Multi-Relational Data Mining using Probabilistic Models Research Summary

Lise Getoor

Computer Science Department
Stanford University
getoor@cs.stanford.edu

Abstract. We are often faced with the challenge of mining data represented in relational form. Unfortunately, most statistical learning methods work only with "flat" data representations. Thus, to apply these methods, we are forced to convert the data into a flat form, thereby not only losing its compact representation and structure but also potentially introducing statistical skew. These drawbacks severely limit the ability of current statistical methods to mine relational databases. Probabilistic models, in particular probabilistic relational models, allow us to represent a statistical model over a relational domain. These models can represent correlations between attributes within a single table, and between attributes in multiple tables, when these tables are related via foreign key joins. In previous work [4, 6, 8], we have developed algorithms for automatically constructing a probabilistic relational model directly from a relational database. We survey the results here and describe how the methods can be used to discover interesting dependencies the data. We show how this class of models and our construction algorithm are ideally suited to mining multi-relational data.

1 Introduction

A large portion of real-world data is stored in commercial relational database systems and we would like to discover interesting statistical correlations in the data. Often the data is stored in multiple tables and there are many-one or many-many relationships between rows in the tables. However, most statistical learning methods work only with "flat" data representations. Thus, to apply these methods, we are forced to convert the data into a flat form. This has several important implications. The inability to mine the database directly produces redundancy and potentially high consistency-maintenance overhead. Databases are typically normalized and flattening the data introduces duplication and we lose our compact representation. But more importantly, we also potentially introduce statistical skew when we flatten the data. These drawbacks severely limit the ability of current statistical methods to mine relational databases.

Probabilistic models, in particular probabilistic relational models, allow us to represent a statistical model over a relational domain. We have developed a learning algorithm for automatically constructing a probabilistic relational model directly from a relational database that avoids flattening the data and avoids making inappropriate independence assumptions. We have applied this algorithm to several

medium-sized real-world databases. Not only is the running time of the learning algorithm is quite reasonable in practice, but more importantly, domain experts found the models our algorithm produced both easy to interpret and compelling. In short, this class of models and our construction algorithm are ideally suited to mining relational data.
This paper is a summary of previous work [4, 6, 5, 8].

2 Probabilistic Relational Models

Probabilistic relational models (PRMs), first introduced by [11], are an extension of Bayesian networks to relational domains. A PRM describes a template for a probability distribution over a database. The template includes a relational component, that describes the relational schema for the domain, and a probabilistic component, that describes the probabilistic dependencies that hold in the domain. A PRM, together with a particular universe of objects, defines a probability distribution over the attributes of the objects and the relations that hold between them. The relational component describes entities in the model, attributes of each entity, and references from one entity to another. The probabilistic component describes dependencies among attributes, both within the same entity and between attributes in related entities. An edge between attributes represents a probabilistic dependence of one attribute on the other attribute. In addition, the probabilistic component can also be used to model uncertainty over object references.

2.1 Relational Language

A schema for a relational model describes a set of *tables*, $\mathcal{X} = \{X_1, \ldots, X_n\}$. Each table has at least one key attribute, denoted X.K. The other attributes (or columns) in the table are either *descriptive attributes* or *foreign key attributes* (key attributes of another table) [1]. As an example, Figure 1 shows a PRM for a database from a tuberculosis (TB) clinic. The database contains information about TB patients, the TB strains that have been seen in the population and people who have been in contact with the patients. In this case, the schema for the TB PRM has three tables: Patient, Strain and Contact.
A descriptive attribute A of table X is denoted $X.A$, and its domain of values is denoted $\mathcal{V}(X.A)$. We assume that domain sizes are finite. For example, the Patient table has descriptive attributes such as *Gender*, *Race* and *Hiv*. The descriptive attributes are shown in ovals Figure 1. The domain of Patient.*Hiv* might be $\{$*negative, positive, not-tested*$\}$.
We use a similar notation, $X.F$, to refer to a foreign key of X. The *domain type* $\mathrm{Dom}[F]$ is X and F is typed, i.e., the schema specifies the table the foreign key is referencing. If F refers to tuples of table Y, then the *range type* $\mathrm{Range}[F]$ is Y. In our TB example, the class Patient has foreign key *Strain* linking a patient to the strain with which they are infected. In addition, the Contact has a foreign key *Patient* that refers to the patient that named this contact (The foreign keys are not shown in Figure 1).

