Learning Bayes Nets with Link Uncertainty for Relational Data Sets

Oliver Schulte and Zhensong Qian

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

Abstract

Many if not most big data sets are maintained in relational databases. Bayes net learning has the potential to discover knowledge about correlations among link types and node attributes in big relational data. The current state of the art algorithm for learning relational Bayes nets captures only correlations among entity attributes *given* the existence of links among entities. The models described in this paper capture a wider class of correlations that involve uncertainty about the link structure. A key challenge for relational learning that scales with data size is to compute event counts in a relational database (sufficient statistics), especially when these involve negated relationships. We describe how the fast Möbius transform provides a scalable solution for this problem.

1 Introduction

Many if not most big data sets are maintained in relational databases. Such datasets represent complex heterogenous networks [**?**]. Scalable link analysis for heterogenous networks with multiple link types is a challenging problem in network science. We describe a method for learning a Bayes net that captures simultaneously correlations between link types, link features, and attributes of nodes.

Motivation Building a Bayes net model is useful for big data analysis because such models provide a compact summary of the statistical relationships in the data. The model supports both descriptive and predictive analytics. Correlations are presented to the user in a graphical way, and queries about probabilistic relationships can be answered quickly using Bayes net inference rather than via database queries run against a large dataset. Applications supported by Bayes net learning include the following:

Knowledge Discovery Dependencies provide valuable insights. For instance, a web search manager may wish to know whether if user searches for a video in Youtube for a product, they are also likely to search for it on the web.

Probabilistic queries Once the model has been built, well-researched Bayes net methods can be used for probabilistic inference and "what-if" queries. For instance, a university administrator what wish to know how raising the average SAT test score of applicants would affect the attrition rate of students.

Database Query Optimization The Bayes net model can also be used to estimate relational statistics, the frequency with which statistical patterns occur in the database [**?**]. This kind of statistical model can be applied for database query optimization [**?**].

Previous work on learning Bayes nets for relational data was restricted to correlations among attributes given the existence of links [**?**]. The larger class of correlations examined in our new algorithms includes two additional kinds:

1. Dependencies between two different types of links.
2. Dependencies among node attributes given the *absence* of a link between the node.

Approach We consider three approaches to multiple link analysis with Bayes nets.

Flat Search Apply a standard Bayes net learner to a single large join table. This table is formed as follows: (1) take the cross product of entity tables. (An entity tables lists the set of nodes of a given type.) (2) For each tuple of entities, add a relationship indicator whose value "true" or "false" indicates whether a certain relationship holds among the entities.

Hierarchical Search Conducts bottom-up search through the lattice of table joins hierarchically. Dependencies (Bayes net edges) discovered on smaller joins are propagated to larger joins. The different table joins include information about the presence of absence of relationships as in the flat search above. This is an extension of the current state of the art Bayes net learning algorithm for relational data [**?**].

Evaluation. We compare the learned models using standard scores (e.g., Bayes Information Criterion, log-likelihood). These results indicate that both flat search and hierarchical search are effective at finding correlations among link types. Flat search can on some datasets achieve a higher score by exploiting attribute correlations that depend on the absence of relationships. Structure learning time results indicate that hierarchical search is substantially more scaleable.

Contributions

1. To our knowledge this is the first application of Bayes net learning to modelling correlations among different types of links.
2. Extension of a lattice search strategy for link type modelling, with a comparison to a flat search join approach.
3. Using the lattice Möbius transform to make the computation of the empirical frequencies of the relations imposed by negated relational links tractable.

Paper Organization We describe Bayes net models for relational data (Poole's Parametrized Bayes Nets). Then we present the learning algorithms, first flat search then hierarchical search. We compare different model search strategies on four databases from different domains. The last part of this paper turns from statistical to computation issues. We propose the Möbius transform to efficiently compute sufficient database statistics involving any number of negated relationships.

