A Tractable Pseudo-Likelihood Function for Bayes Nets Applied to Relational Data

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

Abstract

Bayes nets (BNs) for relational databases are a major research topic in machine learning and artificial intelligence. When the database exhibits cyclic probabilistic dependencies, measuring the fit of a BN model to relational data with a likelihood function is a challenge [5, 36, 28, 9]. A common approach to difficulties in defining a likelihood function is to employ a pseudo-likelihood; a prominent example is the pseudo likelihood defined for Markov Logic Networks (MLNs). This paper proposes a new pseudo likelihood P^* for Parametrized Bayes Nets (PBNs) [32] and other relational versions of Bayes nets. The pseudo log-likelihood $L^* = ln(P^*)$ is similar to the single-table BN log-likelihood, where row counts in the data table are replaced by frequencies in the database. We introduce a new type of semantics based on the concept of random instantiations (groundings) from classic AI research [12, 1]: The measure L^* is the expected log-likelihood of a random instantiation of the 1st-order variables in the PBN. The standard moralization method for converting a PBN to an MLN provides another interpretation of L^*: the measure is closely related to the log-likelihood and to the pseudo log-likelihood of the moralized PBN. For parameter learning, the L^*-maximizing estimates are the empirical conditional frequencies in the databases. For structure learning, we show that the state of the art learn-and-join method of Khosravi *et al.* [18] implicitly maximizes the L^* measure. The measure provides a theoretical foundation for this algorithm, while the algorithm's empirical success provides experimental validation for its usefulness.

Keywords: **Statistical-Relational Learning, Bayes Nets, Pseudo Likelihood, Lattice Search.**

1 Introduction

Many real-world applications store data in relational format, with different tables for entities and their links. The field of statistical-relational learning (SRL) aims to extend machine learning algorithms to relational data [10, 4]. An important machine learning task is to use data to build a *generative statistical model* that represents the joint distribution of the random variables that describe the application domain [10]. One of the most widely used generative model classes are Bayes nets (BNs) [30]. A BN structure is a directed acyclic graph (DAG) whose nodes represent random variables and whose edges represent direct statistical associations. The most common approach to SRL with graphical models is knowledge-based model construction (KBMC) [29, 22, 39]: A 1st-order or class-level model serves as a template, that is instantiated or ground with the information in the database. A major difficulty with KBMC is that the instantiated model may contain cycles even if the class-level model does not [5, 36, 9]. These difficulties have been difficult to solve; Neville and Jensen [28] conclude that "The acyclicity constraints of directed models severely limit their applicability to relational data." A major competitive advantage of undirected models (Markov nets) is that the cyclicity problem does not arise [36, 5]. In this paper we propose a new pseudo likelihood function P^* for relational BNs that is well defined even in the presence of cyclic dependencies.

Approach. We define the measure for Parametrized Bayes Nets (PBNs), an extension of BNs for relational data due to Poole [32]. The sum of a pseudo likelihood measure, over all relational database instances, does not add up to 1. It is common in statistics to use pseudo likelihood functions for learning when proper likelihood functions are conceptually or computationally difficult [2, 8, 31, 24]; see [28, Sec.3.3] for a brief overview. We define the pseudo log-likelihood $L^* = ln(P^*)$ as the sum, over all possible parent-child combinations, of the conditional log-probability of the child value given the parent values, multiplied by the frequency of the parent-child combination. Thus L^* has the same form as the single-table BN log-likelihood,

with frequencies in a database replacing row counts in a table. This paper discusses semantics and learning for L^*.

Semantics. We introduce a new type of semantics based on the concept of random instantiations from classic AI research [12, 1]. In this semantics, an expression such as $P(Flies(X)|Bird(X)) > .9$ is interpreted as "the probability that a randomly chosen bird will fly is greater than .9". For a PBN B with 1st-order variables, given an instantiation of the 1st-order variables with individual constants, we can compute from B, the log-probability of the attributes and links of the individuals. The pseudo log-likelihood L^* is the expected log-likelihood over random instantiations. We also relate the pseudo likelihood to likelihood functions for ground models: The standard moralization method converts a PBN to an MLN [5, 12.5.3]. The measure L^* is closely related to the log-likelihood and to the pseudo log-likelihood of the moralized PBN. In this sense our approach combines likelihood measures from the successful MLN paradigm with Bayes nets.

Learning. *Parameter Estimation.* Like single-table BNs, the maximum L^* estimates for the PBN parameters are the empirical conditional frequencies. The conditional frequencies can be computed from the sufficient database statistics. A number of efficient methods for computing sufficient statistics in a database have been developed [42, 26, 17].

Structure Learning. Given a model selection score S defined for single-table data, the measure L^* can be used as a log-likelihood measure with the score to define a relational version S^*. Then structure search for relational data with a pseudo likelihood score S^* can be implemented using a standard score+search algorithm. An efficient PBN structure learning method is the recent learn-and-join algorithm of Khosravi *et al.* [18]. The learn-and-join algorithm performs a level-wise model search through the table join lattice associated with a relational database, where the results of BN learning on subjoins constrain learning on larger joins. Empirical evaluation shows that the learn-and-join algorithm is orders of magnitude faster than previous MLN learning methods, and produces substantive improvements in accuracy. (10%-30% depending on the dataset used.) We show that the learn-and-join algorithm implicitly maximizes the pseudo log-likelihood L^*. This result provides a theoretical foundation for the algorithm, as well as empirical validation of the usefulness of the L^* measure. Thus while this is a conceptual paper that does not report new simulation results, it presents previous empirical results in a new light. Overall our conclusion is that relational Bayes net learning based on the pseudo likelihood measure is tractable, even in the presence of cyclic dependencies.

Paper Organization. We discuss related work and background material/notation for logic, graphical models, Parametrized Bayes Nets and Markov Logic Networks. Then we define the pseudo likelihood, and discuss the computation of its maximizing parameter values. The next section defines the new random selection semantics for PBNs, and describes the relationship to the log-likelihood and pseudo log-likelihood of MLNs. We conclude with a description of the learn-and-join algorithm and its relationship to our pseudo likelihood measure.

Contributions. The main contributions of the paper can be summarized as follows.

1. A new proposal for a pseudo likelihood function that measures the fit of a Bayes net to relational data. The function is well defined even when cyclic dependencies are present.
2. A novel semantics for the pseudo likelihood based on random groundings rather than a ground model.
3. An analytic solution for the parameter values that maximize the pseudo likelihood function. A description of efficient algorithms for parameter and structure learning with the pseudo likelihood, including the state of the art learn-and-join algorithm.

Additional Related Work. While our presentation uses PBNs as a comparatively straightforward generalization of Bayes nets for relational data, our approach applies to directed SRL models generally, such as Probabilistic Relational Models (PRMs) [9], Bayes Logic Programs (BLPs) [16], and Logical Bayesian Networks (LBNs) [6].

