Join Bayes Nets: A New Type of Bayes net for Relational Data

Oliver Schulte
Computer Science Dept.
Simon Fraser University
oschulte@cs.sfu.ca

Hassan Khosravi
Computer Science Dept.
Simon Fraser University
hkhosrav@cs.sfu.ca

Bahareh Bina
Computer Science Dept.
Simon Fraser University
bba18@cs.sfu.ca

Flavia Moser
Computer Science Dept.
Simon Fraser University
fmoser@cs.sfu.ca

Abstract

Many real-world data are maintained in relational format, with different tables storing information about entities and their links or relationships. The structure (schema) of the database is essentially that of a logical language, with variables ranging over individual entities and predicates for relationships and attributes. Our work combines the graphical structure of Bayes nets with the logical structure of relational databases to achieve knowledge discovery in databases. We introduce Join Bayes nets, a new type of Bayes nets for representing and learning class-level dependencies between attributes from the same table *and* from different tables; such dependencies are important for policy making and strategic planning. Focusing on class-level dependencies brings advantages in terms of the simplicity of the model and the tractability of inference and learning. As usual with Bayes nets, the graphical structure supports efficient inference and reasoning. We show that applying standard Bayes net inference algorithms to the learned models provides fast and accurate probability estimates for queries that involve attributes and relationships from multiple tables.

1 Introduction

Many real-world applications store data in relational format, with different tables for entities and their links. Standard machine learning techniques are applied to data stored in a single table, that is, in nonrelational, propositional or "flat" format [10]. The field of statistical-relational learning (SRL) aims to extend machine learning algorithms to relational data [6]. One of the major machine learning tasks is to use data to build a *generative statistical model* that represents the joint distribution of the random variables that describe the application domain [6]. In the single-table learning setting, the goal is often to represent predictive dependencies between the attributes of a single individual (e.g., between the intelligence and ranking of a student). In the SRL setting, the goal is often to represent in addition dependencies between attributes of different individuals that are related or linked to each other (e.g., between the intelligence of a student and the difficulty of a course given that the student is registered in the course). Many SRL models represent such dependencies on two different levels, a class dependency model and an instance dependency model. For instance, in a graphical SRL model, the nodes in the instance dependency model represent attributes of individuals or relationships [5]. The nodes in the class dependency model correspond to attributes of the tables. A class-level model is instantiated with the specific entities, their attributes and their relationships in a given database to obtain an instance dependency model. For instance, the class-level model may contain a node $age(S)$ to represent the age of a generic member of the student class, and the instance model may contain a node $age(Jack)$ to represent the age of a specific student $Jack$. The node $age(Jack)$ inherits the parameters and associations indicated at the class level for $age(S)$.

In this paper we apply Bayes nets (BNs) to model class-level dependencies between variables that appear in separate tables. What is new about our approach is that we focus on class-level variables only rather than making predictions about individual entities. Our class-level Bayes nets contain nodes that correspond to the descriptive attributes of the database tables, plus Boolean nodes that indicate the presence of a relationship; we refer to these as Join Bayes nets (JBNs). We introduce a new database join operation as a conceptual aid that provides semantics for JBNs. The focus on class-level dependencies brings advantages in terms of the simplicity of the model and the tractability of inference and learning, while it involves some loss of expressive power, because our BN model cannot answer queries about individual entities. Examples of applications that provide motivation for the class-level queries answered by our BN include the following.

(1) *Policy making and strategic planning.* A university administrator may wish to know which program characteristics attract high-ranking students, rather than predict the rank of a specific student in a specific program.

(2) *Query optimization* is one of the applications of SRL where a statistical model predicts a probability for given join conditions that can be used to infer the size of the join result [7]. The join conditions often do not involve specific individuals.

This paper defines JBN models and a probabilistic semantics for them. Our algorithmic contribution is an efficient dynamic programming procedure for parameter learning in JBNs. This algorithm solves the problem of estimating frequencies conditional on the *absence* of a relationship. Due to the construction of our Bayes nets, class-level queries can be answered using standard BN inference algorithms "as is".

Paper Outline We review background from relational databases and Bayes nets. Then we introduce our class-level Bayes nets and define their semantics. We describe algorithms for structure learning and parameter estimation. The algorithms and the inference capabilities of the Bayes nets they learn are evaluated on three data sets, one artificial and two real-word ones (the MovieLens and the Financial data set).