2 Related Work

To our knowledge, there are no implementations of structure learning algorithms for directed graphical models that consider correlations among different link types, let alone together with node attributes. Such implementations exist, however, for other types of graphical models, specifically Markov random fields (undirected models) [**?**] and dependency networks (directed edges with cycles allowed) [**?**]. Structure learning programs for Markov random fields are provided by Alchemy [**?**] and Khot et al [**?**]. Khot et al. use boosting to provide a state-of-the-art dependency network learner. None of these programs are able to return a result on half of our datasets because they are too large. For space reasons we restrict the scope of this paper to directed graphical models and do not go further into undirected model. For an extensive comparison of the learn-and-join Bayes net learning algorithm with Alchemy please see [**?**].

3 Background and Notation

Poole introduced the Parametrized Bayes net (PBN) formalism that combines Bayes nets with logical syntax for expressing relational concepts [**?**]. We adopt the PBN formalism, following Poole's presentation.

3.1 Bayes Nets for Relational Data

A **population** is a set of individuals. Individuals are denoted by lower case expressions (e.g., *bob*). A **population variable** is capitalized. A **functor** represents a mapping $f : \mathcal{P}_1, \ldots, \mathcal{P}_a \rightarrow V_f$ where f is the name of the functor, and V_f is the output type or **range** of the functor. In this paper we consider only functors with a finite range, disjoint from all populations. If $V_f = \{T, F\}$, the functor f is a (Boolean) **predicate**. A predicate with more than one argument is called a **relationship**; other functors are called **attributes**. We use uppercase for predicates and lowercase for other functors.

A **Bayes Net (BN)** is a directed acyclic graph (DAG) whose nodes comprise a set of random variables and conditional probability parameters. For each assignment of values to the nodes, the joint probability is specified by the product of the conditional probabilities, $P(child|parent\_values)$. A **Parametrized random variable** is of the form $f(X_1, \ldots, X_a)$, where the populations associated with the variables are of the appropriate type for the functor. A **Parametrized Bayes Net** (PBN) is a Bayes net whose nodes are Parametrized random variables [**?**]. If a Parametrized random variable appears in a Bayes net, we often refer to it simply as a node.

3.2 Databases and Table Joins

We begin with a standard **relational schema** containing a set of tables, each with key fields, descriptive attributes, and possibly foreign key pointers. A **database instance** specifies the tuples contained in the tables of a given database schema. We assume that tables in the relational schema can be divided into *entity tables* and *relationship tables*. This is the case whenever a relational schema is derived from an entity-relationship model (ER model) [**?**, Ch.2.2]. In our university example, there are two entity tables: a *Student* table and a *Course* table. There is one relationship table *Registered* with foreign key pointers to the *Student* and *Course* tables whose tuples indicate which students have registered in which courses.

The functor formalism is rich enough to represent the constraints of an ER schema by the following translation: Entity sets correspond to types, descriptive attributes to functions, relationship tables to predicates, and foreign key constraints to type constraints on the arguments of relationship predicates.

Table **??** shows a relational schema for a database related to a university. Figure **??** displays a small database instance for this schema together with a Parametrized Bayes Net (omitting the *Teaches* relationship for simplicity.)

The **natural table join**, or simply join, of two or more tables contains the rows in the Cartesian products of the tables whose values match on common fields. In logical terms, a join corresponds to a conjunction [**?**].

(a)

Student		
s-id	Intelligence	Ranking
Jack	3	1
Kim	2	1
Paul	1	2

(b)

Registration			
s-id	C.id	Grade	Satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Kim	103	A	1
Paul	101	B	1
Paul	102	C	2

(c)

Course					
c-id	Prof	Rating	Difficulty	T-a-prof	P-prof
101	Oliver	3	1	1	3
102	David	2	2	1	2
103	Oliver	3	2	1	3

(d)

attribute-relation table								
Intelligence	Rank	Rating	Diff	P-prof	t-a-prof	Reg(S,C)	R.Grade	R.Sat
3	1	3	1	3	1	T	A	1
3	1	2	2	2	1	T	B	2
3	1	3	2	3	1	F	⊥	⊥
2	1	3	1	3	1	F	⊥	⊥
2	1	2	2	2	1	T	A	1
2	1	3	2	3	1	T	A	1
1	2	3	1	3	1	T	B	1
1	2	2	2	2	1	T	C	2
1	2	3	2	3	1	F	⊥	⊥

(e)