Cyclic Directed Models. Directed models with cycles have been studied for single-table data to model feedback effects [34, 23]. These models are based on equilibria in linear dynamic systems and do not directly apply to the discrete attribute datasets we consider, nor do they take account of relational structure. Introducing latent variables is an option for defining directed relational models without cycles [41]. Hidden variable models give rise to challenging parameter estimation problems. Methods for learning PBNs with and without hidden variables can be combined, for instance by using a PBN without hidden variables as an initial starting point for a hidden variable model.

Other Relational Pseudo Likelihood Functions. Other likelihood functions that do not require acyclicity have been developed based on Markov networks and on dependence networks. The Markov pseudo likelihood function [5, Sec.12.8] has played a key role in learning

MLNs [20, 25, 21]. A possible approach to defining a PBN pseudo likelihood is therefore to use the Markov function for the moralized PBN. As we show below, the key differences to our likelihood function are as follows. (1) Our definition uses the probability of a child node conditional on the values of its parents, rather than conditional on the values of its Markov blanket (which includes children and coparents in addition to parents). (2) We use the frequencies of child-parent configurations rather than their counts, and connect this weighting to a novel semantics based on random instantiations. Learning algorithms for MLNs have been based on Markov net methods and on Inductive Logic Programming (ILP) techniques.

Neville and Jensen developed a pseudo likelihood approach for learning *relational dependency networks* [28, 40]. Dependency networks (DNs) approximate a joint distribution as the product of conditional probabilities of each node given its Markov blanket (which renders the node conditionally independent of all others) [13]. This DN pseudo likelihood is similar to the Markov pseudo likelihood [28, Th.1], and the differences (1) and (2) above apply. Learning algorithms for dependency networks have been based on learning independent conditional probability models for each node. This requires the specification of a relational classifier to model the conditional distribution of a node given related entities and their attributes [28], (e.g., relational Naive Bayes or probability tree), often based on aggregate functions. MLNs and our BN pseudo likelihood do not require a separate relational classifier model, nor do they require aggregate functions.

To our knowledge, our proposal is the first pseudo likelihood measure for a Bayes net applied to relational data, rather than for a Markov or dependency network. As we show below, this measure supports the extension of single-table Bayes net learning algorithms to relational data.

2 Background and Notation

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

2.1 Logic and Functors. Parametrized Bayes nets are a basic SRL model; we follow the original presentation of Poole [32]. A **functor** is a function symbol or a predicate symbol. Each functor has a set of values (constants) called the **range** of the functor. A functor whose range is $\{T, F\}$ is a **predicate**, usually written with uppercase letters like P, R. A **functor random variable** is of the form $f(\tau_1, \ldots, \tau_k)$ where f is a functor and each term τ_i is a first-order variable or a constant. We also refer to functor random variables as **functor nodes**, or for short **fnodes**.[1] If an fnode $f(\boldsymbol{\tau})$ contains no variables, it is **ground**, or a **gnode**. If an fnode contains a variable, it is a **vnode**. An assignment of the form $f(\boldsymbol{\tau}) = x$, where x is a constant in the range of f, is an **atom**; if $f(\boldsymbol{\tau})$ is ground, the assignment is a **ground** atom. A **population** is a set of individuals, corresponding to a domain or type in logic. Each first-order variable X is associated with a population $\mathcal{P}_X$ of size $|\mathcal{P}_X|$. An **instantiation** or **grounding** γ for a set of variables $X_1, \ldots, X_k$ assigns a constant $\gamma(X_i)$ from the population of X_i to each variable X_i.

The functor formalism is rich enough to represent the constraints of an entity-relationship (ER) schema [37] via the following translation: Entity sets correspond to populations, descriptive attributes to functions, relationship tables to predicates, and foreign key constraints to type constraints on the arguments of relationship predicates. Figure 1 shows a simple database instance in the ER format [5].

People

Name	Smokes	Cancer
Anna	T	T
Bob	T	F

Friend

Name1	Name2
Anna	Bob
Bob	Anna

Figure 1: A simple relational database instance.

We assume that a database instance assigns a constant value to each gnode $f(\mathbf{a})$, which we denote by $[f(\mathbf{a})]_{\mathcal{D}}$. The value of descriptive relationship attributes is well defined only for tuples that are linked by the relationship. For example, $\mathit{grade}(\mathit{jack}, \mathit{101})$ is not well defined in a university database if $\mathit{Registered}(\mathit{jack}, \mathit{101})$ is false. In this case, we assign the descriptive attribute the special value $\perp$ for "undefined". Thus the atom $\mathit{grade}(\mathit{jack}, \mathit{101}) = \perp$ is equivalent to the atom $\mathit{Registered}(\mathit{jack}, \mathit{101}) = F$. Fierens *et al.* [6] discuss other approaches to this issue. The results in this paper extend to functors built with nested functors, aggregate functions, and quantifiers (e.g., existential); for the sake

[1] The term "functor" is used as in Prolog [3]. In Prolog, a functor random variable is called a "structure". Poole [32] refers to a functor random variable as a "parametrized random variable". We use the term "fnode", (1) for brevity, (2) to emphasize the use of function/predicate symbols, (3) to avoid confusion with the statistical sense of "parametrized", meaning that values have been assigned to parameters.

of notational simplicity we do not consider more complex functors explicitly.

2.2 Bayes Nets. We employ notation and terminology from [30, 35] for a Bayesian Network. A **Bayes net structure** is a directed acyclic graph (DAG) G, whose nodes comprise a set of random variables denoted by V. We consider only discrete finite random variables. The parents of node v_i in graph G are denoted by $\mathbf{PA}_i$, and an assignment of values to the parents is denoted as $\mathbf{pa}_i$. A Bayes net (BN) is a pair $\langle G, \boldsymbol{\theta}_G \rangle$ where $\boldsymbol{\theta}_G$ is a set of parameter values that specify the probability distributions of children conditional on instantiations of their parents, i.e. all conditional probabilities of the form

$$\theta_{ijk} \equiv P(v_i = x_{ik} | \mathbf{PA}_i = \mathbf{pa}_{ij}),$$

where x_{ik} is the k-th possible value of node i and $\mathbf{pa}_{ij}$ is the j-th possible configuration of the parents of v_i. The conditional probabilities are specified in a **conditional probability table** for variable v_i or CP-table. A **Parametrized Bayes net**, or PBN, is a Bayes net whose nodes are functor random variables. We use the following notation for describing the CP parameters in a PBN.

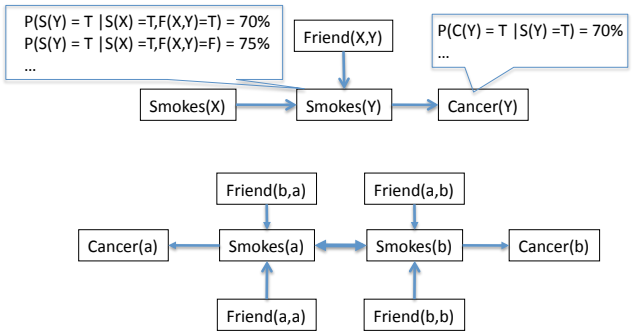

Figure 2: A PBN and its grounding for the database of Figure 1. The double arrow $\leftrightarrow$ is equivalent to two directed edges.

