An Introduction to Probabilistic Graphical Models for Relational Data

Lise Getoor
Computer Science Department/UMIACS
University of Maryland
College Park, MD 20740
getoor@cs.umd.edu

Abstract

We survey some of the recent work on probabilistic graphical models for relational data. The models that we describe are all based upon 'graphical models' [12]. The models can capture statistical correlations among attributes within a single relational table, between attributes in different tables, and can capture certain structural properties, such as the expected size of a join between tables. These models can then be used for a variety of tasks including filling in missing values, data summarization and anomaly detection. Here we describe two complementary semantics for the models: one approach suited to making probabilistic statements about individuals and the second approach suited to making statements about frequencies in relational data. After describing the semantics, we briefly describe algorithms for automatically constructing the models from an existing (non-probabilistic) database.

1 Introduction

There has been a growing interest in probabilistic databases. This growing interest is largely due to expanding data collection opportunities and capabilities. In addition, recent advances in probabilistic modeling, both in terms of efficient learning and estimation algorithms and practical inference algorithms have made the use of these models for large-scale domains practical. In this survey paper, we review some of the work developed in the probabilistic reasoning and machine learning communities commonly refered to as *probabilistic relational models* (PRMs) [11]. PRMs are based on existing probabilistic models (typically Bayesian networks) and extend them to relational and first-order domains.

Halpern [9] identified two distinct semantics for first-order logics of probability, refered to as the *possible-worlds* approach and the *domain-frequency* approach. In the possible-worlds approach, we put a distribution over possible worlds. The distribution can be used to answer degree of belief questions such as 'What is the probability that a particular person, Joe Smith, has purchased a particular tractor part, Widget Supreme?'. In this case, the semantics of the formula is defined by summing up the probability over all of the possible worlds in which Joe Smith has bought Widget Supremes. However, as many have noted [2, 1], there are difficulties with making statistical assertions, such as '75% of the purchases were for widgets'. The second approach, refered to

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

as the domain-frequency approach, is appropriate for giving semantics to statistical queries such as, 'What is the probability that a randomly chosen individual has purchased a widget?'. In this case the semantics are defined in terms of a particular world, and are the result of an experiment performed in this world. In this approach, statements such as '80% of the purchases are made by men' make sense. Whereas questions such as, 'What is the probability Joe Smith purchased a Widget Supreme?' do not make sense; the probability of the purchase, in a particular world, is either 0 or 1.

Here we survey some of the recent work on probabilistic graphical models for relational data and show how the two semantics are supported. We begin by introducing the syntax for probabilistic graphical models for relational data. We then describe the two alternate semantics for the models, and describe how they support the ability to make inferences about missing values and estimate the expected result set size for certain queries. Then, we briefly describe algorithms for automatically constructing the models.

2 Graphical Models for Relational Data: Syntax

First, we describe common syntactic elements of probabilistic relational models: the relational schema and the probabilistic schema. After these are introduced, we describe alternate semantics for the models.

2.1 The Relational Language

The relational schema defines the structure of our domain; it defines the tables and the attributes of the tables in the database. We assume a simple model, which obeys certain basic integrity constraints. In our relational model each table has a primary key, some descriptive attributes and possibly some foreign keys which are references to keys into other tables. For simplicity we assume primary keys are a single attribute, but this requirement can easily be lifted.

Definition 1: A *relation schema* $\mathcal{R}$ consists of a set of tables $\mathcal{R} = \{R_1, \ldots, R_m\}$. Each table R is associated with attributes of three types: a primary key $R.K$, a set $\mathcal{F}(R)$ of foreign keys, and a set $\mathcal{A}(R)$ of descriptive attributes. Each foreign key $R.F$ references a table $S \in \mathcal{R}$ and its domain, $\mathcal{V}(R.F)$, is a subset of $\mathcal{V}(S.K)$, $\mathcal{V}(R.F) \subseteq \mathcal{V}(S.K)$. Each descriptive attribute $R.A$ is associated with a domain of possible values $\mathcal{V}(R.A)$. ∎

Definition 2: A database instance $\mathcal{D}$ over $\mathcal{R}$ consists of a set of tuples $\mathcal{D}(R)$ for each table R. For each $t \in \mathcal{D}(R)$: (1) The primary key $t.K$ is unique within R; (2) For each $F \in \mathcal{F}(R)$, if F refers to tuples from S, $t.F$ is the primary key of some tuple in $\mathcal{D}(S)$; and, (3) For each $A \in \mathcal{A}(R)$, $t.A$ is a value in $\mathcal{V}(R.A)$. ∎

Note that, for simplicity, we assume referential integrity, but our model can be extended to accommodate cases where the foreign key may be *null*, indicating that there is no related tuple.

