BayesBase: Managing Relational Database Schemata for Learning Graphical Models

Zhensong Qian and Oliver Schulte

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

Abstract

We present an algorithm for learning correlations among link types and node attributes in relational data that represent complex networks. The link correlations are represented in a Bayes net structure. This provides a succinct graphical way to display relational statistical patterns and support powerful probabilistic inferences. 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. Our base line method learns a Bayes net from join tables directly. This is a statistically powerful procedure that finds many correlations, but does not scale well to larger datasets. We compare join table search with a hierarchical search strategy.

1 Introduction

Scalable link analysis for relational data 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. Such a Bayes net provides a succinct graphical representation of complex statistical-relational patterns. A Bayes net model supports powerful probabilistic reasoning for answering "what-if" queries about the probabilities of uncertain outcomes conditional on observed events. 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]

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

Discovering such dependencies is useful for several applications.

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

Relevance Determination Once dependencies have been established, they can be used as a relevance filter for focusing further network analysis only on statistically significant associations. For example, the classification and clustering methods of Sun and Han [?] for heterogeneous networks assume that a set of "metapaths" have been found that connect link types that are associated with each other.

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 [?].

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

Flat Search Applies 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 table 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 the 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 or 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 [?].

[1]This research was supported by a Discovery Grant to Oliver Schulte from the Canadian Natural Sciences and Engineering Council. And Zhensong Qian was also supported by a grant from the China Scholarship Council.

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 scalable.

The main contribution of this paper is extending the current state-of-the-art Bayes net learner to model correlations among different types of links, with a comparison of a flat and a hierarchical search strategy.

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 the models on four databases from different domains.

2 Related Work

Approaches to structure learning for directed graphical models with link uncertainty have been previously described, such as [?]. However to our knowledge, no implementations of such structure learning algorithms for directed graphical models are available. Our system builds on the state-of-the-art Bayes net learner for relational data, whose code is available at [?]. Implementations exist 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 $f : \mathcal{P}_1, \ldots, \mathcal{P}_a \to V_f$ where f is the name of the functor, each $\mathcal{P}_i$ is a population, 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.

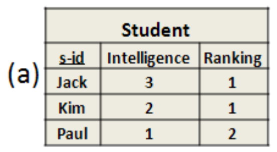
Student($\underline{student\_id}$, *intelligence*, *ranking*)
Course($\underline{course\_id}$, *difficulty*, *rating*)
Professor ($\underline{professor\_id}$, *teaching_ability*, *popularity*)
Registered ($\underline{student\_id, Course\_id}$, *grade*, *satisfaction*)
Teaches($\underline{professor\_id, course\_id}$)

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

3.2 Databases and Table Joins

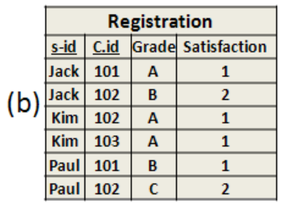
We begin with a standard **relational schema** containing a set of tables, each with key fields, descriptive attributes, and possibly foreign key pointers. A **database instance** specifies the tuples contained in the tables of a given database schema. We assume that tables in the relational schema can 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]. 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. Assuming an ER design, a relational structure can be visualized as a complex network [?, Ch.8.2.1]: individuals are nodes, attributes of individuals are node labels, relationships correspond to (hyper)edges, and attributes of relationships are edge labels. Conversely, a complex network can be represented using a relational database schema.

Table 1 shows a relational schema for a database related to a university. In this 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. Figure 1 displays a small database instance for this schema together with a Parametrized Bayes Net (omitting the *Teaches* relationship for simplicity.)



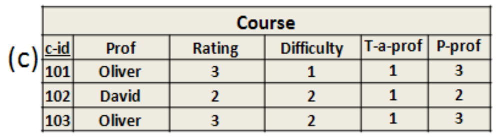
(a)

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



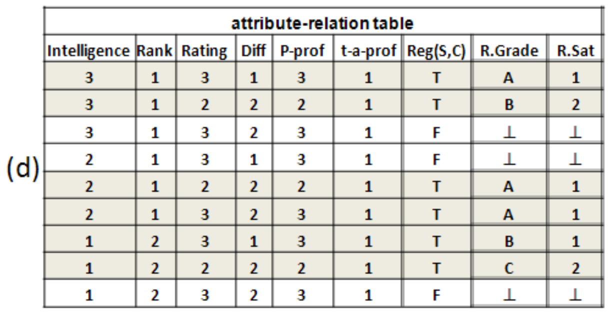
(b)

