra}(U), & \text{and} \\ \vec{u}_{pa} & \text{varies over} & \mathrm{Ra}(\mathrm{Pa}(U)). & \end{array}$$

6.4 Estimating Bayes net parameeers

The Bayes net parameters can be estimated using the empirical conditional frequencies observed in an input dataset $\mathcal{D}^*$: The parameter estimate for a family configuration is the number of instantiations of that family configuration in $\mathcal{D}^*$, divided by the sum of all instantiation counts for that family that agree on the parent values and vary the child values. In our notation, the estimate is defined by

$$\widehat{\theta}(T = t|\mathrm{Pa}(T) = \vec{t}_{pa}; \mathcal{D}^*) = \frac{\mathrm{n}\left[T = t, \mathrm{Pa}(T) = \vec{t}_{pa}; \mathcal{D}^*\right]}{\sum_{t' \in \mathrm{Ra}(T)} \mathrm{n}\left[T = t', \mathrm{Pa}(T) = \vec{t}_{pa}; \mathcal{D}^*\right]}. \quad (4)$$

A theoretical justification for using the observed conditional frequencies is that these estimates maximize a pseudo-likelihood function that measures how well a template BN matches an input dataset (?, ?). The pseudo-likelihood can be interpreted as the expected value of the log-likelihood of a random grounding of the BN nodes in the template model.

7. Empirical Comparison of Count vs. Proportion Feature Functions

Our first set of experiments compared the predictive accuracy of different Bayes net regression equations. The next section describes experiments comparing the Bayes net methods with general weight learning for log-linear models.

7.1 Experimental Conditions and Metrics

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

7.1.1 Datasets

We describe the datasets in terms of their representation as databases with tables. The databases follow an Entity-Relationship (E-R) design (?). An E-R schema can be translated into our function-based logical notation as follows: 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. We used 5 benchmark real-world databases from prior work (?).

MovieLens Database This is a standard dataset from the UC Irvine machine learning repository. It contains two tables representing entity sets: User with 941 tuples and Item (Movies) with 1,682 tuples. The User table has 2 descriptive attributes, *age* and *gender*. We discretized the attribute *age* into three equal-frequency bins. The table Item represents information about the movies. It has 17 Boolean attributes that indicate the genres of a given movie. There is one relationship table Rated corresponding to a Boolean predicate. The Rated contains Rating as descriptive attribute; 80,000 ratings are recorded. We performed a preliminary data analysis and omitted genres that have only weak correlations with the rating or user attributes, leaving a total of three genres (Drama, Horror, Action).

Mutagenesis Database This dataset is widely used in Inductive Logic Programming research (?). We used a previous discretization (?). Mutagenesis has two entity tables, Atom with 3 descriptive attributes, and Mole (decribing molecules), with 5 descriptive attributes. There are two relationship tables, MoleAtom, indicating which atoms are parts of which molecules, and Bond, which relates two atoms and has 1 descriptive attribute.

Hepatitis Database This data is a modified version of the PKDD02 Discovery Challenge database (?). The database contains information on laboratory examinations of 771 hepatitis B- and C-infected patients, taken between 1982 and 2001. The data are organized in 7 tables (4 entity tables, 3 relationship tables) with 16 descriptive attributes. They contain basic information about the patients, results of biopsy, information on interferon therapy, results of out-hospital examinations, and results of in-hospital examinations.

Mondial Database This dataset contains data from multiple geographical web data sources. We follow the modification of She *et al.* (?), and use a subset of the tables and discretized features: 2 entity tables, $Country, Economy$. The descriptive attributes of Country are continent, government, percentage, majority religion, population size. The descriptive attributes of Economy are inflation, gdp, service, agriculture, industry. A relationship table Economy_Country specifies which country has what type of economy. A self-relationship table Borders relates two countries.

UW-CSE database 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 the relationships (i.e. $AdvisedBy$, $TaughtBy$).