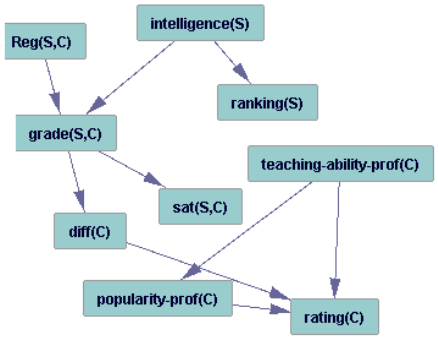

Figure 1: Database Table Instances: (a) *Student*, (b) *Registered* (c) *Course*. To simplify, we added the information about professors to the courses that they teach. (d) The attribute-relation table *Registered*$^+$ derived from *Registered*, which lists for each pair of entities their descriptive attributes, whether they are linked by *Registered*, and the relationships of a link if it exists. (e) A Parametrized Bayes Net for the university schema.

Student(*student_id*, *intelligence*, *ranking*)
Course(*course_id*, *difficulty*, *rating*)
Professor (*professor_id*, *teaching_ability*, *popularity*)
Registered (*student_id*, *Course_id*, *grade*, *satisfaction*)
Teaches(*professor_id*, *course_id*)

Table 1: A relational schema for a university domain. Key fields are underlined. An instance for this schema is given in Figure **??**.

4 Bayes Net Learning With Link Correlation Analysis

We outline the two methods we compare in this paper, flat search and hierarchical search.

4.1 Flat Search

The basic idea for flat search is to apply a standard single-table Bayes net learner to a single large join table. To learn correlations between link types, we need to provide the Bayes net with data about when links are present *and* when they are absent. To accomplish this, we add to each relationship table a **link indicator column**. This columns contains 1 if the link is present between two entities (specified in the key fields), and 0 if the link is absent. We add rows for all pairs of entities of the right type for the link, and enter 0 or 1 in the link indicator column depending on whether a link exists or not. We refer to relationship tables with a link indicator column as **extended** tables. Extended tables are readily computed using SQL queries. If we omit the entity Ids from an extended table, we obtain the **attribute-relation** table that lists all attributes for the entities involved and whether a relationship exists. If the attribute-relation table is derived from a relationship R, we refer to it as R^+.

The attribute-relation table is readily defined for a set of relationships: take the cross-product of all populations involved, and add a link indicator column for each relationship in the set. For instance, if we wanted to examine correlations that involve both the *Registered* and the *Teaches* relationships, we would form the cross-product of the entity types *Student*, *Course*, *Professor* and build an attribute-relation table that contains two link indicator columns *Registered*(S, C) and *Teaches*(P, C). The **full join** is the attribute-relation table for all relationships in the database.

The **flat search Bayes net learner** takes a standard Bayes net learner and applies it to the full join table to obtain a single Parametrized Bayes net. The results of [**?**] can be used to provide a theoretical justification for this procedure; we outline two key points. (1) The full join table correctly represents the *sufficient statistics* of the database: using the full join table to compute the frequency of a joint value assignment for Parametrized Random Variables is equivalent to the frequency with which this assignment holds in the database. (2) Maximizing a standard single-table likelihood score from the full join table is equivalent to maximizing the *random selection pseudo likelihood*. The random selection pseudo log-likelihood is the expected log-likelihood assigned by a Parametrized Bayes net when we randomly select individuals from each population and instantiate the Bayes net with attribute values and relationships associated with the selected individuals.

4.2 Hierarchical Search

Khosravi *et al.* present the learn-and-join structure learning algorithm. The algorithm upgrades a single-table Bayes net learner for relational learning. We describe the fundamental ideas of the algorithm; for further details. The key idea is to build a Bayes net for the entire

database by level-wise search through the *table join lattice*. The user chooses a single-table Bayes net 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 $s, s+1, \ldots$, with the constraint that the absence or presence of learned edges from smaller join tables is propagated to larger join tables. These constraints are implemented by keeping a global cache of forbidden and required edges. Algorithm **??** provides pseudocode for the previous learn-and-join algorithm (LAJ) [**?**].