2.2 The Probabilistic Schema

In a probabilistic schema, we model the conditional dependence of descriptive attributes on other attribute values. To do this, we use a graphical dependence structure within which we can model local conditional probability distributions (CPDs) between descriptive attributes. The probabilistic schema gives a template which describes both the graphical dependence structure of the attributes and the parameters of the local conditional probability distribution for an attribute. Here, we define a simple generic syntax for the probabilistic schema; as we will see in the next sections, alternate probabilistic semantics can be defined using the same basic syntax.

Definition 3: A *probabilistic schema* $\mathcal{PS}$ for a relational schema $\mathcal{R}$ is a pair $(\mathcal{S}, \mathbf{\Theta})$. $\mathcal{S}$ is a directed graph with a node for each descriptive attribute $R.A$ in each of the tables in $\mathcal{R}$. For each $R.A$, $\mathcal{S}$ specifies the parents of each node, denoted Pa$(R.A)$, and $\mathbf{\Theta}$ specifies a CPD, written $P(R.A \mid \text{Pa}(R.A))$. ∎

The parents of a node $R.A$ are defined via a query over the relational schema. In the simplest case, a parent may be an attribute $R.B$ in the same table, in which case we annotate the edge with $\sigma_B(R)$. Another simple case is one in which a parent is defined via a foreign-key join. Suppose that $R.F$ is a reference to $S.K$ and the values of attribute $R.A$ are correlated with the corresponding attributes of $S.C$. In this case, we make $S.C$ to be a parent of $R.A$, and annotate the edge with $\sigma_C(R \bowtie_{R.F=S.K} S)$. By default, each edge represents the dependence on a single value. If the relationship between R and S is one-many, we compute an aggregate γ over C, where γ is any single-valued deterministic function over the multiset. The edge is then annotated with $\gamma_C(R \bowtie_{R.F=S.K} S)$. More complex chains of joins can be used to define parents. Furthermore, rather than specifying each parent as an independent query, it is also possible, though less common, to define a set of parents via a relational query and make use of arbitrary user-defined aggregates [6].

Each $R.A$ has an associated CPD; $\mathbf{\Theta}$ is the entire collection of CPDs. The CPD describes the generic statistical relationship between each node and its parents. The CPD specifies the distribution over the values of $R.A$ given any possible assignment of values to its parents.[1] Let $\mathcal{V}(R.A)$ denote the space of possible values for $R.A$ and $\mathcal{V}(\text{Pa}(R.A))$ denote the space of possible values for the parents of $R.A$. The CPD is *legal* if all of the conditional probabilities are non-negative, and if, for any particular instantiation of $R.A$, $a \in \mathcal{V}(R.A)$, the sum over all possible instantiations of $R.A$, for any particular instantiation, $\boldsymbol{u}$, of $R.A$'s parents is 1, i.e.,

$$\sum_{a \in \mathcal{V}(R.A)} P(R.A = a \mid \text{Pa}(R.A) = \boldsymbol{u}) = 1.$$

The CPD is a function that may be represented in a number of ways. A common representation is as a table of multinomials, in which we have a multinomial distribution for each possible instantiation of parents.. Alternatively, the CPD can be represented as a tree [3], where the interior vertices represent splits on the value of some parent of $R.A$, and the leaves contain distributions over the values of $R.A$. In this representation, we find the conditional distribution over $R.A$ given a particular choice of values $A_{k_1} = a_1, \ldots, A_{k_\ell} = a_\ell$ for its parents by following the appropriate path in the tree down to a leaf; when we encounter a split on some variable A_{k_j}, we go down the branch corresponding to the value of a_j; we then use the distribution stored at that leaf. There are a number of other possible compact functional forms for the distribution such as noisy-or and noisy-max.