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	⊥	⊥

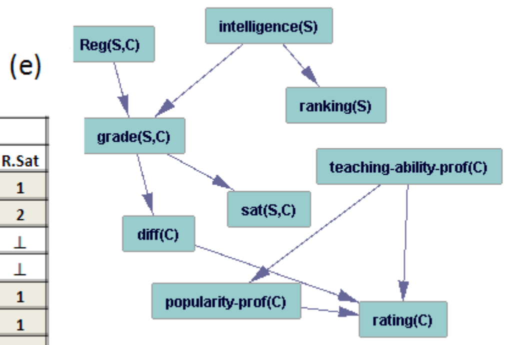


Figure 1: Database Table Instances: (a) *Student* (b) *Registered* (c) *Course*. To simplify, we added the information about professors to the courses that they teach. (d) The attribute-relation table *Registered*$^+$ derived from *Registered*, which lists for each pair of entities their descriptive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

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 [?].

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 propositional or 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 T if the link is present between two entities, and F if the link is absent. (The entities are specified in the primary key fields.) We add rows for all pairs of entities of the right type for the link, and enter T or F 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 (1) all attributes for the entities involved, (2) whether a relationship exists and (3) the attributes of the relationship if it 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 table** 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 please see [?]. 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 larger join tables inherit the absence or presence of learned edges from smaller join tables. These edge constraints are implemented by keeping a global cache of forbidden and required edges. Algorithm 1 provides pseudocode for the previous learn-and-join algorithm (LAJ) [?].

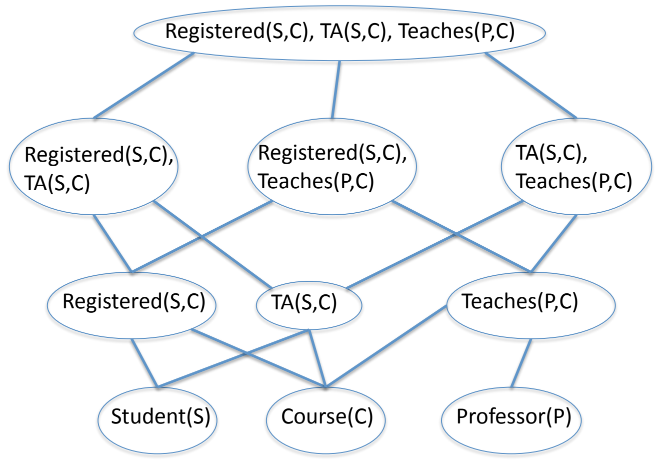


Figure 2: A lattice of relationship sets for the university schema of Table 1. Links from entity tables to relationship tables correspond to foreign key pointers.

To extend the learn-and-join algorithm for multiple 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 1) using natural joins.
2. Starting with the constraints from step 1, run the LAJ algorithm where extended joins replace natural joins. That is, for each relationship set shown in the lattice of Figure 2, 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. The LAJ 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.

Bayes Base Hierarchical Search (B.B.H.) The previous LAJ method without link correlations (Algorithm 1).

Flat Search The new LAJ method that has the potential to find link correlations (Algorithm 1 with the extended join tables instead of natural join tables).

Complete Graph Applies the single-table Bayes net learner to the full join table.

Disconnected Graph Applies the single-table Bayes net learner to the full join table.

To implement Flat Search and the LAJ+ algorithm efficiently, we apply the Fast Möbius Transform to compute tables of sufficient statistics that involve negated relationships. We discuss the details further in Section 7.

Performance Metrics Pseudo Relation Model Selection Scores [?] ?

We report learning time, log-likelihood, Bayes Information Criterion (BIC), and the Akaike Information Criterion (AIC). BIC and AIC are standard scores for Bayes nets [?], defined as follows. We write

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

for the log-likelihood score, 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

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

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.

5.1 Empirical Evaluation: Learning Times

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

5.2 Datasets

We describe the datasets in terms of their representation as databases with tables. The databases follow an Entity-Relationship (E-R) design [?]. An E-R schema can be translated into our function-based logical notation as follows: Entity sets correspond to populations, descriptive attributes to functions, relationship tables to predicates, and foreign key constraints to type constraints on the arguments of relationship predicates. We used one synthetic and 8 benchmark real-world databases from prior work [?].

need to be filled in Also, we may emphasize the difference of such datasets in terms of self-relationships and same type relationships.