7.1.2 Prediction Metrics

We evaluate the algorithms using classification accuracy and conditional log likelihood (CLL). These metrics have been used in previous evaluations of MLN learning (?, ?). For each fact $\mathsf{T}^* = t$ in the test dataset, we evaluate the accuracy of the predicted Gibbs probability $P(\mathsf{T}^* = t|\Delta^*_{-\mathsf{T}})$, where $\Delta^*_{-\mathsf{T}}$ is a complete conjunction for all ground terms other than T^*. Thus $\Delta^*_{-\mathsf{T}}$ represents the values of the input variables as specified by the test dataset. For classification accuracy, a model's prediction is scored as correct if the true value of the ground term in the test dataset receives the highest Gibbs probability. CLL is the average of the logarithm of the Gibbs probability for each fact in the test dataset. Thus $\exp(CLL)$ is the geometric mean of the Gibbs probabilities.[7] Both metrics are reported as averages over all functors that represent descriptive attributes. We do not use Area Under Curve, as it mainly applies to binary values, and most of the attributes in our datasets are nonbinary. 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 (?, ?)). The models were trained on 4 of the 5 folds, then tested on

7. The geometric mean of a list of numbers $x_1, \ldots, x_n$ is $(\prod_i x_i)^{1/n}$.

the remaining one. All results are averages from 5-fold cross validation, over all descriptive attributes in the database.

7.2 Learning the Bayes Net Structure and Parameters

All the methods compared in this experiment require a prior Bayes net structure and parameters. To obtain the structure, the learn-and-join algorithm (?) was applied to each benchmark database. The parameters were computed from the empirical conditional frequencies in the database (Eq. 4) using previously-published algorithms (?). The resulting structure and parameters were used for all methods in this experiment. Table 10 shows the 3 log-linear equations compared with abbreviations. To determine the set of relevant features, we eliminated conjunctions that involved negated relationships (e.g., $\mathit{Friend}(\mathbb{A},\mathbb{B}) = \mathrm{F}$), for the following reasons. (i) We inspected the log-difference weights for these features and found them close to 0. Eliminating such weights approximates regularization. We leave a full regularization approach for eliminating irrelevant features for future work. (ii) Eliminating information from unrelated entities is standard practice in statistical-relational learning (?, ?). (iii) Log-linear weight learning methods (e.g., Alchemy, see Section 9) do not scale to our datasets when features with negated relationships are included (because of the difficulty of computing sufficient statistics for such features).

Table 8: The Bayes net log-linear equations compared in our experiments; cf. Table 8. Theorem 7.1 shows that the log(cp) + frequency and log-diff + frequency are equivalent.

		Feature Function	
		Count	Frequency
Weights	Log-CPs	log(cp) + count	log-diff + freq
	Log-diff.	log-diff + count	

7.3 Results

Table 11 summarizes the results for the Bayes net regression equations. The numbers represent an average over many individual scores, one for each fact in the database. For instance in the biggest dataset, MovieLens, the average is over a total of 170,000 scores; see Table 14.

To aid interpretability, we also report the following transformation of CLL: prob. ratio $= \exp(CLL(\mathit{method}) - CLL(\mathit{log}(\mathit{cp}) + \mathit{count}))$, where *method* is one of log-diff + count and log-diff + frequency. This quantity represents the geometric mean, over all test facts, of the fact likelihood ratio of *method* over the $\mathit{log}(\mathit{cp}) + \mathit{count}$ equation. For instance, the value of 1.05 for the dataset UW and the log-diff + count method means that on (geometric) average, the likelihood that the log-diff + count method assigns to the correct value for the target node is 1.05 times that assigned by the log-(cp) + count method.

Table 9: Conditional log-likelihood (log-probabilities) and classification accuracy (in percent) of Bayes net regression predictions. We show averages and standard deviations. The probability ratios take as the baseline the log(cp) + count method. They can be interpreted as the geometric average ratio of likelihoods assigned by the model to the true target node value.