3 Semantics

The previous section gave syntax for probabilistic relational models. In this section, we define two distinct semantics for the models. One, which we call the *possible worlds* approach, is based on defining a distribution over possible database instances. We will see that this approach is useful when we are attempting to generalize from observed data. This is useful for example, if we are trying to infer values for missing attributes in an incomplete database. The other semantics, which we call the *domain frequency* approach, is based on the defining a distribution over a randomly chosen tuple (or a collection of randomly chosen tuples). We will see that this approach is useful when we are attempting to represent a compact statistical summary of a particular database instance. The two semantics correspond, respectively, to the Type II and Type I approaches described in Halpern [9].

3.1 Probabilistic Relational Models: Possible Worlds Semantics

PRMs, as originally introduced [11], can be seen as extending directed graphical models, commonly refered to as Bayesian networks, to relational domains. The PRM syntax specifies a template for a probability distribution over a database. The template includes the relational component, that describes the relational schema for our

[1] Most often we assume that the attribute domains are discrete and finite; continuous domains can also be supported in which case we must specify conditional density functions.

domain, and the probabilistic component, that describes, in an abstract manner, the probabilistic dependencies that hold in the domain.

The possible worlds semantics for PRMs, which we will denote as PRM$_{pw}$, provides a coherent formal semantics in terms of probability distributions over sets of relational logic interpretations. Given a set of ground objects, a PRM$_{pw}$ specifies a probability distribution over a set of interpretations involving these objects. A probabilistic schema, together with a partial database describing tuples and relations, defines a probability distribution over the unspecified attributes of the tuples. As such, it can be used effectively for representing *degree of belief*, which for example can be used when reasoning over missing values is required.

We will refer to a database instance $\mathcal{D}$ with no missing or unknown values as a *complete instantiation* of a schema. Each of these complete instantiations is considered a possible world, and a PRM$_{pw}$ defines a probability distribution over database instances. In order to make this probability space well-defined, we need to constrain the space in some manner. Several different ways of specifying the probability space have been studied, with varying representational power.

Attribute uncertainty is the simplest way of defining the probability space. Intuitively, we assume that the set of objects and the relations between them are fixed, i.e., external to the probabilistic model. We specify the tuples and relations using a relational skeleton. Then, the PRM$_{pw}$ defines a probability distribution over the assignments to the attributes of the objects in the model.

Definition 4: A *relational skeleton* $\mathcal{D}_s$ of a relational schema is a partial specification of a database instance. It specifies the value for the primary key $R.K$ and all of the foreign keys $F \in \mathcal{F}(R)$; however, it leaves the values of the attributes unspecified.

A PRM$_{pw}$ defines a distribution over the possible worlds consistent with the relational skeleton. The relational skeleton implicitly defines the random variables in our domain; we have a random variable for each attribute of each object in the skeleton. A PRM$_{pw}$ then specifies a probability distribution over *completions* $\mathcal{D}$ of the skeleton.

The probabilistic schema defines the qualitative dependency structure, $\mathcal{S}$. As we saw, the dependency structure associates with each attribute $R.A$ a set of parents Pa$(R.A)$. These correspond to *formal* parents; for different objects, the relational skeleton $\mathcal{D}_s$ determines the *actual* parents. In other words, the formal parents are defined by the probabilistic schema, however the relational skeleton provides the interpretation. The interpretation tells us which tuples join; this in turn tells us which attributes depend on each other. We will typically use capital letters to denote the generic attributes, $R.A$, and use lowercase letters to denote tuple variables, so that $r.A$ refers to the attribute of a specific tuple.