- Let F_{ijk} be the **family formula** that expresses that fnode f_i is assigned its k-th value, and the state of its parents is assigned its j-th value.
- $n_{ijk}(\mathcal{D})$ is the number of groundings of F_{ijk} that evaluate as true in $\mathcal{D}$.
- If $X_1, \ldots, X_k$ are the variables that occur in the family formula F_{ijk}, then its **domain** is the Cartesian product $\mathcal{P}_{X_1} \times \cdots \times \mathcal{P}_{X_k}$. Since the domain of F_{ijk} depends only on i (the family formulas F_{ijk} and $F_{ij'k'}$ contain the same variables), we write

$$m_i \equiv \prod_{i=1}^{k} |\mathcal{P}_{X_i}|$$

for the **domain size** of the families of child node i.

- $p_{ijk} \equiv n_{ijk}/m_i$ is the **empirical frequency** of F_{ijk} in database $\mathcal{D}$.

A **ground** graph $\overline{B}$ is derived from B by instantiating the functor nodes in B in every possible way. There is a directed edge $f_1(\boldsymbol{a}_1) \rightarrow f_2(\boldsymbol{a}_2)$ in $\overline{B}$ just in case there is an edge $f_1(\boldsymbol{\tau}_1) \rightarrow f_2(\boldsymbol{\tau}_2)$ in B and there is a grounding γ such that $\gamma(\boldsymbol{\tau}_i) = \boldsymbol{a}_i$, for $i = 1, 2$.

Examples. The following examples refer to the DB instance of Figure 1. Figure 2 illustrates PBN concepts. We use a Prolog-style list notation for a conjunction of atoms. An example of a family formula F_{ijk} with child node $f_i = \mathit{Smokes}(Y)$ is

$$\mathit{Smokes}(Y) = T, \mathit{Smokes}(X) = T, \mathit{Friend}(X, Y) = T.$$

From Figure 2, the associated conditional probability is $\theta_{ijk} = 70\%$. The number of true groundings is $n_{ijk} = 2$. For the domain size we have

$$m_{\mathit{Smokes}(Y)} = |\mathcal{P}_X| \cdot |\mathcal{P}_Y| = 4.$$

Therefore the database frequency of this formula is $p_{ijk} = \frac{n_{ijk}}{m_{\mathit{Smokes}(Y)}} = 1/2$.

A family formula with child node $\mathit{Cancer}(Y)$ is

$$\mathit{Cancer}(Y) = T, \mathit{Smokes}(Y) = T.$$

The associated conditional probability parameter is 70%. The number of true groundings is 1. For the domain size $m_{\mathit{Cancer}(Y)} = |\mathcal{P}_Y| = 2$. Therefore the empirical frequency of this formula is 1/2.

2.3 Markov Logic Networks. Our presentation of MLNs follows [5]. The set of first-order formulas is defined by closing the set of atoms under Boolean combinations and quantification. A Markov Logic Network (MLN) M is a set $\{(\phi_1, w_1), \ldots, (\phi_m, w_m)\}$ where ϕ_i is a formula, and each w_i is a real number called the weight of ϕ_i. We write $n_i(\mathcal{D})$ for the number of satisfying instantiations of formula ϕ_i in database $\mathcal{D}$. The log-likelihood of an MLN M is given by

$$L_M(\mathcal{D}) = \sum_{i}^{m} w_i n_i - ln(Z_{\mathbf{w}}) \tag{2.1}$$

where $Z_{\mathbf{w}}$ is a normalization constant that depends on the weights but not on the database $\mathcal{D}$. A PBN can be converted to an MLN by the standard moralization method [5, 12.5.3], where we have a formula for each CP-table row whose weight is $ln(\theta_{ijk})$. Thus the MLN $M(B)$ associated with PBN B is the set $\{(F_{ijk}, ln(\theta_{ijk}))\}$. We refer to $M(B)$ as a **moralized Bayes net** (MBN).

Figure 3 shows an MBN and illustrates the Markov network associated with an MLN M. The nodes are the functor nodes that occur in the formulas of M. There is an edge between two fnodes just in case they occur together in some formula ϕ_i. For a given database $\mathcal{D}$, we can construct a ground Markov network just as with a PBN. In this ground network, each instantiation of an MLN formula ϕ_i corresponds to an assignment of values to a clique whose potential is e^{w_i}. In the case of an MBN, the nodes are gnodes, the cliques are ground instances $\overline{f}_{ijk}$ of family formulas, the corresponding clique potential is the conditional probability θ_{ijk}, and the assignment of values to ground nodes is specified by the DB instance $\mathcal{D}$.

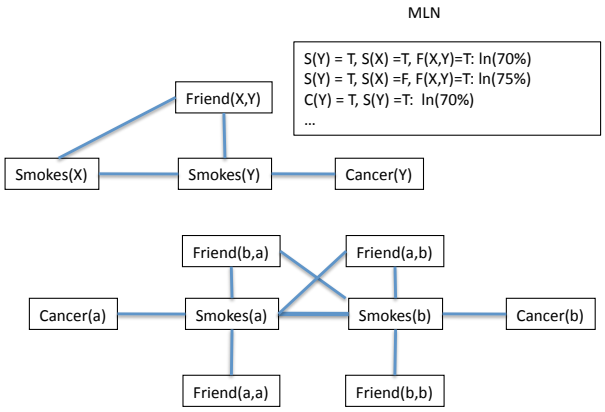

Figure 3: The moralized PBN of Figure 2 and its ground Markov network for the database of Figure 1.

3 Pseudo-likelihood for Parametrized Bayes Nets

We propose the following pseudo log-likelihood function for a PBN B:

$$L^*_B(\mathcal{D}) = \sum_{ijk} p_{ijk}(\mathcal{D}) \cdot ln(\theta_{ijk}). \tag{3.2}$$

The formula may be read as follows. For each possible parent-child configuration, find the corresponding conditional log-likelihood from the PBN B, multiply it by the frequency of the parent-child configuration in the given database $\mathcal{D}$, and sum the results. This compares to the log-likelihood assigned to a single data table D by a BN G as follows. If $\mathbf{V} = \mathbf{x}$ is a joint assignment of values to the nodes in a Bayes net $\langle G, \boldsymbol{\theta}_G \rangle$, its probability $P_G(\mathbf{V} = \mathbf{x})$ is obtained by multiplying the conditional probabilities of each node value assignment given its parent value assignments. So the log-likelihood of the data table is given by

$$ln(P_G(D)) = \sum_{ijk} n_{ijk}(D) \cdot ln(\theta_{ijk}) \tag{3.3}$$

where $n_{ijk}(D)$ is the number of rows in D where the family state F_{ijk} occurs. Thus the relational equation (3.2) is obtained from the single-table equation (3.3) by replacing counts of rows in the data table by instantiation frequencies in the database. We discuss the motivation for this change.