[1] Foreign keys can be constructed from multiple attributes; here for simplicity we describe them as a single attribute

For each foreign key attribute $X.F$, we can define an *inverse* F^{-1}. Let F refer to tuples in table Y. For $y \in Y$, $F^{-1}(y)$ returns the set of tuples in X that refer to y. More precisely,

$$F^{-1}(y) = \{x | x.F = y\}$$

For example the inverse of Contact.*Patient* returns a patient's set of contacts.
Finally, we compose object references by defining the notion of a *reference chain*. This allows us to define functions from objects to other objects to which they are indirectly related. A *reference chain* $\tau = F_1, \ldots, F_k$ is a sequence of foreign keys (inverse or otherwise) such that for all i, $\mathrm{Range}[F_i] = \mathrm{Dom}[F_{i+1}]$.

2.2 Schema Instantiation

An *instance* $\mathcal{I}$ of a schema is simply a standard relational logic interpretation of this vocabulary. It specifies: the set of tuples x in each table; a value for each attribute $x.A$ (in the appropriate domain); and a value for each foreign key attribute $x.F$, which is the key of a tuple in the appropriate range type. For each tuple x in the instance and each of its attributes A, we use $\mathcal{I}_{x.A}$ to denote the value of $x.A$ in $\mathcal{I}$.
A *relational skeleton* σ_r of a relational schema is a partial specification of an instance of the schema. It specifies the set of tuples $\sigma(X_i)$ for each table and the relations that hold between the tuples (e.g., the values for the foreign keys). However, it leaves the values of the descriptive attributes of the tuples unspecified. The relational skeleton defines the random variables in our domain; we have a random variable for each attribute of each tuple in the skeleton. A PRM then specifies a probability distribution over *completions* $\mathcal{I}$ of the skeleton.

2.3 Probabilistic Model

A PRM defines a probability distribution over a set of instances of a schema. Most simply, we assume that the set of tuples and the relations between them are fixed, i.e., external to the probabilistic model. Then, the PRM defines only a probability distribution over the attributes of the tuples in the model.
The probabilistic model consists of two components: the qualitative dependency structure, $\mathcal{S}$, and the parameters associated with it, $\theta_{\mathcal{S}}$. The dependency structure is defined by associating with each attribute $X.A$ a set of *parents* $\mathrm{Pa}(X.A)$. Intuitively, the parents are attributes that are "direct influences" on $X.A$. In Figure 1, the arrows define the dependency structure.
We distinguish between two types of formal parents. The attribute $X.A$ can depend on another probabilistic attribute B of X. This formal dependence induces a corresponding dependency for individual tuples: for any tuple x in $\sigma(X)$, $x.A$ will depend probabilistically on $x.B$. The attribute $X.A$ can also depend on attributes of related tuples $X.\tau.B$, where τ is a reference chain. To understand the semantics of this formal dependence for an individual tuple x, note that $x.\tau$ represents the *set* of tuples that are referenced by x. Except in cases where the reference chain is guaranteed to be single-valued, we must specify the probabilistic dependence of $x.A$ on the multi-set $\{y.B : y \in x.\tau\}$. In order to represent this dependency compactly we will allow $x.A$ to depend probabilistically on some aggregate property of this multi-set. There are many natural and useful notions of aggregation: the mode of the set (most frequently occurring value); mean value

of the set (if values are numerical); median, maximum, or minimum (if values are ordered); cardinality of the set; etc.

Given a set of parents $\text{Pa}(X.A)$ for $X.A$, we can define a local probability model for $X.A$. We associate $X.A$ with a *conditional probability distribution* that specifies $P(X.A \mid \text{Pa}(X.A))$. Here the CPDs are multinomial distributions, represented either as tables or trees. Each CPD specifies for each possible instantiation of the parents, the distribution over the child's value. Let $\mathbf{U}$ be the set of parents of $X.A$, $\mathbf{U} = \text{Pa}(X.A)$. Each of these parents U_i — whether a simple attribute in the same relation or an aggregate of a set of related tuples — has a set of values $\mathcal{V}(U_i)$ in some ground type. For each tuple of values $\mathbf{u} \in \mathcal{V}(\mathbf{U})$, we specify a distribution $P(X.A \mid \mathbf{u})$ over $\mathcal{V}(X.A)$. This entire set of parameters comprises $\theta_{\mathcal{S}}$.