Definition 5: A *probabilistic relational model* Π_{pw} for a relational schema $\mathcal{R}$ and a probabilistic schema $\mathcal{PS} = (\mathcal{S}, \mathbf{\Theta})$, with *possible worlds semantics* is a triple $(\mathcal{S}, \mathbf{\Theta}, \mathcal{D}_s)$. The probabilistic schema together with the relational skeleton induces a "ground" Bayesian network:

- There is a random variable for every attribute of every object $r \in \mathcal{D}_s$, $r.A$.
- Each $r.A$ depends probabilistically on its parents according to the probabilistic schema and the relational skeleton, Pa$_{\mathcal{D}_s}(r.A)$.
- The CPD for $r.A$ is $P(R.A \mid \text{Pa}(R.A))$, as specified by the probabilistic schema.

The PRM$_{pw}$ defines the following distribution:

$$P(\mathcal{D} : \mathcal{S}, \mathbf{\Theta}, \mathcal{D}_s) \quad = \quad \prod_{R_i \in \mathcal{R}} \prod_{r \in \mathcal{D}_s(R_i)} \prod_{A \in \mathcal{A}(R_i)} P(r.A \mid \text{Pa}_{\mathcal{D}_s}(r.A))$$

∎

This expression is very similar to the chain rule for Bayesian networks. There are three primary differences. First, our random variables correspond to the attributes of the tuples defined by the relational skeleton. Thus, a different relational skeleton will result in a distribution defined over a different set of random variables. Second, the set of parents of a random variable can vary according to the relational context of the object — the set of objects to which it is related. These are also determined by the relational skeleton. Third, the parameters are shared; the parameters of the local probability models for attributes of objects in the same class are identical.

As in any definition of this type, we must take care that the resulting function from instances to numbers defines a coherent probability distribution, i.e., the probability of each instance is between 0 and 1, and the sum of the probability of all instances is 1. In Bayesian networks, where the joint probability is also a product of CPDs, this requirement is satisfied if the dependency graph is acyclic: a variable is not an ancestor of itself. A similar condition is sufficient to ensure coherence in PRM$_{pw}$s as well. Additional details are given in [4].

In the discussion so far, we have assumed that the relational skeleton is external to the probabilistic model. The PRM$_{pw}$ framework can be extended to accommodate uncertainty about the structural relationships between objects as well as about their properties [5]. One approach to structural uncertainty we have developed is called *existence uncertainty*. Existence uncertainty is a simple approach to modeling the probability that a relationship exists between any two entities. Suppose we have the relation $Purchase(Key, Person, Item, Price)$. We add a Boolean attribute, the *Exists* attribute, and build a probabilistic model for it, just as any other attribute in our domain. We can model the probability that a person purchases a particular item. This can depend on both properties of the person and properties of the item. The approach is described more fully in [5]. We have also investigated the representation and learning of PRMs with class hierarchies. A discussion of these issues can be found in [4].

3.2 Probabilistic Relational Models: Domain Frequency Semantics

An alternative semantics for PRMs allows them to capture domain frequency information. These semantics describe a statistical model of a *particular* database instantiation. The model captures the tuple frequencies in the database, and in particular it captures the frequencies with which tuples *join*. We will refer to this flavor of probabilistic relational models as a PRM$_{df}$. These semantics are useful for describing a compact statistical model of a database. This compact model can then be used to efficiently answer question about the *expected* number of tuples that will satisfy a query. What makes PRM$_{df}$s unusual is they model correlations across tuples and they can flexibly answer a wide collection of queries. This has many possible uses, including for use by a query optimizer in choosing the appropriate query plan, and for use in computing approximate aggregate query answers. These semantics were first introduced in [8] and their utility for selectivity estimation was shown. Here, we give an introduction to the semantics, but for full details see [4].

As a simple illustration of the domain frequency semantics that we hope to capture, consider two tables R and S such that R has a foreign key, $R.F$, that points to $S.K$. We define a joint probability space over R and S using an imaginary sampling process that randomly samples a tuple r from R and independently samples a tuple s from S. The two tuples may or may not join with each other. We introduce a new *join indicator variable* to model this event. This variable, J_F, is binary valued; it is true when $r.F = s.K$ and false otherwise.