Frequencies vs. Counts. If the population size m_i for functor node f_i is significantly greater than that for f_l, the corresponding terms n_{ijk} range over a larger scale than the n_{ljk} terms. Viewing Equation 3.2 as a log-linear model, this means that scales of the independent variables can differ by orders of magnitude, which is undesirable for learning. Since the $ln(\theta_{ijk})$ weights are negative, using counts for model selection heavily penalizes structures whose families include more first-order variables. Replacing counts by frequencies puts all factors on the same [0,1] scale. For the same reason, MLN researchers usually scale counts to frequencies in the pseudo likelihood for MLNs [21, 25] (cf. Section 4.3). Raina *et al.* use a similar normalization in a log-linear classification model [33]. As we show below, using frequencies has the desirable consequence of making the pseudo likelihood invariant under equivalence transformations of the database design such as table normalization.

Likelihood Maximization. The parameter values that maximize Equation (3.2) are the empirical conditional frequencies in a database.

PROPOSITION 3.1. *The parameter values $\widehat{\theta}_{ijk}$ that maximize the pseudo log-likelihood $L^*_B(\mathcal{D})$ are the conditional empirical frequencies:*

$$\widehat{\theta}_{ijk} = \frac{p_{ijk}}{\sum_{k'} p_{ijk'}}.$$

The result follows because the pseudo likelihood has the same form as the BN log-likelihood, so the standard MLE argument for BN estimation applies. (Replacing counts by frequencies adds a constant factor that does not affect the argument).

Computing MLE Values. The conditional database frequencies for computing $\widehat{\theta}_{ijk}$, can be found from the *sufficient database statistics* p_{ijk}, the joint frequencies of parent-child combinations. Because they are required in many applications, several practical algorithms for computing sufficient statistics in a database have been developed, as well as techniques for optimizing runtime and memory usage [42, 26]. The unrestricted problem of computing the number of groundings of an arbitrary first-order clause is #P-complete [5, Prop.12.4]. However, Vardi shows that the problem is solvable in polynomial time given a constant bound *on the number of first-order variables* that occur in the formula [38]. This is a natural syntactic complexity measure that bounds the number of (generic) objects that are under consideration in a single formula. In SRL models, it is rare to find more than 3 or 4 variables in a single rule or formula. Khosravi *et al.* [17] present a dynamic programming algorithm for computing conditional CP-table entries in a PBN and provide empirical evidence for its feasibility. In sum, the PBN pseudo likelihood is analytically and computationally tractable. To gain further theoretical understanding of this measure, we consider several ways to derive Equation (3.2).

4 Semantics for the Pseudo Likelihood

KBMC or model grounding treats the PBN as specifying a 1st-order pattern of dependencies that is applied to *every* possible instantiation. Classic AI research by Halpern and Bacchus developed a probabilistic semantics for first-order formulas based on the idea of a *random* instantiation [12, 1]. In this section we show that random instantiations provide a semantics for the PBN pseudo likelihood (3.2). We also show that the version of Equation (3.2) with counts instead of frequencies can be given several KBMC interpretations.

4.1 Random Selection Semantics for the Pseudo Log-Likelihood.

We review the random selection semantics briefly in the context of a functor language; extensive treatment in general 1st-order logic was provided by Halpern and Bacchus [12, 1]. The key idea is to view a first-order logical variable as a random variable (population variable) that selects a member of its population with a given probability. In the remainder we assume that the selection is uniform (cf. [12, fn.1]), so for a single population variable X we have

$$P(X = a) = \frac{1}{|\mathcal{P}_X|}.$$

Different population variables are assumed to be mutually independent, so the joint distribution over population variables is uniform as well. Since a function of a random variable is itself a random variable, a distribution over population variables defines a distribution over assignments of values to functor nodes (i.e., a distribution over vnodes).

We apply the concept of a random instantiation to define a pseudo log-likelihood for a PBN B as follows. Let $X_1, \ldots, X_k$ be a list of *all* variables that occur in the fnodes of B. (1) Randomly select an instance (constant) a_i from the population of variable X_i, for each $i = 1, \ldots, k$. (2) Replace each variable in B with the corresponding instance so each vnode in B becomes a gnode. (3) For each gnode in the instantiated BN, assign to it the value defined by the database $\mathcal{D}$. (4) Compute the log-likelihood of this joint assignment using the usual product formula; this defines a log-likelihood for the random instantiation $\mathbf{a_i}$. The expected value of this log-likelihood is the **random log-likelihood** of database $\mathcal{D}$ given PBN B. Figure 4 shows a random instantiation of the PBN in Figure 2 with the corresponding assignment defined by the DB instance of Figure 1; see also Table 1.

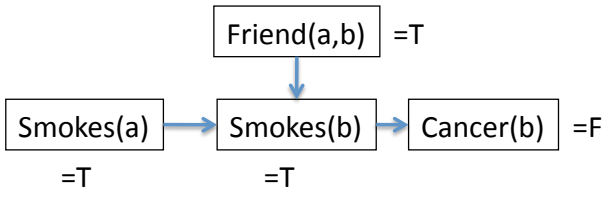

Figure 4: The computation of $P_B^{\gamma}(\mathcal{D})$ for the PBN of Figure 2 and the instantiation $\gamma X = Anna = a$, $\gamma Y = Bob = b$.

For a formal definition, in this section only we use γ to denote a simultaneous substitution of *each* variable in B. We write Γ for the space of all possible groundings of the variables in B; so $\Gamma = \mathcal{P}_{X_1} \times \cdots \times \mathcal{P}_{X_k}$. In the PBN of Figure 2, there are 4 possible groundings of the two variables X, Y, so we have $|\Gamma| = 4$. Applying γ to a vnode $f(\boldsymbol{\tau})$ defines a gnode $\gamma(f(\boldsymbol{\tau}))$; to simplify notation, we write $\gamma(f)$ when the arguments to the functor are not relevant. Recall that $[\gamma(f(\boldsymbol{\tau}))]_{\mathcal{D}}$ denotes the value determined by $\mathcal{D}$ when f is applied to ground term $\gamma(\boldsymbol{\tau})$. For instance, $[Cancer(Anna)]_{\mathcal{D}} = T$ in our sample DB. So the likelihood of the "database slice" defined by γ is given by

$$P_B^{\gamma}(\mathcal{D}) = \prod_i P_B(f_i = [\gamma(f_i)]_{\mathcal{D}} | \mathbf{pa}_i = [\gamma(\mathbf{pa}_i)]_{\mathcal{D}})$$

and the expected value of the log-likelihood $ln(P_B^{\gamma}(\mathcal{D}))$

over all possible equiprobable groundings, is given by

$$L_B^\Gamma(\mathcal{D}) \equiv \frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma} ln(P_B^\gamma(\mathcal{D})) \tag{4.4}$$

where $L_B^\Gamma(\mathcal{D})$ denotes the random log-likelihood for the space of possible PBN groundings Γ. The next proposition shows that the random selection semantics validates the pseudo log-likelihood defined by Equation (3.2).