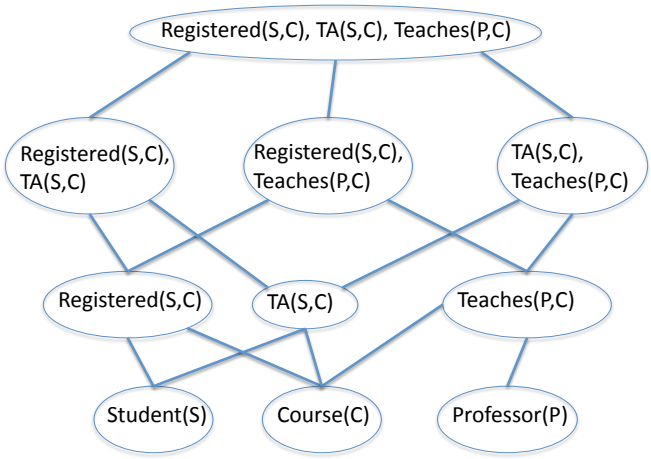

Figure 2: A lattice of relationship sets for the University schema of Table **??**. Links from entity tables to relationship tables correspond to foreign key pointers.

To extend the learn-and-join algorithm for link analysis, we replace the natural join in line 7 by the extended join (more precisely, by the attribute-relation tables derived from the extended join). The natural join contains only tuples that appear in all relationship tables. Compared to the extended join, this corresponds to considering only rows where the link indicator columns have the value T. When the propositional Bayes net learner is applied to such a table, the link indicator variable appears like a constant. Therefore the BN learner cannot find any correlations between the link indicator variable and other nodes, nor can it find correlations among attributes conditional on the link indicator variable being F. Thus the previous LAJ algorithm finds only correlations between entity attributes conditional on the existence of a relationship. In sum, hierarchical search with link correlations can be described as follows.

1. Run the previous LAJ algorithm (Algorithm **??**) using natural joins.
2. Starting with the constraints from step 1, extend them with the LAJ algorithm where extended joins replace natural joins. That is, for each relationship set shown in the lattice of Figure **??**, apply the single-table Bayes net learner to the extended join for the relationship set.

Algorithm 1 Pseudocode for previous Learn-and-Join Structure Learning for Lattice Search.

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

1: Add descriptive attributes of all entity and relationship tables as variables to G. Add a boolean indicator for each relationship table to G.
2: Econstraints = $\emptyset$ [Required and Forbidden edges]
3: **for** m=1 to e **do**
4: Econstraints += Get-Constraints(PBN(E_m , $\emptyset$))
5: **end for**
6: **for** m=1 to r **do**
7: N_m := natural join of R_m and entity tables linked to R_m
8: Econstraints += Get-Constraints(PBN(N_m, Econstraints))
9: **end for**
10: **for all** N_i and N_j with a foreign key in common **do**
11: K_{ij} := join of N_i and N_j
12: Econstraints += Get-Constraints(PBN(K_{ij}, Econstraints))
13: **end for**
14: **return** Bayes Net defined by Econstraints.

5 Evaluation

All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. Our code and datasets are available on the world-wide web [**?**]. We made use of the following single table Bayes Net search implementation: GES search [**?**] with the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; [**?**]).

Methods Compared We compared the following methods.

LAJ The previous LAJ method without link correlations (Algorithm **??**).

LAJ+ The new LAJ method that has the potential to find link correlations (Algorithm **??** with the full join instead of natural join).

Flat Applies the single-table Bayes net learner to the full table join (extended join with all relationship sets in the database).

Performance Metrics We report learning time, log-likelihood, Bayes Information Criterion (BIC), and the Akaike Information Criterion (AIC). We write

$$L(\hat{G}, \mathbf{d})$$

Dataset	#tuples
University	662
Movielens	1585385
Mutagenesis	1815488
Hepatitis	2965919
Small-Hepatitis	19827

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

for the log-likelihood score, where where $\hat{G}$ is the BN G with its parameters instantiated to be the maximum likelihood estimates given the dataset $\mathbf{d}$, and the quantity $L(\hat{G}, \mathbf{d})$ is the log-likelihood of $\hat{G}$ on $\mathbf{d}$.