Accuracy	UW	Mondial	MovieLens	Mutagenesis	Hepatitis
log(cp) + count	78 ± 0.08	40 ± 0.05	64 ± 0.01	62 ± 0.05	49 ± 0.03
log-diff + count	**81** ± 0.06	**45** ± 0.04	62 ± 0.02	**67** ± 0.03	**55** ± 0.02
log-diff + freq	**81** ± 0.06	**45** ± 0.04	**65** ± 0.01	**67** ± 0.03	**55** ± 0.02

CLL	UW	Mondial	MovieLens	Mutagenesis	Hepatitis
log(cp) + count	-0.47 ± 0.10	-1.47 ± 0.17	-1.19 ± 0.07	-0.84 ± 0.03	-1.33 ± 0.07
log-diff + count	-0.42 ± 0.05	-1.36 ± 0.11	-1.10 ± 0.16	-0.77 ± 0.03	-1.20 ± 0.07
Prob. ratio	1.05	1.12	1.09	1.07	1.14
log-diff + freq	**-0.41** ± 0.04	**-1.34** ± 0.09	**-0.71** ± 0.01	**-0.73** ± 0.04	**-1.07** ± 0.10
Prob. ratio	1.06	1.14	1.62	1.12	1.30

7.4 Discussion

Frequency feature functions achieve top performance for classification accuracy. However, classification scores are similar across the three methods, and the difference between frequencies and counts + log-difference weights is small ($< 1\%$), except for a bigger improvement (3%) on MovieLens. Whereas classification accuracy is a 0-1 loss function, CLL is continuous, so the frequency approach's balancing of factors has substantially more impact on CLL. With respect to CLL, we observe the following ranking of methods on each dataset:

Frequency > log-difference count > count.

Therefore: *Using Bayes net parameters, frequency feature functions outperform count feature functions, with both the log-cp and the log-diff weight computation methods.* This finding supports our hypothesis that using frequencies as feature functions is an effective way of addressing the imbalance problem.

Applied to feature counts, log-difference weights improve on log-conditional probabilities. Our explanation for this finding is that log-difference weights are a heuristic for solving the imbalance problem, for the following reasons. Generally speaking, the number of instantiation count increases for a feature when the feature is based on longer relationship chains. In other words, (i) the number of instantiation counts increases with distance to the target entity. For instance, in the Bayes net of Figure **??**, with target node $gender(sam)$, the feature $CoffeeDrinker(sam)$ is at distance 0 from Sam, and has instantiation count at most 1. The feature $Friend(sam, \mathbb{B}) = \mathrm{T}, gender(\mathbb{B}) = \mathrm{W}$ is at distance 1 from Sam (one link away), and its maximum instantiation count is the number of Sam's friends. A feature like $Friend(sam, \mathbb{B}) = \mathrm{T}, Friend(\mathbb{B}, \mathbb{C}) = \mathrm{T}, gender(\mathbb{C}) = \mathrm{W}$—referring to women friends of Sam's friends—is at distance 2 from Sam (two links away), and its maximum instantia-

tion count is the number of friends of friends of Sam. We also expect that the correlation between Sam's gender and that of a friend will generally be stronger than than between Sam's gender and the gender of a friend of a friend. Therefore we can expect that (ii) the probabilistic association of the target node with related entities decreases with distance to the target entity. (iii) As the probabilistic association decreases, so does the log-difference weight since it measures the strength of the probabilistic association. Combining the observations (i), (ii), (iii) entails that features with high instantiation counts tend to have low log-difference weights. Therefore log-difference weights address the imbalance problem by tending to assign lower weight magnitudes to features with high instantiation counts. Table 12 summarizes the quantities related to this analysis and how they relate to each other.

Table 10: Connections between different quantities for features that are consistent with the observed performance of count feature functions.

Distance to Target Entity	Feature Instantiation Count	Correlation with Target Attribute	Log-Difference Weight
+ increases	+ increases	- decreases	- decreases

8. Comparison with RDN-Boost

The experiments in Section 8 held the Bayes net structure and parameters constant and compared transformations of the BN parameters into log-linear weights. In this section we examine a setting where the same features are computed from the BN structure, but weights are *not* computed from the BN parameters. Instead weight values are optimized by a local search method. This experiment used the same conditions and metrics described in Section 8.1. In addition to comparing predictive performance, we also report learning times.

To learn the weights, we applied the default training procedure of the Alchemy package (?). This procedure takes as input a set of features specified as logical formulas, and returns a weight for each formula. We followed the method recommended by the Alchemy group (?) for converting a Bayes net structure to a Markov Logic Network structure: For each family configuration F_{ijk} in the BN, add a conjunction of literals that specifies the state. We also added unit clauses for each node-value combination, as recommended by the Alchemy group; unit clause weights can represent the bias weight of a log-linear equation.