Related Work Researchers in statistical-relational learning have developed a number of generative models that include attributes and relationships of entities; for an overview see [8; 4; 3]. Markov Logic Networks (MLNs) are a prominent class of SR models that are based on undirected graphs [3]. The most direct comparison of JBNs is with other directed models; we discuss Bayes Logic Networks (BLNs) [9] and Probabilistic Relational Models (PRMs) [5, Sec.5.5.3]. Similar points of comparison apply to other SRL models.

The class-level model of a BLN—-called a Bayes Logic Program (BLP)—is syntactically similar to a JBN: a JBN with n nodes into a n translates into n BLP clauses of the form $x_i|parent_{i,1}, parent_{i,2}, \ldots, parent_{i,k}$, where $i = 1, \ldots, n$ indexes the nodes and node x_i has k parents. In addition, a BLP features *combining rules*. These specify how instance-level predictions from information about different related entities are to be combined into a single prediction. For instance, if the task is to predict a specific student's intelligence based on his grade in 10 courses he has taken, the class-level BLP clauses may be used to predict the intelligence based on a single course, and the combining rule would specify how to collect these predictions into a single prediction for the specific student. A feature of JBNs not necessarily present in BLPs is that variables ranging over entities are associated with entity types (e.g., S ranges over entities in the $Student$ table); the use of such types is key for the probabilistic semantics of JBNs. As for inference, it appears that in principle a BLP could be translated into a Bayes net and standard BN inference algorithm could be used to carry out class-level inference; to our knowledge, this approach to lifted inference with BLNs has not yet been evaluated.

The class-level model of a PRM is also a directed graphical model, and the nodes in the PRM graph are essentially the same as in Join Bayes nets (if the PRM includess uncertainty about the existence of links [5, Sec.5.5.3]). Nodes are associated with entity types as in a JBN. In the case in which entity types may be related to themselves (e.g., the $Parent$ relationship relates people to people), a PRM may contain self-loops. In order to make predictions about individual entities given the other entities they are related to, a PRM requires the specification of an aggregate function for many-many relationships [5, Def.5.2]. For instance, if the task is to predict a specific student's intelligence based on his grade in 10 courses he has taken, a PRM may specify that the prediction is to be based on the student's average grade. The CP-tables for a class-level PRM may be defined in terms of the value of the aggregate functions. In that case, standard BN algorithms cannot be applied to the class-level PRM, and adaptations are required [5].

In addition to inference, the two major differences between JBNs and PRMs resp. BLNs concern semantics and learning. (1) In terms of semantics, SR models are usually viewed as a template for instance-level models: For a given database, the class-level model is instantiated with the specific entities, their attributes and their relationships to obtain an instance-level model, which inherits the parameters specified at the class level. In contrast, we do not view our class-level BNs as templates for instance-level BNs. Thus we avoid problems with potential cycles at the entity level, which is a major concern for directed relational models [5]. (2) In order to make predictions about individual entities given the other entities they are related to, BLNs and PRMs require extra components in addition to the Bayes net-like class-level structure (combining rules resp. aggregate functions). While

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

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

these extra components considerably increase the expressive power of these models, they also substantially increase the complexity of learning. In particular, fitting the models to data requires evaluating their predictive power with regard to instance-level predictions that are based on the entire relational context of an entity. In contrast, inference and learning for JBNs can be carried out efficiently with algorithms whose design we outline in this paper.

2 Preliminaries

We employ notation and terminology from [11] 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. A Bayes net (BN) is a pair $\langle G, \theta_G \rangle$ where θ_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 $P(X = x|\mathbf{pa}_X^G)$. These conditional probabilities are specified in a **conditional probability table** for variable X or CP-table. We write $P(X_1 = x_1, ..., X_n = x_n) = p$, sometimes abbreviated as $P(x_1, ..., x_n) = p$, to denote that the joint probability of random variables $X_1, \ldots, X_n$ taking on values $x_1, \ldots, x_n$ is p. We also use vector notation $P(\mathbf{X} = \mathbf{x}) = p$.