Definition 1. *A* probabilistic relational model (PRM) *Π for a relational schema $\mathcal{R}$ is defined as follows. For each $X \in \mathcal{R}$ and each descriptive attribute A of X, we have:*

- *a set of* parents *$Pa(X.A) = \{U_1, \ldots, U_l\}$, where each U_i has the form $X.B$ or $X.\tau.B$, where τ is a reference chain;*
- *a* conditional probability distribution (CPD) *that represents $P_\Pi(X.A \mid Pa(X.A))$.*

∎

2.4 PRM semantics

Given any skeleton, we have a set of random variables of interest: the attributes $x.A$ of the tuples in the skeleton. The PRM specifies a probability distribution over the possible joint assignments of values to these random variables. As with Bayesian networks, the joint distribution over these assignments can be factored. That is, we take the product, over all $x.A$, of the probability in the CPD of the specific value assigned by the instance to the attribute given the values assigned to its parents. Formally, this is written as follows:

$$\begin{aligned} P(\mathcal{I} \mid \sigma, \mathcal{S}, \theta_{\mathcal{S}}) &= \prod_{x \in \sigma} \prod_{A \in \mathcal{A}(x)} P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\text{Pa}(x.A)}) \\ &= \prod_{X_i} \prod_{A \in \mathcal{A}(X_i)} \prod_{x \in \sigma(X_i)} P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\text{Pa}(x.A)}) \end{aligned} \tag{1}$$

This expression is very similar to the chain rule for Bayesian networks. There are two primary differences. First, our random variables are the attributes of a set of tuples. Second, the set of parents of a random variable vary according to the relational context of the tuple— the set of tuples to which it is related.

As in any definition of this type, we have to take care that the resulting function from instances to numbers does indeed define a *coherent* probability distribution, i.e., where 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 PRMs as well, see [4, 5] for details.

3 Learning PRMs

In order to learn a PRM from an existing database, we adapt and extend the machinery that has been developed over the years for learning Bayesian networks from data [2, 9] to the task of learning PRMs from structured data [4, 6]. In the learning problem, our input contains a relational schema that specifies the basic vocabulary in the domain — the set of tables, the attributes of each table and the possible foreign-key joins between tuples in the different tables Our training data consists of a fully specified instance of that schema stored in a database.

There are two components of the learning task: parameter estimation and structure learning. In the parameter estimation task, we assume that the qualitative dependency structure of the PRM is known; i.e., the input consists of the schema and training database (as above), as well as a qualitative dependency structure $\mathcal{S}$. The learning task is only to fill in the parameters that define the CPDs of the attributes. In the structure learning task, we must discover the dependency structure $\mathcal{S}$ as well. We discuss each of these problems in turn.

3.1 Parameter Estimation

Here the dependency structure is known, e.g., we are given the structure $\mathcal{S}$ that determines the set of parents for each attribute, and our task is to learn the parameters $\theta_{\mathcal{S}}$ that define the CPDs for this structure. The key ingredient in parameter estimation is the *likelihood function*, the probability of the data given the model. This function measures the extent to which the parameters provide a good explanation of the data. Intuitively, the higher the probability of the data given the model, the better the ability of the model to predict the data. The likelihood of a parameter set is defined to be the probability of the data given the model: $L(\theta_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) = P(\mathcal{I} \mid \sigma, \mathcal{S}, \theta_{\mathcal{S}})$.

As in many cases, it is more convenient to work with the logarithm of this function:

$$\begin{aligned} l(\theta_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) &= \log P(\mathcal{I} \mid \sigma, \mathcal{S}, \theta_{\mathcal{S}}) \\ &= \sum_{X_i} \sum_{A \in \mathcal{A}(X_i)} \left[\sum_{x \in \sigma(X_i)} \log P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\mathrm{Pa}(x.A)}) \right]. \end{aligned} \quad (2)$$

The key insight is that this equation is very similar to the log-likelihood of data given a Bayesian network [9]. In fact, it is the likelihood function of the Bayesian network induced by the structure given the skeleton: the network with a random variable for each attribute of each tuple $x.A$, and the dependency model induced by $\mathcal{S}$ and σ, as discussed in Section 2.4. The only difference from standard Bayesian network parameter estimation is that parameters for different nodes in the network — those corresponding to the $x.A$ for different tuples x from the same class — are forced to be identical. This similarity allows us to use the well-understood theory of learning from Bayesian networks.