We refer to moralization+weight learning as the **MBN** method, for "Moralized Bayes Net" (?). MBN has been the state-of-the-art method for log-linear prediction with Bayes nets (?). Figure 7 shows the program flow for the MBN method. Markov Logic Network weight learning optimizes for log-linear inference with counts as feature functions (?). The prediction probabilities were computed exactly using the log-linear equation 2 with counts as feature functions. We used an exact computation rather than approximate inference (e.g., MC-SAT), to avoid confounding the effect of the log-linear equation with that of inference implementation. Experiments with MC-SAT produced similar results. We also computed

the results with frequencies as feature functions, which for optimized weights were very similar, so we do not present them.

8.1 Results

The Bayes net frequency regression predictions are competitive with those from a model with optimized general weights.
Accuracy. Table 13 shows the scores of the MBN method, together with the log-difference frequency results from Table 11. The log-difference frequency model scores slightly higher than the MBN weights on every dataset, with the biggest differences on Mutagenesis (5%), MovieLens (5%) and Hepatitis (4%).

Table 11: Conditional log-likelihood (log-probabilities) and classification accuracy (percent) of MBN and log-difference frequency predictions.

Accuracy	UW	Mondial	MovieLens	Mutagenesis	Hepatitis
MBN	80 ± 0.05	44 ± 0.04	60 ± 0.02	62 ± 0.02	51 ± 0.02
log-diff + freq	**81** ± 0.06	**45** ± 0.04	**65** ± 0.03	**67** ± 0.03	**55** ± 0.02

CLL	UW	Mondial	MovieLens	Mutagenesis	Hepatitis
MBN	-0.44 ± 0.07	**-1.28** ± 0.07	-0.79 ± 0.03	-0.91 ± 0.09	-1.18 ± 0.26
log-diff + freq	**-0.41** ± 0.04	-1.34 ± 0.09	**-0.71** ± 0.01	**-0.73** ± 0.04	**-1.07** ± 0.10
Prob. ratio	1.03	0.94	1.08	1.20	1.12

CLL. The log-difference frequency model scores better than MBN model on UW, MovieLens, Mutagenesis and Hepatitis (probabilities 3–20% higher) and scores slightly worse on Mondial (6% lower probability); see Figure 8.
Learning Times. Table 14 shows run time results for structure and parameter learning. We see *clear scalability advantages for the maximum likelihood conditional probability estimates used in the Bayes approach*: they take seconds to compute, whereas Alchemy weight optimization requires as much as 10 hours in the worst case (Hepatitis).
Summary. The findings from this section and the previous one support our claim that balancing the scales of feature functions is important for using Bayes net parameters in a log-linear model. The combination of transformed BN parameters +feature frequencies is as predictively accurate as using feature counts with optimized general weights. While the Bayes net log-linear model is comparable in accuracy, its parameters can be learned much faster than general log-linear weight learning.

9. Conclusion and Future Work

This paper presented a new log-linear inference equation for applying Bayes nets to relational data. For a fixed template Bayes net, the equation defines the Gibbs conditional probability of a target node given an assignment of values to all other nodes. A log-linear model is defined by: a set of features, and for each feature, a feature function and a feature

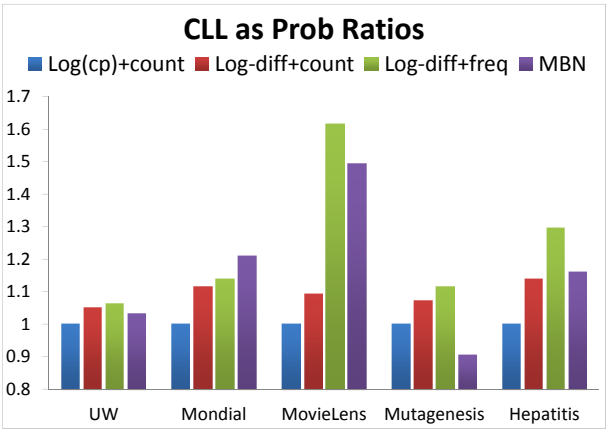

Figure 6: Predictive Performance averaged over all five benchmark databases. With Bayes net parameters, the frequency model performs better than the count model in terms of likelihood ratios.