We assume 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 are divided into *entity tables* and *relationship tables.* This is the case whenever a relational schema is derived from an entity-relationship model (ER model) [13, Ch.2.2]. The symbol E refers to entity tables, and the symbol R refers to relationship tables. Table 1 shows a relational schema for a university domain. A field or attribute named $name$ in table T is denoted by $T.name$. Each attribute has a domain of values denoted by $dom(T.name)$. The number of tuples in a table T for a database instance $\mathcal{D}$ is written as $|T|_{\mathcal{D}}$. We view a descriptive attribute of an entity table E as a deterministic function of an entity from E, and a descriptive attribute of a relationship table R as a deterministic function of entities linked by R. The relationship R itself can be viewed as a Boolean function that indicates for each entity tuple of the appropriate type whether it is linked by R. The **natural join** of two tables is the set of tuples from their cross product that agree on the values of fields common to both tables.

3 Join Bayes Nets and the Attribute-Relation Table

A Join Bayes net contains a node for each attribute field in the database, and a Boolean indicator node for each relationship table. The definition assumes that a given basic entity table is referenced at most once in a given relationship table. A generalization for the case in which entity sets may be related to themselves is treated in [12].

Definition 1 *A **Join Bayes Net** (JBN) structure for a* *database schema with entity tables and relationship tables is a DAG G with one node for each descriptive attribute $A.name$ in the database, whose domain is $dom(A.name)$, and one binary node for each relationship table in the database.*

We adopt the following functional *notation for the variables in a JBN*. We use a mnemonic Roman letter, e.g. V, to refer to a given entity table (e.g., S for the $Student$ table, C for the $Course$ table). An entity attribute node for the table is denoted by $name(V)$ (e.g., $ranking(S)$). The node for a descriptive attribute $R.name$ of a relationship table is denoted by $name(V_1, \ldots, V_k)$ where $V_1, \ldots, V_k$ refers to the entity tables linked to R by foreign key constraints (e.g., $grade(S, C)$). Similarly, the indicator node for R is denoted by $R(V_1, \ldots, V_k)$ (e.g., $Registered(S, C)$). Figure 1(e) shows a JBN for the university schema with this notation.

We associate with a given database $\mathcal{D}$ a joint distribution $P_{\mathcal{D}}$ over relationships and descriptive attributes, which is defined by a new join table—called the **attribute-relation table**—that is constructed as follows.

1. Form the cross product of all entity tables.
2. Extend the table with descriptive attributes of the relationship tables and one additional Boolean field for each relation. The boolean field for relationship table R takes the value T when the relationship R holds for the corresponding entity tuple and takes on the value F otherwise. When R is true for an entity tuple, the descriptive attributes of R are filled in with the corresponding values.
3. When R is false for an entity tuple, the descriptive attributes of R are assigned the value $\perp$ for "undefined".
4. Remove the primary key columns.

The attribute-relation table is viewed as a regular data table whose row frequencies represent a joint distribution over its columns, which is the **database distribution** P_D. Figure 1(d) shows the attribute-relation table for a small instance of the university schema.

Discussion The database distribution is closely related to joins as expressed in Datalog-style query languages like the DRC [13]. In logic queries, a table join corresponds to a conjunction; for instance, the join of the Registration table with the Student table selecting courses with $rating = 2$ is expressed by the query formula $\langle S, C : Registered(S, C), rating(C) = 2\rangle$. The probability assigned to this conjunction by the database distribution is the size of the join result in the database that corresponds to the conjunction, divided by the maximum size of the join result given the foreign key constraints:

$$P_{\mathcal{D}}(Registered(S, C) = T, rating(C) = 2) = \frac{|\langle S, C : Registered(S, C), rating(C) = 2\rangle|_{\mathcal{D}}}{|Student|_{\mathcal{D}} \times |Course|_{\mathcal{D}}} \tag{1}$$

Equation (1) illustrates that the probabilities assigned by the attribute-relation table have a natural alternative interpretation. It also implies that from an estimation of the database distribution $P_{\mathcal{D}}$, we can readily compute an estimate of join sizes, which is an important application for query selection.