Consider the task of performing *maximum likelihood* parameter estimation. Here, our goal is to find the parameter setting $\theta_{\mathcal{S}}$ that maximizes the likelihood $L(\theta_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S})$ for a given $\mathcal{I}$, σ and $\mathcal{S}$. Thus, the maximum likelihood model is the model that best predicts the training data. This estimation is simplified by the *decomposition* of log-likelihood function into a summation of terms corresponding to the

various attributes of the different classes. Each of the terms in the square brackets in (2) can be maximized independently of the rest. Hence, maximum likelihood estimation reduces to independent maximization problems, one for each CPD. In fact, a little further work reduces Eq. (2) even further, to a sum of terms, one for each multinomial distribution $\theta_{X.A|\mathbf{u}}$. Furthermore, there is a closed form solution for the parameter estimates. In addition, while we do not describe the details here, we can take a *Bayesian approach* to parameter estimation by incorporating parameter priors. For an appropriate form of the prior and by making standard assumptions, we can also get a closed form solution for the estimates.

3.2 Structure Learning

We now move to the more challenging problem of learning a dependency structure automatically. The main problem here is finding a good dependency structure among the huge number of many possible ones. As in most learning algorithms, 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. We discuss each of these in turn.

Hypothesis Space Fundamentally, our hypothesis space is determined by our representation language: a hypothesis specifies a set of parents for each attribute $X.A$. We must, however, restrict our hypothesis space to ensure that the structure we are learning is a legal one. We are learning our model based on one training database, but would like to apply it in other settings, with potentially very different relational structure. We want to ensure that the structure we are learning will generate a consistent probability model for any skeleton we are likely to see. We can do this by constructing a class dependency graph for the candidate PRM 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, 5] for more details.

Scoring Function 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 $P(\theta_{\mathcal{S}} \mid \mathcal{S})$ 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 $\mathcal{I}$. Formally, using Bayes rule, we have that:

$$P(\mathcal{S} \mid \mathcal{I}, \sigma) \propto P(\mathcal{I} \mid \mathcal{S}, \sigma) P(\mathcal{S} \mid \sigma)$$

where the denominator, which is the marginal probability $P(\mathcal{I} \mid \sigma)$ 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 $\mathcal{I}$ 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 Now that we have a hypothesis space and a scoring function that allows us to evaluate different hypotheses, we need only provide a procedure for finding a high-scoring hypothesis in our space. For Bayesian networks, we know that the task of finding the highest scoring network is NP-hard [1]. As PRM learning is at least as hard as Bayesian network learning (a Bayesian network is simply a PRM with one table and no relations), we cannot hope to find an efficient procedure that always finds the highest scoring structure. Thus, we must resort to heuristic search.

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 can show that, as in Bayesian network learning, each of these local changes requires that we recompute only the contribution to the score for the portion of the structure that has changed in this step; this has a significant impact on the computational efficiency of the search algorithm. 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.

4 Experimental Results

We have run our algorithm on a variety of synthetic and real-world domains, see [6, 7, 5] for additional results. Here we describe one of the real-world domains.

Figure 1 shows the PRM learned for a database of epidemiological data for 1300 patients from the San Francisco tuberculosis (TB) clinic, and their 2300 contacts [17]. For the patient table, the schema contains demographic attributes such as age, gender, race, and place of birth (pob), as well as medical attributes such as HIV status, disease site (for TB), whether the X-ray results indicate cavitary infection, etc. The clustered attribute indicates whether this patient is a member of a cluster of TB outbreaks (a number of patients with the same TB strain) or whether the patient is infected with a unique strain of TB (likely to be an independent reactivation of latent infection and not part of a disease transmission chain). In addition, we have a strain table representing the different TB strains that have been identified among the patient population. The strain table has attributes for drug susceptibility results for streptomycin (str), isoniazid (inh), rifampin (rif), ethambutol (emb), and pyrazinamide (pza). The fitness attribute reflects the relative prevalence of a particular strain of M. tuberculosis in the study population:

