Bayes Nets for combining logical and probabilistic structure

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

Abstract

We outline a new approach to using Bayes nets for a probabilistic extension of a logical structure or schema. 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 for relational structures. Another reason why relational structures are important is that they can represent other types of structures. We introduce a new type of Bayes nets for representing and learning class-level dependencies between attributes from the same table and from different tables. Because relational databases contain multiple tables, we cannot apply Bayes net learning algorithms as is. We outline novel learning algorithms that achieve efficiency by treating the database tables as a factored representation of the statistical information in the data.

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. The field of statistical-relational learning (SRL) aims to extend machine learning algorithms to relational data [3]. In the SRL setting, the goal is often to represent 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. 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. Our work applies Bayes nets (BNs) to model class-level dependencies between variables that appear in separate tables. 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). 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. We have developed efficient algorithms for structure and parameter learning in JBNs. Our parameter learning procedure is a dynamic program that addresses the problem of estimating frequencies conditional on the absence of a relationship. Our structure learning scheme upgrades a single-table Bayes net learner to a relational learner through a learn-and-merge approach where the learner is applied to single tables and then to join tables, and the results of learning on smaller tables constrain learning on bigger tables. Due to the construction of our Bayes nets, class-level queries can be answered using standard BN inference algorithms as is.

2 Preliminaries: Bayes Nets, Logic, Notation

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 $P(X = x|\mathbf{pa}_X^G)$.

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*. 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.

It is well known that the structure of a relational schema can be represened in the logical structure of a formal first-order vocabulary. By translating the data structure into logical structure, we can use logic as a formal foundation for specifying the syntax and semantics of our Bayes net models for the database. One standard way to translate a relational schema into logic is by representing descriptive attributes as functions of entities and entity tuples. Formally, we define the following logical language $\mathcal{L}$ for a given database schema $\mathcal{D}$. [5] generalizes this definition for relationships with arity greater than 2 and types with more than one variable.

1. A finite list of constants $c_1, c_2, \ldots$, which includes the special constants $T, F, \perp$. The first two stand for truth values, and $\perp$ denotes "undefined".
2. A finite list of unary and binary function symbols f_1, f_2. Each function f has as its range a finite set of constants, denoted by $range(f)$. A function whose range equals $\{T, F\}$ is a predicate. We use uppercase Roman letters

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

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

like E, R and variants to distinguish predicates from other function symbols.

3. A unary predicate E is also called an (entity) **type**. For each type E, we have one variable X_E of type E. A **term** of type E is either the variable X_E or a constant that satisfies $E(c) = T$. We write θ for a generic term and use the vector notation $\boldsymbol{\theta}$ for a list of terms. The **domain** of X_E, denoted as $dom(X_E)$, comprises the set of constants of type E.
4. Each unary function f that is not a predicate is associated with an argument type E; intuitively the arguments of f must be of type E. In this case we say that f is a descriptive attribute of type E. Each binary function, including predicates, is similarly associated with an argument type pair (E_1, E_2).
5. A binary predicate R is also called a **relationship**. Each binary function symbol g is associated with a relationship R such that the function is defined for arguments of the appropriate type if and only if they are related by R. That is, the function g satisfies the constraint that $g(\theta_1, \theta_2) = \bot$ if and only if $R(\theta_1, \theta_2) = F$. In that case we say that g is a descriptive attribute of R.
6. An atomic **assignment** is a statement of the form $f(\boldsymbol{\theta}) = a$ where the terms $\boldsymbol{\theta}$ are of the right type for f and a is in the range of f. An **assignment** F is a finite conjunction of assignments, written Prolog-style as $f_1(\boldsymbol{\theta}_1) = a_1, f_2(\boldsymbol{\theta}_2) = a_2, \ldots$.
7. A substitution $F[X/a]$ is the result of replacing all occurrences of variable X in F by a constant a of the same type as X. A **ground** assignment contains no variables. A **grounding** of F is a simultaneous substitution of values for all variables in F. A given database instance $\mathcal{D}$ determines for each ground assignment whether all assignments in the clause are true in $\mathcal{D}$. We denote by $|F|_{\mathcal{D}}$ the number of groundings of an assignment F that result in a conjunction that is true in $\mathcal{D}$.