PROPOSITION 4.1. *Let B be a PBN and $\mathcal{D}$ a relational database. Then*

$$L_B^*(\mathcal{D}) = L_B^\Gamma(\mathcal{D}).$$

Proof. For each family formula F_{ijk}, let $\gamma_{ijk}(\mathcal{D})$ be the number of simultaneous groundings of all variables in B that satisfy F_{ijk}. Write r_i for the size of the Cartesian product of the populations of variables that do *not* occur in F_{ijk}. For instance, if variables X_1, X_2 occur in F_{ijk}, then $r_i = \prod_{l=3}^k |\mathcal{P}_{X_l}|$. Then we have

$$\begin{aligned}\gamma_{ijk}(\mathcal{D}) &= n_{ijk}(\mathcal{D})\cdot r_i\\ |\Gamma| &= m_i \cdot r_i\end{aligned}$$

Therefore $p_{ijk}(\mathcal{D}) = \gamma_{ijk}(\mathcal{D})/|\Gamma|$ and the pseudo log-likelihood (3.2) can be written as

$$\sum_{ijk} p_{ijk}(\mathcal{D})\cdot ln(\theta_{ijk}) = \frac{1}{|\Gamma|}\sum_{ijk}\gamma_{ijk}(\mathcal{D})\cdot ln(\theta_{ijk}). \tag{4.5}$$

Each factor $ln(\theta_{ijk})$ in Equation (4.4) appears in the sum $\sum_{\gamma\in\Gamma} ln(P_B^\gamma(\mathcal{D}))$ once for each simultaneous grounding $\gamma \in \Gamma$ that satisfies F_{ijk} in database $\mathcal{D}$. Therefore we have

$$\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma} ln(P_B^\gamma(\mathcal{D})) = \frac{1}{|\Gamma|}\sum_{ijk}\gamma_{ijk}(\mathcal{D})\cdot ln(\theta_{ijk}). \tag{4.6}$$

Equations (4.6) and (4.5) together establish the identity of the pseudo log-likelihoods (4.4) and (3.2).

4.2 Random Selection Semantics for the Pseudo Likelihood. The random likelihood version of Equation (4.4), given by

$$P_B^\Gamma(\mathcal{D}) \equiv \exp(L_B^\Gamma(\mathcal{D})) = \prod_{\gamma\in\Gamma} P_B^\gamma(\mathcal{D})^{\frac{1}{|\Gamma|}} \tag{4.7}$$

has a useful interpretation as well. Consider each simultaneous instantiation γ of the variables in B as a constant-size *hyperunit* (similar to a hyperedge in a hypergraph). Then we can think of the values of the ground functor nodes that the data determine for γ as feature values for the hyperunit. Equation (4.7) computes the product over all hyperunits, of the BN-probability of the hyperunit's features, raised to the root of the number of hyperunits. In other words, it is the *geometric mean*[2] *of the product of feature vector probabilities for hyperunits*; see Table 1 for illustration. Since hyperunits have individuals in common, they are interdependent. The geometric mean is a smoothed product likelihood that adjusts for the dependencies.

	Hyperentity		Hyperfeatures						
Γ	X	Y	F(X,Y)	S(X)	C(X)	S(Y)	C(Y)	P_B^γ	$ln(P_B^\gamma)$
γ_1	Anna	Bob	T	T	T	T	F	0.105	-2.254
γ_2	Bob	Anna	T	T	F	T	T	0.245	-1.406
γ_3	Anna	Anna	F	T	T	T	T	0.263	-1.338
γ_4	Bob	Bob	F	T	F	T	F	0.113	-2.185

Table 1: The single-table interpretation of the random likelihood for the PBN of Figure 2 and the database of Figure 1. A simultaneous grounding of *all* variables in the PBN defines a hyperentity. The values of functors for the hyperentity define its hyperfeatures. The PBN assigns a likelihood to the hyperfeatures. The rounded numbers shown were obtained using the CP parameters of Figure 2 together with $P_B(Smokes(X) = T) = 1$ and $P_B(Friend(X, Y) = T) = 1/2$, chosen for easy computation. (a) The random likelihood is the geometric mean of the joint probabilities given by $(0.105 \cdot 0.245 \cdot 0.2625 \cdot 0.1125)^{1/4} \approx 0.166$. (b) The random log-likelihood is the average of the log-likelihoods for each grounding, given by $-(2.254 + 1.406 + 1.338 + 2.185)/4 \approx -1.8$. By Proposition 4.1, this equals our PBN pseudo log-likelihood.

Schema Invariance. The fact that the PBN pseudo likelihood is equivalent to an expression defined in terms of a single (hyper)population has the important consequence that it is invariant under syntactic equivalence transformations of the database. For instance, database normalization operations may move information about a descriptive attribute from one table to another [15]. For any fixed set of populations (entity types), such operations do not affect the pseudo likelihood because they do not change the feature values associated with a hyperunit.

To illustrate, suppose we have a university database with courses and instructors. Course attributes include *level* and *difficulty*. There is also a relationship $Teaches(C, P)$ that records which professor teaches which course; there is a unique instructor for each course. Now a DB design may include the course at-

[2]The geometric mean of $x_1, \ldots, x_n$ is $(\prod_i x_i)^{1/n}$.

tributes in the *Teaches* table as descriptive relationship attributes, so they correspond to fnodes $level(C, P)$ and $difficulty(C, P)$. If the level of a course predicts its difficulty, a PBN would include an edge

$$level(C, P) \rightarrow difficulty(C, P).$$

Then the number of groundings of a family formula like

$$level(C, P) = hi, difficulty(C, P) = hi$$

will be the number of courses with the corresponding attributes, multiplied by the irrelevant number of professors in the database. In contrast, the frequency of the family formula will be the frequency of courses with the corresponding attributes, since the irrelevant factor cancels out. A similar syntactic invariance holds with respect to transformations of the model, rather than the database: Adding irrelevant parents with additional variables increases the number of satisfying groundings of a family formula but not their frequency.

4.3 KBMC/Grounding Semantics. There does not appear to be a direct KBMC semantics for the PBN pseudo log-likelihood L^* with frequencies. But there are several possibilities for the variant with counts:

$$L_B^+(\mathcal{D}) = \sum_{ijk} n_{ijk}(\mathcal{D}) \cdot ln(\theta_{ijk}). \tag{4.8}$$

(1) *Product Formula.* For a PBN B, we may apply the usual BN product scheme as follows: for each instantiation of a formula formula $\overline{F}_{ijk}$, multiply the associated conditional probabilities θ_{ijk}. The logarithm of the product leads to Equation (4.8).