The BIC score is defined as follows [**?**; **?**]

$$BIC(G, \mathbf{d}) = L(\hat{G}, \mathbf{d}) - par(G)/2 \times ln(m)$$

where the data table size is denoted by m, and $par(G)$ is the number of free parameters in the structure G. The AIC score is given by

$$AIC(G, \mathbf{d}) = L(\hat{G}, \mathbf{d}) - par(G).$$

BIC and AIC are standard scores for Bayes nets [**?**]. AIC is asympotically equivalent to selection by cross-validation, so we may view it as a closed-form approximation to cross-validation, which is computationally demanding for relational datasets.

Datasets We used one synthetic and three benchmark real-world databases, with the modifications described by Schulte and Khosravi [**?**]. See that article for more details.

University Database. We manually created a small dataset, based on the schema given in Table **??**. The dataset is small and is used as a tested for the correctness of our algorithms.

MovieLens Database. A dataset from the UC Irvine machine learning repository. The data are organized in 3 tables (2 entity tables, 1 relationship table, and 7 descriptive attributes).

Mutagenesis Database. A dataset widely used in ILP research. It contains two entity tables and two relationships.

Hepatitis Database. A modified version of the PKDD'02 Discovery Challenge database. The data are organized in 7 tables (4 entity tables, 3 relationship tables and 16 descriptive attributes). In order to make the learning feasible, we under sampled Hepatitis database to keep the ratio of positive and negative link indicator equal to one.

5.1 Results

Learning Times Table **??** provides the model search time for each of the link analysis methods. This does not include the time for computing table joins since this is essentially the same for all methods (the cost of the full table join). On the smaller and simpler datasets, all search strategies are fast, but on the medium-size and more complex datasets (Hepatitis, MovieLens), hierarchical search is much faster due to its use of constraints. Adding prior knowledge as constraints could speed the structure learning substantially.

Dataset	Flat	LAJ+	LAJ
University	1.916	1.183	0.291
Movielens	38.767	18.204	1.769
Mutagenesis	3.231	3.448	0.982
Small-Hepatitis	9429.884	8.949	10.617

Table 3: Model Structure Learning Time in seconds.

University	BIC	AIC	log-likelihood	# Parameter
Flat	-17638.27	-12496.72	-10702.72	1767
LAJ+	-13495.34	-11540.75	-10858.75	655
LAJ	-13043.17	-11469.75	-10920.75	522

MovieLens	BIC	AIC	log-likelihood	# Parameter
Flat	-4912286.87	-4911176.01	-4910995.01	169
LAJ+	-4911339.74	-4910320.94	-4910154.94	154
LAJ	-4911339.74	-4910320.94	-4910154.94	154

Mutagenesis	BIC	AIC	log-likelihood	# Parameter
Flat	-21844.67	-17481.03	-16155.03	1289
LAJ+	-47185.43	-28480.33	-22796.33	5647
LAJ	-30534.26	-25890.89	-24479.89	1374

Hepatitis	BIC	AIC	log-likelihood	# Parameter
Flat	-7334391.72	-1667015.81	-301600.81	1365357
LAJ+	-457594.18	-447740.51	-445366.51	2316
LAJ	-461802.76	-452306.05	-450018.05	2230

Table 4: Performance of different Searching Algorithms by dataset.

Statistical Scores As expected, adding edges between link nodes improves the statistical data fit: the link analysis methods LAJ+ and Flat perform better than the learn-and-join baseline in terms of log-likelihood on all datasets shown in table **??**. On the small synthetic dataset University, flat search appears to overfit whereas the hierarchical search methods are very close. On the medium-sized dataset MovieLens, which has a simple structure, all three methods score similarly. Hierarchical search finds no new edges involving the single link indicator node (i.e., LAJ and LAJ+ return the same model).

The most complex dataset, Hepatitis, is a challenge for flat search, which seems to overfit severely with a huge number of parameters that result in a model selection score that is an order of magnitude worse than for hierarchical search. Because of the complex structure of the Hepatitis schema, the hierarchical constraints appear to be effective in combating overfitting.