Example. Consider the logical language for the database instance $\mathcal{D}$ shown in Figure 1. Let S be a variable of type *Student* and C a variable of type *Course*. An example assignment is

$$ranking(S) = 1, difficulty(C) = 2, Registered(S, C) = F.$$

With the grounding $S/Jack, C/101$, the assignment is false in $\mathcal{D}$ because $difficulty(101) = 1$ in $\mathcal{D}$. With the grounding $S/Jack, C/103$, the assignment is true in $\mathcal{D}$.

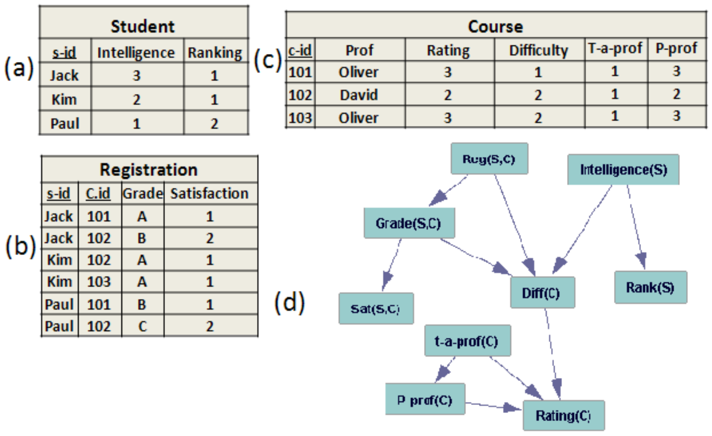
Join Bayes Nets One of the main aims of a Join Bayes net is to model correlations between attributes of related objects at the class-level. In our logical language, a class-level model is based on variables rather than constants. Define a **variable function term**, or function term for short, to be a term of the form $f(X)$ or $f(X_1, X_2)$ where the variable(s) is (are) of the appropriate argument type for f.

Definition 1 *A **Join Bayes Net** (JBN) structure for a database schema with associated logic $\mathcal{L}$ is a DAG whose nodes are exactly the variable function terms in $\mathcal{L}$. The domain of values of a node $f(\boldsymbol{\theta})$ is the range of f.*

A given database instance $\mathcal{D}$ defines a natural probability for a given assignment F: the ratio of groundings which make F true in $\mathcal{D}$ over the number of possible groundings given the type constraints on F. Formally, let $X_1, \ldots, X_k$ be the list of variables that occur in F. The quantity $|dom(X)|$ is the size of the domain of X, that is, the number of constants of the same type as X. The probability of a joint assignment is then given by:

$$P_D(F) = \frac{|F|_{\mathcal{D}}}{|dom(X_1)| \times |dom(X_2)| \times \cdots \times |dom(X_k)|}.$$

From a statistical point of view, this formula defines a joint data distribution over the variables in the JBN. The remainder of the paper considers learning JBN models from this distribution.


(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

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) A Join Bayes Net for the university schema variables.

3 Learning Join Bayes Nets

In principle, learning tasks for a JBN can be performed by first constructing a large table enumerating all possible groundings of the variables in the relational language, and then applying a regular propositional BN learning algorithm to this table. But since this approach is not computationally feasible, the challenge is to develop learning algorithms that treat the tables in the database as a factored representation of the database distribution.