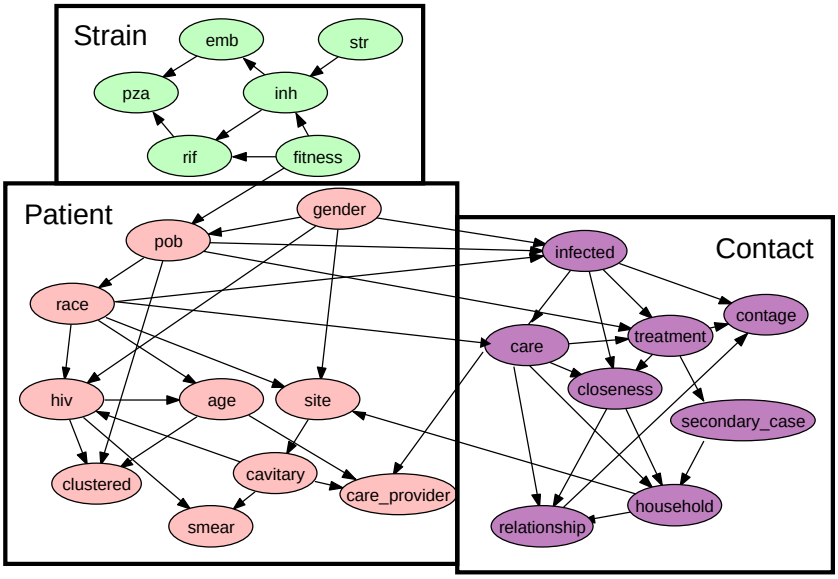

Fig. 1. The PRM structure learned for the TB domain.

strains identified only once were considered to have low fitness, strains isolated from 2–5 different patients were considered to have moderate fitness, and strains found in 3–6 individuals were considered to have high fitness. Each patient is also asked for a list of people with whom he has been in contact; the contact table has attributes that specify the relationship of contact (sibling, coworker, etc.), contact age, whether the contact is a household member, etc.; in addition, the type of diagnostic procedure that the contact undergoes (*care*) and the result of the diagnosis (*infected*) are also reported. The secondary_case attribute indicates whether the contact later becomes a patient in the clinic.

We learn a rich dependency structure both within entities and between attributes in different entities. Our domain experts found many of the dependencies to be quite reasonable, for example: the dependence of age at diagnosis on HIV status — typically, HIV-positive patients are younger, and are infected with TB as a result of AIDS; the dependence of the contact's age on the relationship of contact with the patient — contacts who are coworkers are likely to be younger than contacts who are parents and older than those who are school friends; or the dependence of HIV status on race — Asian patients are rarely HIV positive whereas white patients are much more likely to be HIV positive.

We also discovered dependencies that are clearly relational, and that would have been difficult to detect using a non-relational learning algorithm. For example there is a correlation between the race of the patient and the fitness of the strain. Patients who are Asian are more likely to be infected with a strain which is unique in the population (low fitness), whereas other ethnicities are more likely to have strains that recur in several patients (high fitness). The reason is that Asian pa-