This sampling process induces a distribution

$$P_{\mathcal{D}}(J_F, A_1, \ldots, A_n, B_1, \ldots, B_m)$$

over the values of the join indicator J_F, the descriptive attributes of R, $\mathcal{A}(R) = \{A_1, \ldots, A_n\}$, and the descriptive attributes of S, $\mathcal{A}(S) = \{B_1, \ldots, B_m\}$.

Now, consider any query Q over R and S of the form: $r.\mathbf{A} = \mathbf{a}, s.\mathbf{B} = \mathbf{b}, r.F = s.K$ (where we abbreviate a multidimensional select using vector notation). The probability that this query is satisfied by a randomly chosen tuple r from R and s from S is simply:

$$P_{\mathcal{D}}(\mathbf{A} = \mathbf{a}, \mathbf{B} = \mathbf{b}, J_F = \textit{true}).$$

In other words, we can estimate the result for any query of this form using the joint distribution $P_{\mathcal{D}}$ defined using our sampling process.

Generalizing from this example to more general select-join queries, let Q be a query over tuple variables $r_1, \ldots, r_k$ (which may or may not refer to the same tables). Let σ_Q be the set of equality select clauses in Q; i.e., $\sigma_Q = \{r_{i_1}.A_{i_1} = a_{i_1}, \ldots, r_{i_l}.A_{i_l} = a_{i_l}\}$ and let $\bowtie_Q$ be the set of foreign-key joins in Q; i.e., $\bowtie_Q = \{r_{j_1}.F_{j_1} = s_{j_1}.K, \ldots, r_{j_m}.F_{j_m} = s_{j_m}.K\}$. We can write Q as follows:

$$\bowtie_Q (\sigma_Q(R_1 \times \ldots \times R_k)).$$

We introduce an indicator variable $\mathcal{I}_Q$ indicating when the equalities in Q hold.

Definition 6: For any query Q over database $\mathcal{D}$, the joins in Q induce a distribution $P_{\mathcal{D}}(Q)$ over the attributes of the tuple variables in Q, $R_1 \times \ldots \times R_k$. The probability that the query is satisfied is:

$$P_{\mathcal{D}}(\mathcal{I}_Q) = \frac{|\bowtie_Q (\sigma_Q(R_1 \times \ldots \times R_k))|}{|R_1| \times \ldots \times |R_k|} . \blacksquare$$

We can also view this joint distribution as generated by an imaginary process, where we uniformly and independently sample a sequence of tuples $r_1, \ldots, r_k$ from $R_1, \ldots, R_k$, and then select as the values of $A_1, \ldots, A_n$ the values of $r.A_1, \ldots, r.A_n$.

While the expression in Definition 6 is computable for any select-join query, we will find it more useful if we can define a unique distribution induced by our database. To achieve this, we restrict our attention to a finite, acyclic collection of foreign-key joins. If there exists a partial ordering $\prec$ over the tables in our schema $\mathcal{R}$, such that, for any foreign key $R.F$ with $\mathcal{V}(R.F) \subseteq \mathcal{V}(S.K)$, $S \prec R$, then we say that $\mathcal{R}$ is *table-stratified*.

Now we can define a new relation $\mathcal{U}$, which is the *universal foreign-key closure* of a database $\mathcal{D}$ with respect to a table stratification. This relation is never actually materialized, we merely use it as a tool in defining a unique distribution induced by our database. Intuitively, the query that we construct introduces a tuple variable for each table in our schema, and has a unique tuple variable for each foreign-key in the schema.

Definition 7: Let $\mathcal{D}$ be a database with relational schema $\mathcal{R}$ and let $\prec$ be a table stratification of $\mathcal{D}$. The *universal foreign-key closure* of $\mathcal{D}$ is defined by the query $\mathcal{U}$ we construct below. $\mathcal{T}(\mathcal{U})$ will be the set of tuples variables in our query. Initially, $\mathcal{T}(\mathcal{U})$ has one tuple variable for each of the tables that are leaves in $\prec$. Each tuple variable is initially marked unprocessed.