We define the database distribution over the full cross product of the entities rather than just the join of entities with relationship tables. In relationship tables, entities with more links appear more frequently than others. As a result, the probability of an attribute value derived from the join of relationships with entities may not reflect the real statistical information. For instance, in Figure 1, in the join of the $Student$ table with the $Registered$ table the frequency of rows with $rating = 2$ is $1/2$, whereas the frequency of rows in the $Course$ table with $rating = 2$ is $1/3$. This is one of the problems often raised for basing statistical learning on a join table. The attribute relation table overcomes the problem by using the cross product of entities, so all entities from a given table appear in the same number of rows regardless of how many links they have. The subset of rows of the attribute-relation table in which the indicator variable $R = T$ corresponds to the join of the entity tables with the relationship table R. We now consider learning a JBN model for a given database distribution.

4 Parameter Estimation with a Virtual Join Algorithm

This section treats the problem of computing conditional frequencies in the database distribution, which corresponds to computing sample frequencies in the single table case. The main problem is computing probabilities conditional on the *absence* of a relationship. For instance, to compute $P_{\mathcal{D}}(difficulty(C) = 2|intelligence(S) = 3, Registered(S, C) = T)$, a frequency count on the constraints given by the query is done on the join of the $Registered$, $Student$, and $Course$ tables. However, computing conditional probabilities on queries with false relationships (e.g., $Registered(S, C) = F$) raises difficulties because it involves non-existent links (cf. [5]). This problem arises because a JBN includes relationship indicator variables such as $Registered(S, C)$, and building a JBN therefore requires modelling the case where a relationship does not hold. In principle, frequencies in the database conditional on the absence of links can be computed with frequency counts over the rows in the attribute-relation table where the link is absent. However, because materializing this table is generally not feasible, we instead use a *virtual join* algorithm that computes the frequencies in the entity join table without actually constructing the entity join. The virtual join algorithm is a dynamic programming algorithm for estimating joint probabilities in a database instance whose database operations involve only: (1) Joins of existing relationship tables with entity tables, and (2) Joins of existing relationship tables with other existing relationships tables that share an entity type (foreign key pointer). Relationship tables, such as $Registered$, are typically much smaller than the cross product of their related entities [5], so the join operations (1) and (2) are feasible for SQL queries, and our algorithm is much more efficient than explicitly constructing the attribute relation table.

Virtual Join Algorithm: Outline and Example Our algorithm computes joint probabilities. Conditional probabilities can easily be computed from joint probabilities via the equation $P(x|y) = P(x, y)/\sum x' P(x', y)$ where the summation is taken over all possible values of x. The basic idea can be described as follows. From probability laws, we have the relation

$$P(\mathbf{x}, R = F) = P(\mathbf{x}) - P(\mathbf{x}, R = T). \tag{2}$$

Equation (2) shows how we can reduce a probability involving a nonexistent relationship $R = F$ to two other computations that do not involve the nonexistent relationship: (1) the case in which we do not condition on the value of R, and (2) the case in which we condition on $R = T$. Let us consider first the case in which the joint probability involves only a single relationship variable together with descriptive attributes of entities (cf. [5, Sec.5.8.4.2]). In that case, the probability $P(\mathbf{x}, R = T)$ can be obtained from a frequency

(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)

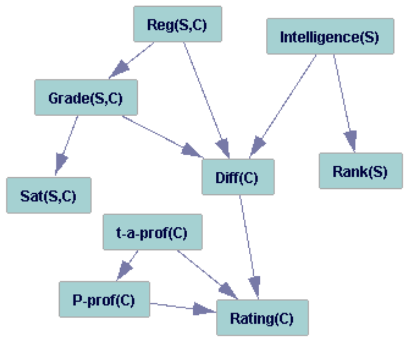


Figure 1: Database Table Instances: (a) *Student*, (b) *Registered* (c) *Course*. To simplify, we added the information about professors to the courses that they teach. (d) The attribute-relation table is formed in two steps: (1) take the cross product of the student and course table (3 x 3 = 9 rows) and extend it with the matching attribute and relationship information. (2) Remove the primary entity keys from the cross product of the entities. (e) A Join Bayes Net for the university schema variables.

count in the relationship table R in the database. The probability $P(\mathbf{x})$ may involve descriptive attributes from more than one entity table. It can be computed using the fact that distinct entity tables are independent, unless they are linked by a relationship variable [12], so the joint probability $P(\mathbf{x})$ is calculated by multiplying frequencies from entity tables.