University Database. We manually created a small dataset, based on the schema given in Table 1. The dataset is small and is used as a testbed for the correctness of our algorithms.

MovieLens Database. only use the 1M version?

This is a standard dataset from the UC Irvine machine learning repository. **really? I can NOT find it online.** user:6039; Movie: 3883; rating: 1000129 age, 7 bins based on the original MovieLens design

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

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

Financial This dataset is a modified version of the financial dataset from the PKDD 1999 cup. We adapted the database design to fit the ER model by following the modification from CrossMine [?] and Graph-NB [?]. The data are organized in 6 tables (3 entity tables, 3 relationship tables, 15 descriptive attributes).

need to be filled in more details

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

IMDB Database, combination?.

we based on the MovieLens 1M dataset which is a commonly-used rating dataset (www.grouplens.org), and adding more related attribute information about the actors, directors and movies from the Internet Movie Database (IMDB) (www.imdb.com, July 2013).

4 entity tables : actors (gender 2, quality 6: 98690),directors(quality 6,avg_revenue 5:2201),movies(year 4,country 4,runningtime 4:3832),users(age 7,gender 2,occuption 5 :6039) and 3 relation tables: u2base(rating 5: 996159), movies2actors(cast_num 4:138349), movies2directos(genre 9:4141)

need to be filled in

Mondial Database. A geography database, featuring one self-relationship, *Borders*, that indicates which countries border each other. This dataset contains data from multiple geographical web data sources. We follow the modification of She[?], and use a subset of the tables and disretized features: 2 entity tables, *Country*, *Economy*. The descriptive attributes of Country are continent, government, percentage, majority religion, population size. The descriptive attributes of Economy are inflation, gdp, service, agriculture, industry. A relationship table Econ-

Dataset	Relationships	Self Relationships	Same Type Relationships	#Tuples	#Attribute Columns
University	2	N	N	336	12
Movielens(1M)	1	N	N	1,010,051	7
Movielens(0.1M)	1	N	N	83,402	7
Mutagenesis	2	N	Y	24,326	11
Financial	3	N	N	225,932	15
Hepatitis	3	N	N	11,316	19
IMDB	3	N	N	1,251,038	17
Mondial	2	Y	N	870	18
UW-CSE	2	Y	N	612	14

Table 2: Real datasets characteristics, including size of datasets in total number of table tuples.

Dataset	Building Time	Sum(counts)	Tuples	Compress Ratio
University	1.68	2,280	351	6.5
Movielens(1M)	2.70	23,449,437	252	93,053.32
Movielens(0.1M)	0.62	1,582,762	239	6,622.44
Mutagenesis	1.67	905,205	1,631	555.00
Financial	1421.87	149,046,585,349,303	3,013,011	49,467,653.90
Hepatitis	3536.76	17,846,976,000	12,374,892	1,442.19
IMDB	7467.85	5,030,412,758,502,710	15,538,430	323,740,092.05
Mondial	1112.84	4,655,957	1,746,870	2.67
UW-CSE	3.84	10,201,488	2,828	3,607.32

Table 3: Contengency Table (C.T.) Description : Building Time in seconds.

omy_Country specifies which country has what type of economy. A self-relationship table Borders relates two countries.

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

5.3 Learning Times

Table 3 shows the data preprocessing time that the different methods require for table joins. This is the same for all methods, namely the cost of computing the full join table using the fast Möbius transform described in Section 7.

[fill in discussion]

Table 4 provides the model search time for each of the link analysis methods. 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.

6 Empirical Evaluation: Statistical Scores

As expected, adding edges between link nodes improves the statistical data fit: the link analysis methods LAJ+

Dataset	BBH3	Flat	Compl.	Discon.
University	1.523	1.486	0.186	0.135
Movielens(1M)	1.528	0.968	0.101	0.066
Movielens(0.1M)	1.178	0.986	0.083	0.065
Mutagenesis	1.780	1.869	0.115	0.096
Financial	96.308	1,241.077	0.203	0.080
Hepatitis	416.704	N.T.	0.272	0.134
IMDB	551.643	N.T.	0.286	0.220
Mondial	190.162	1,289.534	0.280	0.093
UW-CSE	2.896	2.367	0.181	0.100

Table 4: Model Structure Learning Time in seconds.