We will construct the full set of tuple variables in the query as follows. While there are tuple variables in $\mathcal{T}(\mathcal{U})$ that have not been processed:

- Let r be an unprocessed tuple variable in $\mathcal{T}(\mathcal{U})$.
- Suppose r is a tuple from R. For each $F \in \mathcal{F}(R)$, where $R.F$ refers to S, add a new unique tuple variable s to $\mathcal{T}(\mathcal{U})$. This tuple variable is marked unprocessed. We say that s is the tuple variable *associated* with $r.F$. We also add the join $r.F = s.K$ to $\bowtie_{\mathcal{U}}$.

Let $\mathcal{A}(\mathcal{U}) = \{A_1, \ldots, A_m\}$ be the attributes in $\mathcal{U}$ (in order to avoid ambiguity, assume attributes are prefixed by their associated tuple variable), and let $\mathcal{T}(\mathcal{U}) = \{t_1, \ldots, t_j\}$ be the tuple variables in the query. $\mathcal{U}$ is simply a query over the cross product of the relations with a new copy of the relation introduced for each tuple variable that we add. $\blacksquare$

Given this query $\mathcal{U}$, we can define the probability distribution $P_{\mathcal{U}}$. It is the distribution induced by the occurrence frequency of different value combinations for the attributes of the tuple variables in $\mathcal{U}$ and join events among the tuple variables. PRM$_{df}$s allow us to compactly model $P_{\mathcal{U}}$. A probabilistic schema $\mathcal{PS}$ describes a set of independence assumptions and the local distributions of attributes given their parents. Intuitively, we say that a database $\mathcal{D}$ is a model of a probabilistic schema $\mathcal{PS}$, $\mathcal{D} \models \mathcal{PS}$, if the conditional independence assumptions made in $\mathcal{PS}$ hold in $\mathcal{D}$ and if the local distributions match the frequencies in $\mathcal{D}$. This is made more precise in [4].

Further, PRM$_{df}$s allow us to efficiently answer certain frequency queries by constructing a small query-specific Bayesian network that can be used to compute the desired frequencies. We begin by restricting attention to a form of select-join queries over multiple tables we call *inverted-tree foreign-key-join* queries. These queries are over a subset of the tuple variables of the universal foreign-key closure. Intuitively, they are over an upwardly closed fragment of the forest defined via the foreign-key joins in $\mathcal{U}$, and may themselves form a forest. We refer to these as *legal* queries.

Given a probabilistic schema $\mathcal{PS}$ and a legal query Q, the domain frequency semantics are defined as follows:

Definition 8: Suppose PRM$_{df}$ consists of $\mathcal{PS} = (\mathcal{S}, \mathbf{\Theta})$ over $\mathcal{D}$ and $\mathcal{D} \models \mathcal{PS}$. Let Q be a legal query. $Vars(Q)$ is a set of random variables which includes, for each attribute $r.A$ of every tuple variable $r \in Q$, $r.A$, and for each $r.F = s.K$, it has a random variable $r.J_F$. For every node $r.V$ introduced, $r.V$ has parents Pa$(r.V)$ defined by $\mathcal{S}$ and the CPD of $r.V$ is as specified in $\mathbf{\Theta}$. Then, $\mathcal{PS}$ defines the following distribution:

$$P_{\mathcal{D}}(Q) = \prod_{A_i \in Vars(Q)} P(A_i \mid \mathrm{Pa}A_i, J^*_{\mathcal{U}}(A_i) = \mathbf{T})$$

$J^*_{\mathcal{U}}(A_i)$ are the join indicator variables that are ancestors of A_i in $\mathcal{U}$. ∎

4 PRM Construction

Now that we have seen the different semantics for PRMs, we have left the question of how the models can be automatically constructed. It turns out that although the semantics for the models are very different, the learning algorithms are largely the same and are closely related to the work in learning Bayesian networks [10]. The input to the construction algorithm is the relational schema, including the possible foreign-key joins between tuples; and a database instance. Our goal is to find a probabilistic schema $\mathcal{PS}(\mathcal{S}, \mathbf{\Theta})$ that optimizing some scoring criteria.

We can set this problem up as an optimization problem. There are three important components that need to be defined: the **hypothesis space** which specifies which structures are candidate hypotheses that our learning algorithm can return; a **scoring function** that evaluates the "goodness" of different candidate hypotheses relative to the data; and the **search algorithm**, a procedure that searches the hypothesis space for a structure with a high score.