(2) *Moralized Markov Log-likelihood.* Applying Equation (2.1), the MLN log-likelihood function for the moralized PBN $M(B)$ is

$$L_{M(B)}(\mathcal{D}) = \sum_{ijk} n_{ijk}(\mathcal{D}) \cdot ln(\theta_{ijk}) - ln(Z_{\boldsymbol{\theta}}). \tag{4.9}$$

Omitting the partition function component $ln(Z_{\boldsymbol{\theta}})$ leads to Equation (4.8). The normalization constant Z is not related to how well the model fits the facts in a database.

(3) *Markov Pseudo Likelihood.* Because of the intractability of the partition function, MLN researchers have based learning on the Markov network pseudo likelihood [5, Sec.12.8]. Equation (4.8) is related to the MLN pseudo likelihood for $M(B)$ as follows. The Markov pseudo likelihood [2] of an assignment $\mathbf{V} = \mathbf{x}$ of values to each node is the product, over all nodes v, of the probability that $v = x$ conditional on the neighbors of v, where x is the value of v specified by $\mathbf{x}$. This conditional probability is the product of all clique potentials in which v participates, divided by a local normalization constant Z_v. For an MBN, this means that each clique potential is counted *once for each clique family member*. Given the interpretation of the θ_{ijk} potentials as conditional probabilities, a natural alternative is to count each potential only *once overall*, as the conditional probability of the family's child. Making this change, omitting the local normalization constants Z_v, and taking logarithms leads to Equation (4.8). Intuitively, the Markov pseudo likelihood computes the probability of the value of each node in the ground network given its *neighbors* (Markov blanket), then multiplies these probabilities. The PBN pseudo likelihood with counts computes the probability of the value of each node in the ground network given its *parents*, which are a subset of its Markov blanket, then multiplies these probabilities.

Discussion. The semantic interpretations of the PBN pseudo likelihood function indicate that our approach is broadly applicable to Bayes net-like SRL models, either directly or through the connection with the count version (4.8). (i) The pseudo likelihood can be used for any directed SRL model whose structure converts to an MLN structure, and whose conditional probability parameters translate into weights via moralization. As MLNs are intended to be a unifying framework for SRL [5, 12.5], this includes a large class of SRL models, such as PRMs [9], BLPs [16], and LBNs [6]. (ii) At the ground model level, the pseudo likelihood can be used with any directed SRL model that via KBMC defines a ground directed graph with local CP-table parameters, which again includes a large class as KBMC is the dominant type of SRL semantics. (iii) The random selection semantics applies to any logic-based SRL model that includes 1st-order variables, such as BLPs, LBNs, PBNs. In the remainder of the paper we consider structure learning with the PBN pseudo likelihood.

5 Model Selection with the Pseudo Likelihood

An important role of a likelihood function is to guide model selection. A single-table model selection score has the form $S(G, D)$ where G is a graphical model and D a data table. We consider scores that trade off data fit against model complexity, and that can be computed given the following quantities.

1. The log-likelihood $L_G(D)$.
2. The sufficient data statistics for joint events $p_{ijk}(D)$.
3. The number of parameters par_G.
4. The sample size m = number of rows in D.

An example is the Bayes Information Criterion (BIC) score [27, Ch.8.3.2]

$$BIC(G, D) \equiv ln(P_{\widehat{G}}(D)) - par(G) \cdot \frac{ln(m)}{2}$$

where $\widehat{G}$ is the BN G with its parameters instantiated by the maximum likelihood estimates given the datatable D, and $par(G)$ is the number of free parameters in the structure G. We can define a relational model selection score $S^*(B, \mathcal{D})$ by substituting the pseudo log-likelihood L^*_B for the single-table likelihood. For instance, the relational BIC version would be

$$BIC^*(G, \mathcal{D}) \equiv L^*_{\hat{B}}(\mathcal{D}) - par(B) \cdot \frac{ln(m)}{2}.$$

Provided that the relational model selection score can be efficiently evaluated, standard BN search+score structure learning algorithms can be applied. We discuss the computation of the components of a relational score S^* in turn.

(1) *Likelihood Computation.* The computation of the pseudo log-likelihood requires only the computation of sufficient DB statistics (Section 3).

(2) *Parameter Complexity.* The number of parameters $par(B)$ is the same for a PBN as for a standard BN (typically the number of CP-table entries).

(3) *Sample Size.* Determining a global sample size m is not straightforward when the data are distributed across tables. A reasonable possibility is to use the fact that most score functions decompose, that is, they can be written as the sum of local scores $\sum_i S_i(G, D)$, where each local score involves only node i and its parents. The number m_i of possible instantiations of the F_i family is a plausible local score size. This can be justified in terms of the hyperpopulation discussed in Section 4. If we use $m \equiv |\Gamma| = |\mathcal{P}_{X_1}| \cdot |\mathcal{P}_{X_2}| \cdots |\mathcal{P}_{X_k}|$ as a global sample size, then $ln(|\Gamma|) = \sum_{j=1}^{k} ln(|\mathcal{P}_{X_j}|)$. For the BIC difference $BIC(B, \mathcal{D}) - BIC(B', \mathcal{D})$ between two PBNs that differ only in a local family, the difference in the sample size term reduces to the difference in the local sample sizes m_i. In the next section we show that a state of the art PBN structure learning algorithm maximizes the pseudo likelihood.

6 Empirical Results: The Learn-and-Join Algorithm

Khosravi *et al.* present the learn-and-join (LAJ) structure learning algorithm [18]. The algorithm upgrades a single-table BN learner for relational learning. We briefly review the algorithm and its empirical performance, then discuss how it relates to the PBN pseudo likelihood.

6.1 Review: The Learn-and-Join Algorithm

The key idea of the algorithm can be explained in terms of the *table join lattice.* Recall that the (natural) join of two or more tables, written $T_1 \bowtie T_2 \cdots \bowtie T_k$ is a new table that contains the rows in the Cartesian products of the tables whose values match on common fields. A table join corresponds to logical conjunction [37]. Say that a join table J is a **subjoin** of another join table J' if $J = J' \bowtie J^*$ for some join table J^*. If J is a subjoin of J', then the fields (columns) of J are a subset of those in J'. The subjoin relation defines the table join lattice. The basic idea of the learn-and-join algorithm is that join tables should inherit edges between descriptive attributes from their subjoins. This gives rise to the following constraints for two attributes v_1, v_2 that are both contained in some subjoin of J. (i) v_1 and v_2 are adjacent in a BN B_J for J if and only if they are adjacent in a BN for some subjoin of J. (ii) If all subjoin BNs of J orient the link as $v_1 \rightarrow v_2$ resp. $v_1 \leftarrow v_2$, then B_J orients the link as $v_1 \rightarrow v_2$ resp. $v_1 \leftarrow v_2$. The learn-and-join algorithm then builds a PBN for the entire database $\mathcal{D}$ by level-wise search through the table join lattice. The user chooses a single-table BN learner. The learner is applied to table joins of size 1, that is, regular data tables. Then the learner is applied to table joins of size $s, s+1, \ldots$, where the constraints (i) and (ii) are propagated from smaller joins to larger joins.