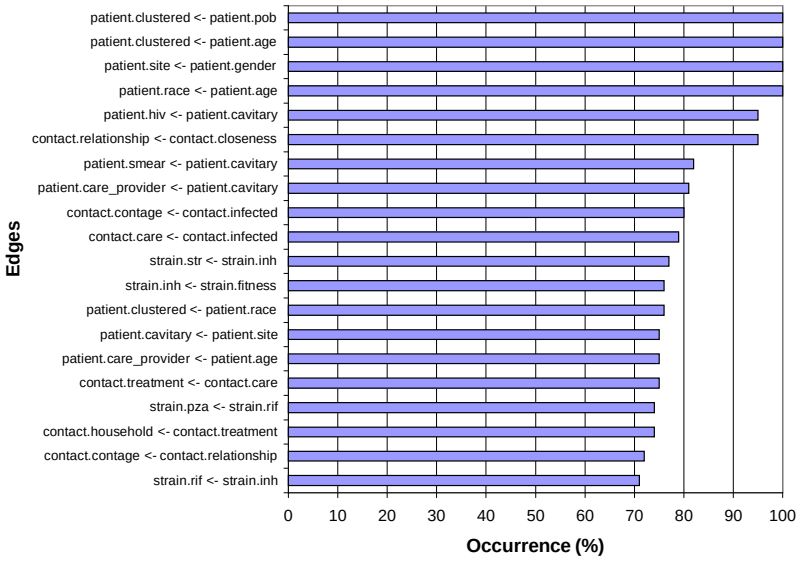

Fig. 2. The most commonly occurring edges in 100 learned models

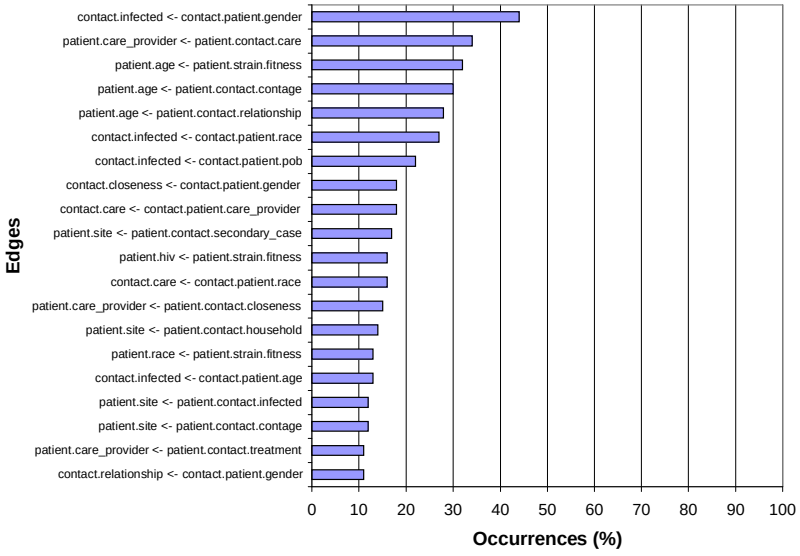

Fig. 3. The most commonly occurring inter-relational edges in 100 learned models .

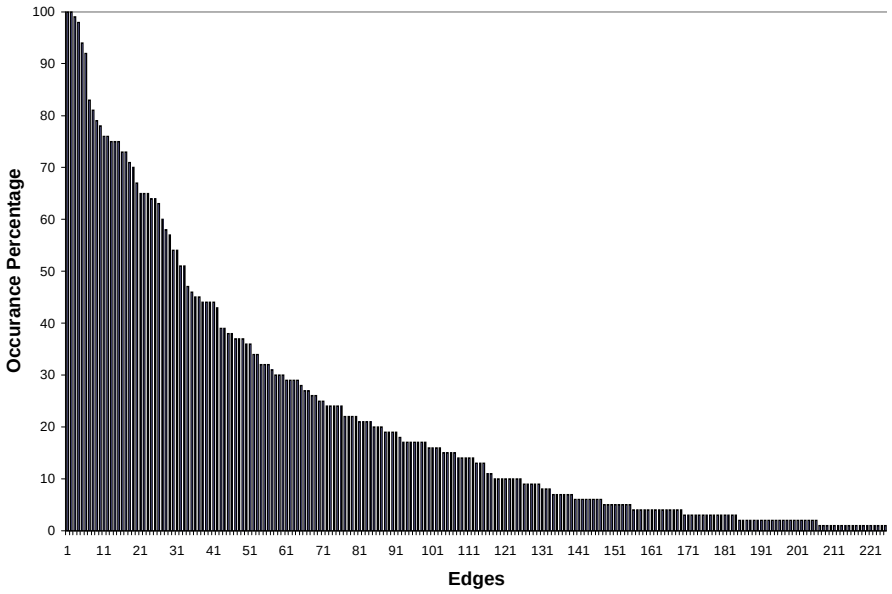

Fig. 4. The distribution of occurrences of edges in 100 learned models.

tients are more often immigrants, who immigrate to the U.S. with a new strain of TB, whereas others are often infected locally. Another interesting relational correlation is between the place of birth of the patient and whether or not their contacts become secondary cases. US-born patient's contacts are more likely to become secondary cases than foreign born patients. This is also explained by the immigrants likelihood of having a reactivated (non-infectious) TB case.