Hypothesis Space. A hypothesis $\mathcal{PS}$ specifies a set of parents for each attribute $R.A$. We must restrict attention to probabilistic schemas which will generate a consistent probability model for any skeleton or query we are likely to see. We can do this by constructing a dependency graph for the candidate structure and ensuring that the class graph is acyclic. We maintain the graph during learning, and consider only models whose dependency structure passes the appropriate test; see [4] for more details.

Scoring. The second key component is the ability to evaluate different structures in order to pick one that fits the data well. We adapt Bayesian *model selection* methods to our framework. Bayesian model selection utilizes a probabilistic scoring function. In line with the Bayesian philosophy, it ascribes a prior probability distribution over any aspect of the model about which we are uncertain. In this case, we have a prior $P(\mathcal{S})$ over structures, and a prior $P(\mathbf{\Theta} \mid \mathcal{S})$ over the parameters given each possible structure. The *Bayesian score* of a structure $\mathcal{S}$ is defined as the *posterior* probability of the structure given the data $\mathcal{D}$:

$$P(\mathcal{S} \mid \mathcal{D}) \propto P(\mathcal{D} \mid \mathcal{S})P(\mathcal{S})$$

where the denominator, which is the marginal probability $P(\mathcal{D})$ is a normalizing constant that does not change the relative rankings of different structures. This score is composed of two main parts: the prior probability of the structure, and the probability of the data given that structure. The marginal likelihood is a crucial component, which has the effect of penalizing models with a large number of parameters. Thus, this score automatically balances the complexity of the structure with its fit to the data. In the case where $\mathcal{D}$ is a complete assignment, and we make certain reasonable assumptions about the structure prior, there is a closed form solution for the score.

Search Algorithm. The simplest heuristic search algorithm is greedy hill-climbing search, using our score as a metric. We maintain our current candidate structure and iteratively improve it. At each iteration, we consider a set of simple local transformations to that structure, score all of them, and pick the one with highest score. We restrict attention to simple transformations such as adding, deleting or reversing an edge. We stop when we have reached a local optimum in the model space. We can make our search procedure more robust by incorporating any of the standard extensions to greedy search such as random restarts, maintaining a tabu list or using a simulated annealing form of search algorithm. While this is not guaranteed to find the optimal solution, it is quite efficient and works surprisingly well in practice.

5 Conclusion

We have presented two approaches to structured statistical models: a possible worlds semantics and a domain frequency semantics. Experimental results using these models have been presented elsewhere: we have shown that they are useful for classifying web pages and bibliographic citations [5], for discovering patterns in tuberculosis epidemiology [7]; and for selectivity estimation in a variety of real-world domains [8]. Here we have attempted to present a concise summary of the two approaches, leaving out some of the technical details, in an effort to make the methods more accessible and to highlight the commonalities and differences between the approaches, and contrast with some of the other approaches described in this special issue.

References

[1] F. Bacchus. On probability distributions over possible worlds. In *Proceedings of 4th Workshop on Uncertainty in Artificial Intelligence*, 1988.

[2] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, 1990.

[3] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1996.

[4] L. Getoor. *Learning Probabilistic Models of Relational Data*. PhD thesis, Stanford University, 2001.

[5] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models with link uncertainty. *Journal of Machine Learning Research*, 3:679—707, 2002.

[6] L. Getoor and J. Grant. PRL: A logical approach to probabilistic relational models. *Machine Learning Journal*, 26, 2006.

[7] L. Getoor, J. Rhee, D. Koller, and P. Small. Understanding tuberculosis epidemiology using probabilistic relational models. *AI in Medicine Journal*, 30:233—256, 2004.

[8] L. Getoor, B. Taskar, and D. Koller. Using probabilistic models for selectivity estimation. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2001.

[9] J. Halpern. An analysis of first-order logics for reasoning about probability. *Artificial Intelligence*, 46:311–350, 1990.

[10] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. MIT Press, 1998.

[11] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the 15th National Conference on Artificial Intelligence*, 1998.

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