Inductively, consider a joint probability involving $m > 0$ false relationships $R^1 = F, .., R^m = F$. Then first, change one of the false relationships to be true, e.g., $R^1 = T$, and compute the joint probability for this case recursively, since it involves one less false relationship. Second, change the state of the chosen relationship to be unspecified, e.g., $R^1 = \mathit{unspecified}$, and compute the conditional probability for this case recursively, since it involves one less false relationship. In our dynamic program, frequencies with fewer false relationship variables are computed first, so the two frequencies can be looked up from the previous computations.

Example. Figure 2 shows how to compute a joint probability with exactly one false relationship for the database instance of Figure 1. To illustrate the case with multiple relationships, suppose the university schema features another entity table $TA(\underline{ta\_id}, expertise)$ to record the expertise of teaching assistants and another relationship table relation $Assigned(\underline{ta\_id}, \underline{course\_id})$ to record which assistants are assigned to which course. Figure 2 shows how the computation of a joint probability with two false relationships can be reduced to two probabilities, each without the false relationship $Assigned(TA, C) = F$. [12] provides further implementation details, including pseudocode and complexity analysis. In the next section we apply the parameter estimation algorithm to build Join Bayes nets for three relational datasets.

5 Evaluation and Experiments

We present results of applying our learning algorithms to three relational data sets, the MovieLens and Financial real-world databases, and an artifical University database for the schema given in 1. Our evaluation method comprises the following steps.

1. Learn a JBN structure for each database. For comparison, we also apply a standard structure learning algorithm for Markov Logic Networks to each database.
2. Fill in the CP-tables with maximum likelihood estimates.
3. Apply a standard Bayes net inference algorithm to estimate conditional frequencies in the database, and compare the estimates to the result of directly computing conditional frequencies with SQL queries.

5.1 System Resources, Algorithms and Datasets

Our implementation used many of the procedures in version 4.3.9-0 of CMU's Tetrad package [2]. Our Java code is available from the senior author upon request. All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM.

Learning Algorithms A description of our structure learning method is beyond the scope of this note, but is provided in [12]. Our method is modular in that it upgrades *any* propositional single-table BN learner to a JBN learner. We used the Tetrad implementation of GES search [1] with the BDeu score (uniform structure prior, ESS=8) as the base single-table BN learning program. After learning a JBN structure, parameter estimation is carried out using the algorithm described in the previous section.

Inference Algorithms JBN inference was carried out with Tetrad's Rowsum Exact Updater algorithm. A direct comparison of class-level inference with other SRL formalisms is difficult as the implementations we could find support only instance-level queries. For example, both the Alchemy package for MLNs [3] and the Balios BLN engine [9] support only queries with ground atoms. We could not obtain source code for PRM inference.

Data sets Our datasets are available on-line at ftp://ftp.fas.sfu.ca/pub/cs/oschulte/datasets/.

University Database. In order to check the correctness of our algorithms directly, we manually created a small data set, based on the schema given in 1. The entity tables contain 38 students, 10 courses, and 6 Professors. The *Registered* table has 92 rows and the RA table has 25 rows.

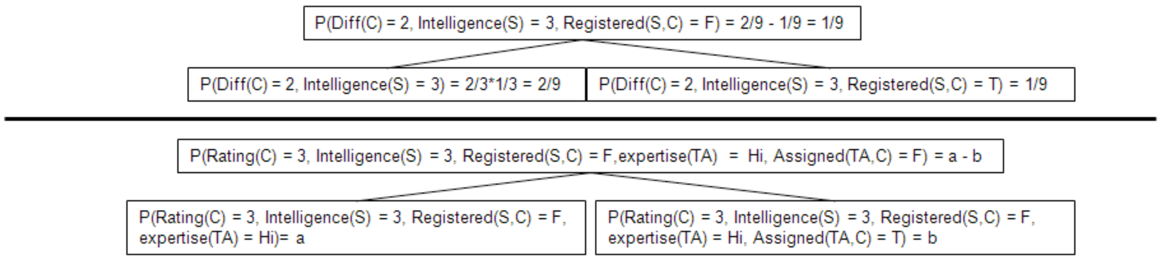

P(Diff(C) = 2, Intelligence(S) = 3, Registered(S,C) = F) = 2/9 - 1/9 = 1/9
P(Diff(C) = 2, Intelligence(S) = 3) = 2/3*1/3 = 2/9
P(Diff(C) = 2, Intelligence(S) = 3, Registered(S,C) = T) = 1/9
P(Rating(C) = 3, Intelligence(S) = 3, Registered(S,C) = F,expertise(TA) = Hi, Assigned(TA,C) = F) = a - b
P(Rating(C) = 3, Intelligence(S) = 3, Registered(S,C) = F, expertise(TA) = Hi)= a
P(Rating(C) = 3, Intelligence(S) = 3, Registered(S,C) = F, expertise(TA) = Hi, Assigned(TA,C) = T) = b