Table 12: Parameter learning times for MBN and Bayes net methods. We characterize each database by its counts of ground atoms, tuples, and Bayes net parameters, and its time for structure learning. The Bayes net parameter count is the number of family configurations in the Bayes net.

Dataset	Literals (×1000)	Tuples (×1000)	Parms. (×10)	Struct. (s)	MBN (s)	Bayes (s)
UW	3	1	12	36	5	**2**
Mondial	2	1	58	12	90	**3**
MovieLens	170	82	33	72	10800	**8**
Mutagenesis	35	15	88	30	14400	**3**
Hepatitis	71	15	79	24	36000	**3**

weight. The predicted conditional probability is the exponentiated weighted sum of feature function values. In our proposed model, the features are all family configurations in for a family in the Bayes net, whose child node is either the target node or a child of the target node. The weight associated with a child-parent feature is computed as the log-difference of two quantities determined by the BN parameters: (i) the conditional probability of the child value, given its parent configuration, and (ii) the marginal probability of the child value. The feature function is the frequency with which the feature is instantiated in the given query, normalized with respect to relevant features only.

Our experiments on five benchmark datasets compared our log-difference frequency equation to several alternatives: using counts as feature frequencies, and using log-conditional probabilities as feature weights. We also compared using transformed Bayes net parameters as weights to using weights directly learned from the data by log-linear optimization methods. Our frequency equation achieved the best predictive performance on all but one dataset. Using the maximum likelihood values as Bayes net parameters is much faster than optimizing weights using standard log-linear methods (Markov Logic), typically seconds vs. hours.

Different model classes each have their advantages and disadvantages. Nonetheless the combination of Bayes nets and our proposed log-linear model, offers a unique set of advantages compared to other inference methods for multi-relational data, in terms of the *interpretability* and *scalability* of both structure and parameter learning: Feature weights are readily interpreted as a log-transformation of the Bayes net conditional probability parameters, and the Bayes net parameters can be computed in closed-form as the empirical frequencies.

We have established novel connections between the use of frequency feature functions and what, at first sight, appear to be unrelated issues such as the imbalance problem, pruning irrelevant features, maximum likelihood estimation, and the random selection interpretation of template Bayes nets. (1) The imbalance problem arises because feature instantiation counts in relational data can diverge by orders of magnitude. Rescaling counts as frequencies produces feature function values on the same scale. According to our experiments, changing the feature function is a more effective approach to the imbalance problem than using the weight parameters to rescale (assign smaller weight magnitudes to larger feature counts). (2) It is important not only to prune irrelevant features, but also to define instantiation frequencies over the space of relevant features only. (3) Maximum likelihood estimation is competive with optimizing weight parameters from the data only when relevant feature frequencies are used as feature functions. (4) Under mild assumptions, our frequency equation is equivalent to a *random selection* method, where the prediction score for a target node value is defined as the expected score, with respect to a random instantiation of the template Bayes net, computed using the standard Bayes net equation for a Gibbs conditional probability.

There are several avenues for future work. While we focus on Bayes nets, the imbalance problem arises also for other relational models. Our solution of changing the predictor space from counts to frequencies applies to log-linear models in general. (?, ?, ?). The frequency equation can be combined with other log-linear learning methods, for example within a model ensemble. Functional gradient boosting (?) is a powerful technique for learning such ensembles.

Our model introduces a 1-1 correspondence between log-linear weights and Bayes net parameters. Therefore log-linear regularization techniques (?) can be used for smoothing parameter estimates in template Bayes nets, and for detecting irrelevant features.

Local Gibbs probability models may be inconsistent in the sense that there is no joint distribution that agrees with the local conditional probabilities (?). An open theoretical question is whether our local frequency equations for different target nodes are guaranteed to be mutually consistent. If they are inconsistent, a possible approach is to apply the recent averaging methods for dependency networks (?, ?).

Our log-linear equation with relevant feature frequencies appears to be a principled, fast-to-learn, and accurate model for relational prediction with Bayes nets.

Acknowledgements

This work was supported by Discovery Grants to Oliver Schulte from the Natural Science and Engineering Council of Canada. Zhensong Qian was supported by a grant from the China Scholarship Council. A preliminary versions of this paper was presented at the StarAI 2012 workshop. We are indebted to workshop reviewers and participants for helpful comments.