University	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-554.01	-99.50	-5.50	94
Flat	-3731.88	-668.20	-5.20	663
Complete	-386566.28	-50000.10	-5.10	49995
Disconnected	-121.44	-31.89	-8.89	23

Movielens(1M)	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-4927.84	-295.44	-3.44	292
Flat	-5168.08	-315.44	-3.44	312
Complete	-11461.86	-678.44	-3.44	675
Disconnected	-179.20	-19.31	-4.31	15

Movielens(0.1M)	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-1198.42	-87.10	-3.10	84
Flat	-1691.20	-123.10	-3.10	120
Complete	-4160.11	-294.09	-3.09	291
Disconnected	-120.88	-14.34	-3.34	11

Mutagenesis	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-9083.88	-726.96	-5.96	721
Flat	-9224.25	-1120.59	-5.59	1115
Complete	-5806905.08	-423374.59	-5.59	423369
Disconnected	-292.7	-43.91	-7.91	36

Financial	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-68562.16	-2443.74	-10.74	2433
Flat	-188908818.94	-7206670.64	-10.66	7206660
Complete	-12218648361.31	-374400009.12	-10.67	374399999
Disconnected	-469.21	-61.79	-12.79	49

Hepatitis	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-10364.93	-585.58	-16.58	569
Flat	N.T.	N.T.	N.T.	N.T.
Complete	N.T.	N.T.	N.T.	N.T.
Disconnected	-473.81	-76.30	-18.30	58

IMDB	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-2072673.77	-60070.39	-11.39	60059
Flat	N.T.	N.T.	N.T.	N.T.
Complete	N.T.	N.T.	N.T.	N.T.
Disconnected	-647.90	-66.01	-12.01	54

Mondial	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-3980.41	-357.20	-18.20	339
Flat	-8886625.98	-865255.07	-14.07	865241
Complete	N.T.	N.T.	N.T.	N.T.
Disconnected	-336.50	-74.98	-19.98	55

UW-CSE	Pseudo_BIC	Pseudo_AIC	Norm_log-likelihood	# Para.
BBH3	-3187.86	-248.1	-8.10	240
Flat	-42038.65	-3939.01	-6.01	3933
Complete	-356947710.61	-22118404.95	-6.02	22118399
Disconnected	-287.63	-55.24	-10.24	45

Table 5: Statistical Performance of different Searching Algorithms by dataset.

Zhensong: paramete learning not terminated, mysql, may show some realy SQL queries?

and Flat perform better than the learn-and-join baseline in terms of log-likelihood on all datasets shown in table 5, except for MovieLens where the Flat search has a lower likelihood. 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 hierarchy constraints appear to be effective in combating overfitting.

The situation is reversed on the Mutagenesis dataset where flat search does well: compared to previous LAJ algorithm, 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: The two relationships in the dataset, Bond and MoleAtm, involve the same descriptive attributes. The hierarchical search learns two separate Bayes nets for each relationship, then propagates both sets to the final result. However, the union of the two graphs may not be a compact model of the associations that hold in the entire database. A solution to this problem would be to add a final pruning phase where redundant edges can be eliminated. We expect that with this change, hierarchical search would be competitive with flat search on the Mutagenesis dataset as well.

7 Computing Relational Contingency Tables

The learning algorithms described in this paper rely on the availability of the extended relational tables R^+ (see Figure 1). 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 1 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 1, 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 2). 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 1). 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.

7.1 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.

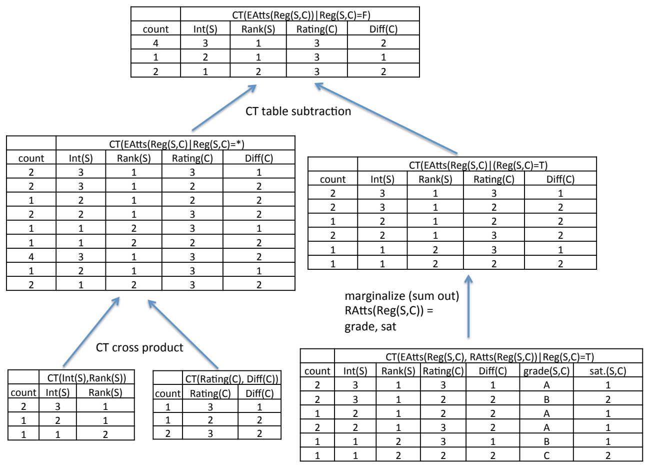
- $\mathit{1Nodes}(R)$ denotes the set of entity attribute nodes for the population variables involved in the relationship R.
- $\mathit{1Nodes}(\mathbf{R})$ is the union of the entity attributes for each relationship $R \in \mathbf{R}$.
- $\mathit{2Nodes}(R)$ denotes the set of relationship attribute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