Figure 2: To illustrate the recursive scheme of our parameter estimation algorithm. The top example for the database instance in Figure 1 reduces the computation of a joint probability involving one false relationship to two without any false relationship indicators. The bottom example shows for a generic database instance how the computation of a joint probability involving two false relationships can be reduced to two with just one false relationship each.

MovieLens Database. The second data set is the MovieLens data set from the UC Irvine machine learning repository. It contains two entity tables: $User$ with 941 tuples and $Item$ with 1,682 tuples, and one relationship table $Rated$ with 100,000 ratings. The $User$ table has 3 descriptive attributes $age, gender, occupation$. We discretized the attribute age into three bins with equal frequency. The table $Item$ represents information about the movies. It has 17 Boolean attributes that indicate the genres of a given movie; a movie may belong to several genres at the same time. For example, a movie may have the value T for both the war and the $action$ attributes. The full table with 100,000 ratings exceeded the memory limits of Tetrad, so we randomly picked 40% of the ratings of the relationship table as input data.

Financial Database. The third data set is a modified version of the financial data set from the PKDD 1999 cup. We adapted the database design to fit the ER model. We have two entity tables: $Client$ with 5369 tuples and $Account$ with 4,500 tuples. Two relationship tables, $CreditCard$ with 5,369 tuples and $Disposition$ with 892 tuples relate a client with an account. The Client table has 10 descriptive attributes: the client's age, gender and 8 attributes on demographic data of the client. The Account table has 3 descriptive attributes: information on loan amount associated with an account, account opening date, and how frequently the account is used.

5.2 Experimental Results

We evaluate structure learning, parameter estimation and inference with Join Bayes nets. Table 2 presents a summary of the run time for parameter learning and structure learning for the data sets. The computation times are well within the range of practical feasibility (40 min for the most difficult experiment).

Learning The graphs learned are shown in Figures 1, 3, and 4. In the MovieLens data set, the algorithm finds a number of cross-entity table links involving the age of a user. Because genres have a high negative correlation, the algorithm produces a dense graph among the genre attributes. We simplified the graph by omitting genre variables that have only indirect links with the rating or User attributes. The richer relational structure of the Financial data set is reflected in a more complex graph with several cross-table links. The birthday of a customer (translated into discrete age levels) has especially many links with other variables.

The university database is small enough to materialize its attribute-relation table and verify the correctness of our parameter estimates directly. For the larger databases, this is not feasible. The next section provides an indirect way

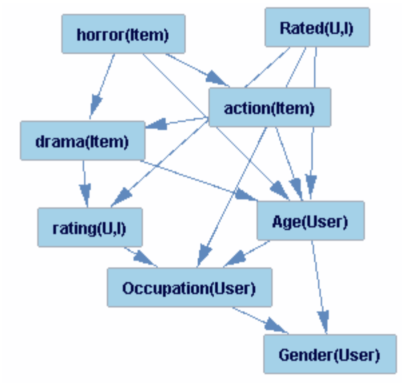


Figure 3: The JBN structures learned by our merge learning algorithmfor the MovieLens Data set.

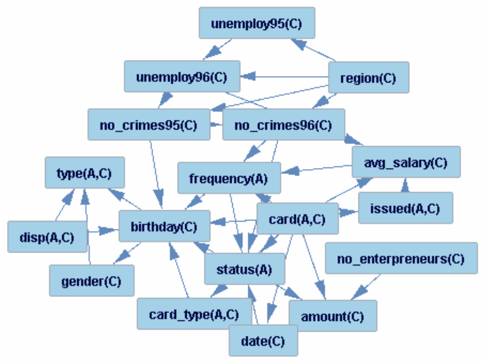


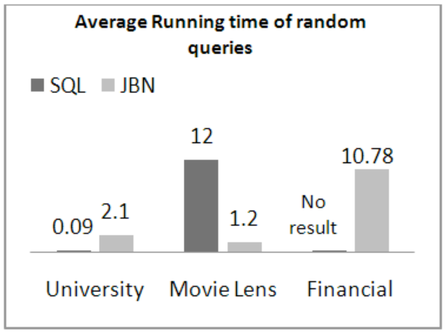
Figure 4: The JBN structures learned by our merge structure learning algorithmfor the Financial Data set.

to check the learning algorithms by comparing the probabilities estimated by the JBN with the database frequencies computed directly from SQL queries.