The situation is reversed on the Mutagenesis dataset where flat search does well: compared to attribute-only search, it manages to fit the data better with a less

complex model. Hiearchical search performs very poorly compared to flat search (lower likelihood yet many more parameters in the model). Investigation of the models shows that the reason for this phenomenon is a special property of the Mutagenesis dataset: whereas generally relationships are sparse—very few pairs of entities are actually linked—in Mutagenesis most entities whose type allows a link are linked. As a result, we find strong correlations between attributes conditional on *the absence of relationships*. The LAJ+ algorithm is constrained so that it cannot add Bayes net edges between attribute nodes at its second stage, when absent relationships are considered. As a result, it can represent attribute correlations conditional on the absence of relationships only indirectly through edges that involve link indicators. A solution to this problem would be to add a phase to the search so that we first learn edges between attributes first conditional on both the existence of relationships, then conditional on their nonexistence. The last phase then would consider edges that involve relationship nodes. We expect that with this change, hierarchical search would be competitive with flat search on the Mutagenesis dataset as well.

6 Computing Relational Contingency Tables

The learning algorithms described in this paper rely on the availability of the extended relational tables R^+ (see Figure **??**). Our first naive implementation constructs this tables using standard joins. While this was sufficient for our experiments, the cross-products carry a quadratic costs for binary relations, and therefore do not scale to large datasets. Moreover, the hierarchical search requires joins of the extended tables. In this section we describe a "virtual join" algorithm that computes the required data tables without the quadratic cost of materializing a cross-product.

Our starting point is the observation that a statistical learning algorithm like a Bayes net learner does not require an enumeration of individuals tuples, but only *sufficient statistics* [**?**; **?**]. These can be represented in *contingency tables* as follows [**?**]. Consider a fixed list of relationship nodes $R_1, R_2, \ldots, R_m$, and attribute nodes $f_1, \ldots, f_j$. A **query** is a set of ($node = value$) pairs where each value is of a valid type for the node. The **result set** of a query in a database $\mathcal{D}$ is the set of instantiations of the population variables such that the query evaluates as true in $\mathcal{D}$. For example, in the database of Figure **??** the result set for the query
$intelligence(S) = 2$, $rank(S) = 1$, $rating(C) = 3$, $Diff(C) = 1$, $Registered(S, C) = F$
is the singleton $\{\langle kim, 101\rangle\}$. The **count** of a query is the cardinality of its result set. Each subset of nodes $\mathbf{V} = v_1, \ldots, v_n$ has an associated **contingency table** denotes by $CT(\mathbf{V})$. This is a table with a row for each of the possible assignments of values to the nodes in $\mathbf{V}$. The row corresponding to $V_1 = v_1, \ldots, V_n = v_n$ records the count of the corresponding query. Figure fig:ct shows a contingency table.

	CT(Reg(S,C),EAtts(Reg(S,C)), RAtts(Reg(S,C)))						
	EAtts(Reg(S,C))					RAtts(Reg(S,C))	
count	Int(S)	Rank(S)	Rating(C)	Diff(C)	Reg(S,C)	grade(S,C)	satisfaction(S,C)
2	3	1	3	1	T	A	1
2	3	1	2	2	T	B	2
1	2	1	2	2	T	A	1
2	2	1	3	2	T	A	1
1	1	2	3	1	T	B	1
1	1	2	2	2	T	C	2
4	3	1	3	2	F	n/a	n/a
1	2	1	3	1	F	n/a	n/a
2	1	2	3	2	F	n/a	n/a

Figure 3: An example contingency table for the attribute-relation table of Figure **??**, where for illustration we have added counts for another student like Jack and another course like 103.

A **conditional contingency table**, written $ct(V_1, \ldots, V_k | V_{k+1} = v_{k+1}, \ldots, V_{k+m} = v_{k+m})$ is the contingency table whose column headers are $V_1, \ldots, V_k$ and whose counts are defined by subset of instantiations that match the conditions to the right of the — symbol.

Our flat search algorithm can be implemented by computing the contingency table for all functor nodes, then presenting it to a Bayes net learner. The hierarchical search algorithms can be implemented by computing the contingency tables for each relationship chain in the lattice (cf. Figure **??**). For a relationship set, the nodes in the contingency table comprise (1) the descriptive attributes of the entities involved in the relationship set, (2) the descriptive attributes of the relationships, and (3) a Boolean relationship node for each member of the set. This section describes a method for computing these contingency tables level-wise.