Structure Learning We outline a general schema for upgrading a standard single-table propositional BN learner to a statistical relational learner that performed well in our experiments. The propositional learner is used as a function call in the body of our algorithm. We require that the propositional learner takes as input, in addition to a single table of cases, also a set of edge constraints that specify required and forbidden directed edges. The output of the algorithm is a DAG G for a database $\mathcal{D}$ with variables as specified in Definition 1. Our approach is to "learn and merge": we apply the BN learner to single tables and combine the results successively into larger graphs corresponding to larger table joins. First, we apply the BN learner to single tables. Next, we form larger tables by joining each relationship table R with the entity tables that are linked to R by foreign key constraints. The BN learner is applied to these join tables with the constraint that if two variables already appear in a single table, the BN learner should introduce an edge between them in the join table search if and only if the BN learner introduced one previously in the single table search. In the final phase, we form larger tables by joining two of the previous join tables. The BN learner is applied to each 2nd-order join, again with the constraint that edges betweens pairs of variables previously considered must agree with the BN learned for the 1st-order join.

Parameter Estimation Parameter estimation is chiefly the problem of computing conditional frequencies in the database distribution. The main problem is computing probabilities conditional on the absence of a relationship [2] without materializing the set of tuples that are not related. Our basic principle is to use a 1 minus trick: We recursively apply the laws of probability to derive the probability of an event conditional on a relationship not holding from (1) the probability conditional on the relationship holding, and (2) the unconditional probability with the relationship status unspecified. A full description of the algorithm with pseudocode may be found in [5].

Evaluation We carried out several experiments to evaluate the correctness and feasibility of our algorithms. All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. For single table BN search we used GES [1] with the BDeu score (structure prior uniform, ESS=8). We analyzed three datasets: a synthetic one for the University domain, the Financial Database from the PKDD 1999 cup, and the MovieLens Database from the UC Irvine machine learning repository. Due to space constraints, we show the JBN only for the MovieLens dataset, which contains two entity tables: User with 941 tuples and Item with 1,682 tuples, and one relationship table Rated with 40,000 ratings. The User table has 3 descriptive attributes age; gender; occupation. The table Item represents information about the genre of a given movie. The learned graph is shown in Figure 2, and runtimes in Table 2.

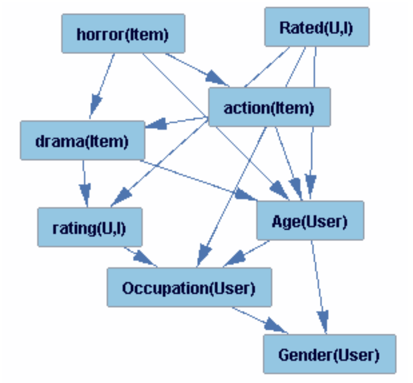


Figure 2: The JBN structures learned by our merge learning algorithmfor the MovieLens Data set.

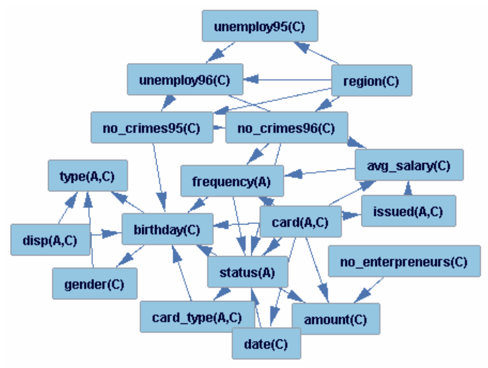


Figure 3: The JBN structures learned by our merge structure learning algorithmfor the Financial Data set.

Data set	**PL**	**SL in JBN**	**SL in MLN**
University	0.495	0.64	22.91
Movie Lens	2,018	135	Terminated w/o Result
Financial	2,472	574	Terminated w/o Result

Table 2: The run times—in seconds—for structure learning (SL) and parameter learning (PL) on our three data sets. MLN inference was carried out with Alchemy.

References

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

[2] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [4].

[3] Lise Getoor and Ben Taskar. Introduction. In Getoor and Taskar [4], pages 1–8.

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

[5] 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.