Example. We illustrate the learn-and-join algorithm on the example database of Figure 1. Applying the single-table BN learner to the *People* table may produce a single-edge graph *Smokes* $\rightarrow$ *Cancer*. Then we form the join

$$J = People \bowtie Friend \bowtie Friend;$$

this join table has four attribute columns that indicate the values of *Smokes* and *Cancer* for each member of a pair in the *Friend* table. Using the functor notation to distinguish different "copies" of the same attribute, we may denote the columns as $Smokes(X), Smokes(Y), Cancer(X), Cancer(Y)$. The BN learner is applied to J, with the constraint from the *People* BN that there must be an edge

$$Smokes(Y) \rightarrow Cancer(Y).$$

Also, the algorithm is constrained such that no edges may point into $Smokes(X)$ or $Cancer(X)$. The intuition behind this constraint is that it suffices to model the conditional distribution of just one "copy" of the *Smokes* attribute, namely the vnode $Smokes(Y)$. The BN learner then may find an edge

$$Smokes(X) \rightarrow Smokes(Y).$$

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

Table 2: Runtime to produce a parametrized MLN, in minutes. The LAJ column shows structure learning time + weight learning time for the Learn-and-Join Algorithm. The other columns show the run-times for previous MLN learning methods. Since these methods do not scale to the full datasets, randomly constructed subsample databases were used.

Since the dependency represented by this edge is valid only for pairs of people that are friends (i.e., conditional on $Friend(X, Y) = T$), the algorithm adds an edge $Friend(X, Y) \rightarrow Smokes(Y)$. The final result is the PBN shown in Figure 2.

Evaluation. Khosravi *et al.* present a complexity analysis that shows that the edge inheritance constraint keeps the BN model search space roughly constant for each join table. We review briefly the main points of their empirical findings [18]. On benchmark datasets the learn-and-join algorithm produces a PBN 1,000-10,000 times faster than the state-of-the art Markov Logic Network learners; see Table 2. The table is from [18]. The databases MovieLens and Mutagenesis are standard benchmarks; for details on the datasets and experimental setup see [18].

To evaluate the predictive accuracy of the learned PBN models, Khosravi *et al.* applied the learned PBNs with MLN inference techniques: Each PBN structure was converted to an MLN set of formulas using the standard moralization method, and standard weight learning methods were used to estimate the MLN parameters for the MLN structure. In the context of PBN learning with the pseudo-likelihood, applying MLN inference is natural since the pseudo-likelihood function for the PBN is similar to the likelihood function of the converted MLN (Section 4.3). Moreover, MLN inference techniques are among the state-of-the-art in SRL, so this experiment compares the predictive performance of the learned JBNs with a high-quality competitor. The predictive accuracy of the MBN models using MLN inference was substantially higher than that of the models learned with MLN methods. Depending on the dataset used, improvement ranged from 15% to 25%. Table 3 from [18] provides details for the benchmark datasets MovieLens and Mutagenesis. Other standard measures, such as CLL, the average log-likelihood of database facts (ground atoms) predicted by the model, and area-under-curve show similar improvement.

6.2 Relationship to Pseudo Likelihood Learning. We now discuss how the learn-and-join algorithm may be viewed as an instance of *context-specific* BN learning with the pseudo likelihood. In the BN learning literature, a **context** is a specification of values for some of the variables in the BN [7]. A context-specific dependence between variables v_1 and v_2 is one that holds conditional on a context. Several BN learning method for context-specific dependencies have been developed [7]. Extending our previous notation, we write $n_{ijk}(\mathcal{D})|(\mathbf{V} = \mathbf{x})$ for the number of groundings in database $\mathcal{D}$ that satisfy both the family formula F_{ijk} and the context $\mathbf{V} = \mathbf{x}$, and $p_{ijk}(\mathcal{D})|(\mathbf{V} = \mathbf{x})$ for the frequency of F_{ijk} in database $\mathcal{D}$ conditional on the context. The LAJ algorithm can be seen as learning with the PBN pseudo likelihood in a context-specific manner, where the contexts are conjunctions (joins) of the form $R_1 = T, \ldots, R_k = T$.

(1) *Sufficient Statistics.* We observe that for a join table J, *the sufficient statistics computed from J are the sufficient database statistics conditional on the corresponding relationship(s) being true.* For instance, consider the *Friend* table with corresponding fnode $Friend(X, Y)$ or $F(X, Y)$ for short. The LAJ algorithm uses the join table

$$J = People \bowtie Friend \bowtie Friend.$$

For a family formula F_{ijk} involving only variables X, Y and the atom $F(X, Y) = T$, such as

$$F(X, Y) = T, Smokes(X) = T, Smokes(Y) = F$$

the number of rows in the join table satisfying F_{ijk} is the number of instantiations that satisfy the conjunction in the data base. Therefore we have $n_{ijk}(\mathcal{D}) = n_{ijk}(Friend)$. The equivalence of the sufficient statistics for relationship tables holds also for larger joins of relationship tables. The data join table J is equivalent to restricting the Grounding Table 1 to the rows with $Friend(X, Y) = T$.

(2) *Parameter Complexity.* The learn-and-join algorithm computes the parameter count as the number of CP-table entries without the Boolean relationship indicator R. Therefore it is equivalent to the number of CP-table entries with $R = T$, which is the local parameter count for that context.

(3) *Sample Size.* The learn-and-join algorithm treats the sample size as the number of rows in the current join table, which is the number of groundings that satisfy the specified relational context.

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

Table 3: The table shows predictive performance for the models obtained by LAJ vs. previous MLN structure learning methods. Training was performed on 2/3 of the database and testing on the other 1/3. More details and further simulation results are given by Khosravi *et al.* [18].

(4) *Likelihood Computation.* The current version of the learn-and-join algorithm considers cross-table correlations only conditional on the existence of relationships. Applying a standard BN learner to a join table implicitly uses the count likelihood (4.8) rather than the frequency version (3.2). Nonetheless, the empirical results of Khosravi *et al.* are a positive test for both the count version and the frequency version, for the following reasons. (i) Because the BN learner is applied to a join table J that fixes the local relational context, and with it the 1st-order variables in the PBN, there is no problem with a bias against structures with more 1st-order variables. (ii) The simulations employed the BDeu score with a uniform structure prior. This score represents a likelihood function $P_B(T)$ derived from Bayesian hyperparameters [14], without any other term, such as an explicit penalty term for free parameters. Therefore dividing the log-likelihood $ln(P_B(J))$ by the join table size does not make a difference to BDeu maximization, that is, for this score database counts and database frequencies are equivalent.

In sum, we may view the learn-and-join algorithm as an context-specific structure learning method for the pseudo log-likelihood (3.2), where the contexts considered are joins of relationship tables.