So long as a database probability involves only positive relationships, and can be carried out by regular table joins or optimized virtual joins [**?**]. Computing joint probabilities for a family containing one or more negative relationships is harder. A naive approach would explicitly construct new data tables that enumerate tuples of objects that are *not* related (see Figure **??**). However, the number of unrelated tuples is too large to make this scalable (think about the number of user pairs who are *not* friends on Facebook). In their work on learning Probabilistic Relational Models with existence uncertainty, Getoor et al. provided a subtraction method for the special case of estimating counts with only a single negated relationship [**?**, Sec.5.8.4.2]. They did not treat contigency tables with multiple negated relationships, which we consider next.

6.1 Level-Wise Computation of Contingency Tables: The Subtraction Method

We first introduce some notation. Let R be a relationship node and let $\mathbf{R}$ be a set of relationship nodes.

- $1Nodes(R)$ denotes the set of entity attribute nodes

for the population variables involved in the relationship R.

- *1Nodes*($\mathbf{R}$) is the union of the entity attributes for each relationship $R \in \mathbf{R}$.
- *2Nodes*(R) denotes the set of relationship attribute nodes for the population variables involved in the relationship R.
- *2Nodes*($\mathbf{R}$) is the union of the relationship attributes for each relationship $R \in \mathbf{R}$.
- *ANodes*(R) is the set of both entity and relationship attributes for relationship R, similarly for *ANodes*($\mathbf{R}$).

We next define some basic operations on CT-tables. These are analogues to relational algebra operations on simple tables.

1. Let $ct_1(\mathbf{V}), ct_2(\mathbf{V})$ be two union-compatible contingency tables with the same column headers, and (hence) with the same rows, except for the count entries. The **table difference** $ct_1 - ct_2$ is a table with the same column headers, such that
$$\#_{[ct_1 - ct_2]}(\mathbf{V} = \mathbf{v}) = \#_{[ct_1]}(\mathbf{V} = \mathbf{v}) - \#_{[ct_1]}(\mathbf{V} = \mathbf{v}).$$
2. Let $ct_1(\mathbf{V}), ct_2(\mathbf{V}')$ be two disjoint contingency tables, i.e., $V \cap V' = \emptyset$. Then the **cross-product** $ct_1(\mathbf{V}) x ct_2(\mathbf{V}')$ is a table with the column headers $\mathbf{V} \cup \mathbf{V}'$ whose rows form the cross-product of the rows in ct_1 and in ct_2, and whose count satisfy
$$\#_{[ct_1 \times ct_2]}(\mathbf{V} = \mathbf{v}, \mathbf{V}' = \mathbf{v}') = \#_{[ct_1]}(\mathbf{V} = \mathbf{v}) x \#_{[ct_1]}(\mathbf{V} = \mathbf{v}).$$
3. Let $\mathbf{V}'$ be a proper subset of columns $\mathbf{V}$ for a contingency table $ct(\mathbf{V})$. Then the **marginal** table $margin(ct, \mathbf{V}')$ is a table whose headers are $\mathbf{V} - \mathbf{V}'$, such that the count $\#_{margin(ct,\mathbf{V}')}(\mathbf{V} - \mathbf{V}' = \mathbf{v}^*$ is the sum of the counts in $ct(\mathbf{V})$ that satisfy $(\mathbf{V} - \mathbf{V}') = \mathbf{v}^*$.

We are now ready to state some contingency algebra equivalences that allows us to compute counts for rows with negative relations from rows with positive relations.

Let R be a relationship node and $\mathbf{R}$ be a set of relationship nodes not containing R. Then the following equations hold.

$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = F) = ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = *) - ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2No}$$
$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = *) = ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = T) \times \prod_{1Node(R) \in 1Nodes(R)} |1Node(R)^{\mathbf{R}}|$$
$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = T) = \mathit{margin}(ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R), \mathit{2Nodes}(R) | R = T), \mathit{2No}$$

The first equation is important because it shows how counts with a false relationship can be computed from counts with the relationship unspecified, and with the relationship value true. The second equation is important because it shows that counts with the relationship unspecified can be computed from ct tables that omit the relationship, multiplied by tables for entity attributes that are involved in the relationship (and that do not already appear in other relationships). The third equation is important because it shows how a table without the descriptive attributes of R can be computed from a complete CT table with R by marginalizing out the descriptive attributes of R.