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 $\mathit{margin}(ct, \mathbf{V}')$ is a table whose headers are $\mathbf{V} - \mathbf{V}'$, such that the count $\#_{\mathit{margin}(ct, \mathbf{V}')}(\mathbf{V} - \mathbf{V}' = \mathbf{v}^*$ 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(EAtts(Reg(S,C))|Reg(S,C)=F)			
count	Int(S)	Rank(S)	Rating(C)	Diff(C)
4	3	1	3	2
1	2	1	3	1
2	1	2	3	2

	CT(EAtts(Reg(S,C)|Reg(S,C)=*)			
count	Int(S)	Rank(S)	Rating(C)	Diff(C)
2	3	1	3	1
2	3	1	2	2
1	2	1	2	2
2	2	1	3	2
1	1	2	3	1
1	1	2	2	2
4	3	1	3	2
1	2	1	3	1
2	1	2	3	2

	CT(EAtts(Reg(S,C)|(Reg(S,C)=T)			
count	Int(S)	Rank(S)	Rating(C)	Diff(C)
2	3	1	3	1
2	3	1	2	2
1	2	1	2	2
2	2	1	3	2
1	1	2	3	1
1	1	2	2	2

	CT(Int(S),Rank(S))	
count	Int(S)	Rank(S)
2	3	1
1	2	1
1	1	2

	CT(Rating(C), Diff(C))	
count	Rating(C)	Diff(C)
1	3	1
1	2	2
2	3	2

	CT(EAtts(Reg(S,C), RAtts(Reg(S,C))|Reg(S,C)=T)					
count	Int(S)	Rank(S)	Rating(C)	Diff(C)	grade(S,C)	sat.(S,C)
2	3	1	3	1	A	1
2	3	1	2	2	B	2
1	2	1	2	2	A	1
2	2	1	3	2	A	1
1	1	2	3	1	B	1
1	1	2	2	2	C	2

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

$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = F) =$$
$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = *)$$
$$- \, ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = T)$$

$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = *) =$$
$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}) \times_{V \in \mathit{1Nodes}(R) - \mathit{1Nodes}(\mathbf{R})} ct\,V)$$

$$ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}), \mathit{2Nodes}(\mathbf{R}), \mathit{1Nodes}(R) | R = T) =$$
$$\mathit{margin}(ct(\mathbf{R}, \mathit{1Nodes}(\mathbf{R}),$$
$$\mathit{2Nodes}(\mathbf{R}, \mathit{1Nodes}(R), \mathit{2Nodes}(R) | R = T), \mathit{2Nodes}(R))\Big)$$

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 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 $ct(\mathit{ANodes}(\mathbf{R}), \mathbf{R}$.

could
be done p
abilisticall
instead?
one s
equation?
Maybe d
that way
AI vs. VL
make
pseudocod

8 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 with a pruning step to eliminate redundant edges.

A key issue for scalability is that most of the learning time is taken up by forming table joins, whose size is the cross product of entity tables. These table joins provide the sufficient statistics required in model selection. To improve scalability, computing sufficient statistics needs to be feasible for cross product sizes in the millions or more. A possible solution may be the virtual join methods that compute sufficient statistics without materializing table joins, such as the Fast Möbius Transform [**?**; **?**].

A valuable direction for future work is to compare learning link correlations with directed and undirected models, such as Markov Logic Networks [**?**]. As we explained in Section 2, current relational learners for undirected models do not scale to most of our datasets. One option is to subsample the datasets so that we can compare the statistical power of directed and undirect learning methods independently of scalability issues. Khosravi *et al.* were able to obtain structure learning results for Alchemy [**?**], but did not evaluate the models with respect to link correlations. For the MLN-Boost system, we were able to obtain preliminary results on several benchmark databases (including Mutagenesis and Hepatitis), by selecting the right subset of target predicates. MLN-Boost is the current state-of-the-art learner for Markov Logic Networks [**?**]. The Bayes net models were competitive with the MLN-Boost models on a standard cross-validation measure of predictive accuracy.