7 Conclusion

We proposed a new pseudo likelihood function P^* to measure the fit of a Bayes net to relational data. This function is well-defined even in the presence of cyclic dependencies. The form of the pseudo likelihood function P^* is very similar to that of the standard single-table BN likelihood, replacing counts in a data table by frequencies in a database. The function P^* has several attractive theoretical and computational properties. (1) Parameter learning is efficient as the P^* maximizing estimates are the empirical conditional frequencies. (2) There is a new type of semantics for P^* in terms of random instantiations. The function is also closely related to the log-likelihood of Markov Logic Networks. (3) The fast Learn-and-Join Algorithm for BN structure learning in relational data is a context-specific learner for P^*, where the contexts considered specify the existence of various relationship chains between entities.

Acknowledgements

This research was supported by a Discovery Grant from the Natural Sciences and Engineering Council of Canada (NSERC). Preliminary results were presented to the AI group at the University of British Columbia, the IJCAI-STRUCK and IJCAI-GKR workshops (2009), and the Formal Epistemology Conference at Carnegie Mellon University (2010). We are grateful to the audiences for helpful questions and comments, especially Clark Glymour, Brandon Fitelson and Peter Spirtes. Anonymous referees provided useful suggestions.

References

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

[2] J. Besag, *Statistical analysis of non-lattice data*, The Statistician, 24 (1975), pp. 179–195.

[3] I. Bratko, *Prolog (3rd ed.): programming for artificial intelligence*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[4] L. de Raedt, *Logical and Relational Learning*, Cognitive Technologies, Springer, 2008.

[5] P. Domingos and M. Richardson, *Markov logic: A unifying framework for statistical relational learning*, in Introduction to Statistical Relational Learning [11].

[6] D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon, *Logical bayesian networks and their relation to other probabilistic logical models*, in ILP, S. Kramer and B. Pfahringer, eds., vol. 3625 of Lecture Notes in Computer Science, Springer, 2005, pp. 121–135.

[7] N. Friedman and M. Goldszmidt, *Learning Bayesian networks with local structure*, in NATO ASI on Learning in graphical models, 1998, pp. 421–459.

[8] S. Geman and C. Graffine, *Markov random field image models and their application to computer vision.*, in Proceedings of the 1986 International Congress of Mathematicians, A. Gleason, ed., American Mathematical Society, 1987, pp. 1496–1517.

[9] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar, *Probabilistic relational models*, in Introduction to Statistical Relational Learning [11], ch. 5, pp. 129–173.

[10] L. Getoor and B. Taskar, *Introduction*, in Getoor and Taskar [11], pp. 1–8.

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

[12] J. Y. Halpern, *An analysis of first-order logics of probability*, Artificial Intelligence, 46 (1990), pp. 311–350.

[13] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, C. Kadie, and P. Kaelbling, *Dependency networks for inference, collaborative filtering, and data visualization*, Journal of Machine Learning Research, 1 (2000), pp. 49–75.

[14] D. Heckerman, D. Geiger, and D. Chickering, *Learning Bayesian networks: The combination of knowledge and statistical data*, Machine Learning, 20 (1995), pp. 197–243.

[15] W. Kent, *A simple guide to five normal forms in relational database theory*, Commun. ACM, 26 (1983), pp. 120–125.

[16] K. Kersting and L. de Raedt, *Bayesian logic programming: Theory and tool*, in Introduction to Statistical Relational Learning [11], ch. 10, pp. 291–318.

[17] H. Khosravi, O. Schulte, and B. Bina, *Virtual joins with nonexistent links*, 19th Conference on Inductive Logic Programming (ILP), 2009. URL = `http://www.cs.kuleuven.be/~dtai/ilp-mlg-srl/papers/ILP09-39.pdf`.

[18] H. Khosravi, O. Schulte, T. Man, X. Xu, and B. Bina, *Structure learning for Markov logic networks with many descriptive attributes*, in Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)., 2010, pp. 487–493.

[19] A. C. Klug, *Equivalence of relational algebra and relational calculus query languages having aggregate functions*, J. ACM, 29 (1982), pp. 699–717.

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

[21] ——, *Learning markov logic network structure via hypergraph lifting*, in ICML, A. P. Danyluk, L. Bottou, and M. L. Littman, eds., ACM, 2009, pp. 64–71.

[22] D. Koller and A. Pfeffer, *Learning probabilities for noisy first-order rules*, in IJCAI, 1997, pp. 1316–1323.

[23] G. Lacerda, P. Spirtes, J. Ramsey, and P. O. Hoyer, *Discovering cyclic causal models by independent components analysis*, in UAI, D. A. McAllester and P. Myllymäki, eds., AUAI Press, 2008, pp. 366–374.

[24] P. Liang and M. I. Jordan, *An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators*, in ICML '08: Proceedings of the 25th international conference on Machine learning, New York, NY, USA, 2008, ACM, pp. 584–591.

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

[26] A. W. Moore and M. S. Lee, *Cached sufficient statistics for efficient machine learning with large datasets*, J. Artif. Intell. Res. (JAIR), 8 (1998), pp. 67–91.

[27] R. E. Neapolitan, *Learning Bayesian Networks*, Pearson Education, 2004.

[28] J. Neville and D. Jensen, *Relational dependency networks*, Journal of Machine Learning Research, 8 (2007), pp. 653–692.

[29] L. Ngo and P. Haddawy, *Answering queries from context-sensitive probabilistic knowledge bases*, Theor. Comput. Sci., 171 (1997), pp. 147–177.

[30] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.

[31] M. Pihlaja, M. Gutmann, and A. Hyvrinen, *A family of computationally efficient and simple estimators for unnormalized statistical models*, in Uncertainty in Artificial Intelligence (UAI), 2010.

[32] D. Poole, *First-order probabilistic inference*, in IJCAI, G. Gottlob and T. Walsh, eds., Morgan Kaufmann, 2003, pp. 985–991.

[33] R. Raina, Y. Shen, A. Y. Ng, and A. McCallum, *Classification with hybrid generative/discriminative models*, in NIPS, S. Thrun, L. K. Saul, and B. Schölkopf, eds., MIT Press, 2003.

[34] T. Richardson, *A discovery algorithm for directed cyclic graphs*, in UAI, E. Horvitz and F. V. Jensen, eds., Morgan Kaufmann, 1996, pp. 454–461.

[35] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, MIT Press, 2000.

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

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

[38] M. Y. Vardi, *On the complexity of bounded-variable queries*, in PODS, ACM Press, 1995, pp. 266–276.

[39] M. Wellman, J. Breese, and R. Goldman, *From knowledge bases to decision models*, Knowledge Engineering Review, 7 (1992), p. 35?53.

[40] R. Xiang and J. Neville, *Pseudolikelihood em for within-network relational learning*, in ICDM, IEEE Computer Society, 2008, pp. 1103–1108.

[41] Z. Xu, K. Kersting, and V. Tresp, *Multi-relational learning with gaussian processes*, in IJCAI, C. Boutilier, ed., 2009, pp. 1309–1314.

[42] X. Yin, J. Han, J. Yang, and P. S. Yu, *Crossmine: Efficient classification across multiple database relations*, in Constraint-Based Mining and Inductive Databases, 2004, pp. 172–195.