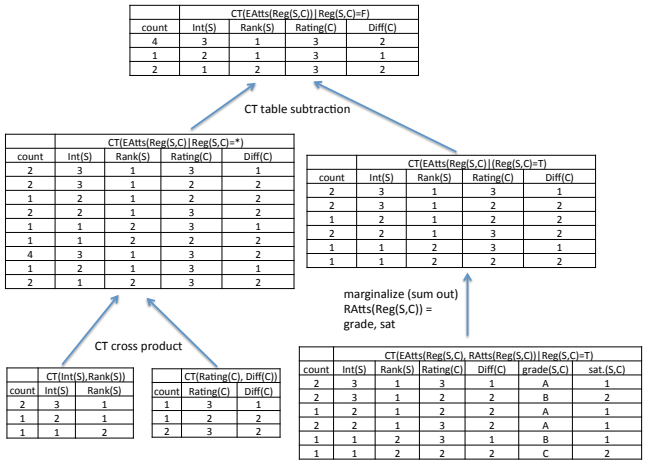

Figure 4: Examples of the table algebra operations and equations.

Figure reffig:table-equation illustrates the equations.

For a given chain of relationship nodes $\mathbf{R} = R_1, \ldots, R_m$, the equations can be applied as follows.

1. Find the CT table for $R_1 = T, \ldots, R_m = T$.
2. For $j = 1, \ldots, m-1$, apply the proposition to find $ct(\mathit{ANodes}(\mathbf{R}), R_{j+1}, \ldots, R_m | R_1 = T, \ldots, R_j = T$.
3. Apply the proposition one last time to find $ct(\mathit{ANodes}(\mathbf{R}), \mathbf{R}$.

7 Conclusion

We described different methods for extending relational Bayes net learning to correlations involving links. Statistical measures indicate that Bayes net methods succeed in finding relevant correlations. There is a trade-off between statistical power and computational feasibility (full table search vs constrained search). Hierarchical search often does well on both dimensions, but needs to be extended to handle correlations conditional on the absence of relationships.

Most of the computation time is taken up by forming table joins, whose size is the cross product (Cartesian product) of entity tables. These table joins provide the sufficient statistics required in model selection. For Bayes net learning to be possible for big data, computing sufficient statistics needs to be feasible for cross product sizes in the millions or more. We described a possible solution: virtual join methods that compute sufficient statistics without materializing cross products such as the fast Möbius transform, and tuple ID propagation [**?**].

References

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

[2] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.

[3] Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. *ACM SIGMOD Record*, 30(2):461–472, 2001.

[4] H. Khosravi, T. Man, J. Hu, E. Gao, and O. Schulte. Learn and join algorithm code. URL = `http://www.cs.sfu.ca/~oschulte/jbn/`.

[5] Tushar Khot, Jude Shavlik, and Sriraam Natarajan. Boostr, 2013. URL = `http://pages.cs.wisc.edu/~tushar/Boostr/`.

[6] Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.

[7] David Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.

[8] Oliver Schulte. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.

[9] Oliver Schulte. Challenge paper: Marginal probabilities for instances and classes. ICML-SRL Workshop on Statistical Relational Learning., June 2012.

[10] Oliver Schulte and Hassan Khosravi. Learning graphical models for relational data via lattice search. *Machine Learning*, 88(3):331–368, 2012.

[11] Oliver Schulte, Hassan Khosravi, Arthur Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with bayes nets. In *Inductive Logic Programming (ILP)*, 2012.

[12] Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*, volume 3. Morgan & Claypool Publishers, 2012.

[13] The Tetrad Group. The Tetrad project, 2008. http://www.phil.cmu.edu/projects/tetrad/.

[14] J. D. Ullman. *Principles of database systems*. W. H. Freeman & Co., 2 edition, 1982.

[15] Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S. Yu. Crossmine: Efficient classification across multiple database relations. In *ICDE*, pages 399–410. IEEE Computer Society, 2004.