Inference To avoid bias, we randomly generated 10 queries, each involving 4 nodes, for each data set according to the following procedure. We compared the probabilities predicted by the JBN with the frequencies in the database

Data set	PL	SL in JBN
University	0.495	0.64
Movie Lens	2,018	135
Financial	2,472	574

Table 2: The run times—in seconds—for structure learning (SL) and parameter learning (PL) on our three data sets.



DataSet	University	MovieLens	Financial
Average Probability difference	0.003	0.027	N/A

Figure 5: Comparing the probability estimates and run times from the learned JBN models with SQL queries. Not all SQL queries for the Financial data set terminated with a result. The average is taken over 10 randomly generated queries.

as computed by an SQL query, as well as the run times for computing the probability using the JBN vs. the SQL. We do not expect the probabilities predicted by a JBN to be exactly the same as the data frequencies, for the same reason that in the single table case a BN learner would not just reproduce the sample frequencies: the absence of links in the graph entails probabilistic independence between variables that may be slightly correlated in the data. But since we use maximum likelihood estimates, and our sample sizes are not small, we would expect the predicted probabilities to be fairly close to the sample frequencies if the JBN structure is adequate. This expectation is confirmed by our results: we see in Figure 5 that the predicted probabilities are close to the data frequencies. For the small university data set, SQL queries are faster than JBN inference. But for the larger MovieLens data set, model inference is much faster, and for the largest Financial data set, SQL queries were infeasible when conditioning on the absence of relationships, whereas the JBN returns an answer in around 10 seconds. Where the SQL queries did return a frequency, it was close to the JBN estimate.

We observed that the number of tuples in the database table is a very significant factor for the speed of SQL queries but does not affect JBN inference. This is an important observation about the *data scalability of JBN inference*: While the learning algorithms depend on the size of the database, once the learning is completed, query processing is independent of database size. So for applications like query optimization that involve many calls to the statistical inference procedure, the investment in learning a JBN model is quickly amortized in the fast inference time.

6 Conclusion

We showed how Join Bayes nets can be used to represent class-level dependencies between attributes of entities or relationships. This contrasts with instance-level dependencies between attributes of specific entities. Class-level generic dependencies are of interest in themselves, and they support applications like policy making, strategic planning, and query optimization. We defined a new semantics for class-level Bayes nets based on a new database join operation. The focus on class-level dependencies brings gains in tractability of learning and inference. We described efficient and scalable algorithms for structure and parameter estimation in Join Bayes nets. Inference can be carried out with standard algorithms "as is". An evaluation of our methods on three data sets shows that our algorithms are computationally feasible for realistic table sizes, and that the learned structures represented the statistical information in the databases well. After learning has compiled the database statistics into a Join Bayes net, querying these statistics via the net is faster than directly with SQL queries, and does not depend on the size of the database.

References

[1] David Maxwell Chickering and Christopher Meek. Finding optimal bayesian networks. In *UAI*, pages 94–102, 2002.

[2] The Tetrad project: Causal models and statistical data, 2008. http://www.phil.cmu.edu/projects/tetrad/.

[3] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* [8].

[4] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[5] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [8].

[6] Lise Getoor and Ben Taskar. Introduction. In Getoor and Taskar [8], pages 1–8.

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

[8] Lise Getoor and Ben Tasker. *Introduction to statistical relational learning*. MIT Press, 2007.

[9] Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [8].

[10] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffmann, 1988.

[12] Oliver Schulte, Hassan Khosravi, Flavia Moser, and Martin Ester. Join bayes nets: A new type of bayes net for relational data. *CS-Learning Preprint Archive*, http://arxiv.org/abs/0811.4458, 2008.

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