The model shown in Figure 1 shows the PRM learned using the greedy search algorithm. In order to get a better understanding of the robustness of the learned dependencies, we also ran experiments where we introduced randomness into the search. The learning algorithm for this set of experiments is a version of simulated annealing: in early phases of the search we are likely to take random steps (rather than the best step), but as the search proceeds (e.g., the temperature cools) we are less likely to take random steps and more likely to take the best greedy step. Figure 4 shows the distribution of occurrences of edges in 100 learned models. We see that of the possible 310 possible edges that can occur in our models, we see about two-thirds of them occurring at least once[2]. Figure 2 shows the most frequently occurring edges. Figure 3 shows the most frequently occurring edges between attributes in different relations. It is not surprising that the multi-relational edges are less frequent. First, intuitively attributes of the same object or tuple are more tightly coupled and thus more likely to have correlations. Sec-

[2] The models learned had tree structured CPDs; among other things this allows finer-grain dependencies to be learned. If we restrict attention to table CPDs, then in 100 runs only about a third of the possible edges occur.

ond, by the nature of our phased search, we will uncover dependencies between attributes within the same table before we consider dependencies between attributes in different tables, thus we will only add the latter to our model in the case where the correlation has not already been explained.
There are several caveats to be made about looking at these numbers. First, many dependency structures are *score equivalent*, so that we can not always distinguish between a model that has for example an edge going one direction versus a model that has the same edge directed in the opposite direction. Second, in making inferences the path of influence between two variables is what really matters, not just whether they are directly connected. Nevertheless, these experiments give us some additional insight into the domain.
Looking at the edge occurrences supports our earlier discoveries. For example, 50 of the models learned have an edge going from *hiv* to *age* and 45 of the models have an edge going from *age* to *hiv*; 72 of the models have an edge between contact relationship and contact age and 28 have the same edge in the reverse direction; and 69 of the learned models have an edge from *race* to *hiv* while 27 have the reversed edge. For the multi-relational dependencies, the most common correlations between strain fitness and patient attributes are on the attributes *race*, *pob* and *age*

5 Related Work

Two recent developments within the ILP community are related to PRMs: stochastic logic programs (SLPs)[12, 3] and Bayesian logic programs (BLPs)[10]. The semantics for these two approaches are quite different, with the BLP semantics being the closest to PRMs. An SLP defines a sampling distribution over logic programming proofs; as a consequence, it induces a probability distribution over the possible ground facts for a given predicate. On the other hand, a BLP consists of a set of rules, along with conditional probabilities and a combination rule; following the approach of knowledge-based model construction [16], the BLP essentially specifies a propositional Bayesian network. This approach is very similar to the probabilistic logic programs of [14, 15].
Learning algorithms for these approaches are being developed. Methods for learning SLPs are described in [13]. A maximum likelihood approach is taken for parameter estimation for an SLP which is based on maximizing the posterior probability of the program. The task of learning the structure of an SLP is quite different from learning a PRM structure and is based on more traditional ILP approaches. On the other hand, while BLPs are more closely related to PRMs, and methods for learning BLPs have been suggested in [10], learning algorithms have not yet been developed. Methods for learning PRMs may be applicable to learning BLPs.

6 Conclusion

It is important for us to develop sound statistical methods for directly mining multi-relational data stored in a relational database. Probabilistic relational models are one approach for achieving this goal. PRMs offer the advantage of having semantics based on well-founded statistical principles developed for Bayesian

networks and graphical models. The learning algorithms query the database for statistics directly and do not require the data to be extracted from the database before the learning algorithm can be applied. In addition, the learning algorithms are data driven and do not require the user to postulate individual hypothesis. And the learned PRMs give a full model of the joint domain and thus are easier to interpret than a collection of rules. In summary, PRMs are an important new tool for multi-relational data mining.

References

[1] D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer Verlag, 1996.

[2] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[3] J. Cussens. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the Fifteenth Annual Conference on Uncertainty*, pages 126–133, Stockholm, Sweden, 1999. Morgan Kaufman.

[4] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1307, Stockholm, Sweden, 1999. Morgan Kaufman.

[5] L. Getoor. *Learning Statistical Models from Relational Data*. PhD thesis, Stanford University, 2001. to appear.

[6] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*. Springer-Verlag, 2001.

[7] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proceedings of the International Conference on Machine Learning*, Williamstown, MA, 2001. Morgan Kaufman.

[8] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 461–472, Santa Barbara, CA, 2001. ACM Press.

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

[10] K. Kersting, L. de Raedt, and S. Kramer. Interpreting Bayesian logic programs. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 29–35. AAAI Press, 2000.

[11] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 580–587, Madison, WI, 1998. AAAI Press.

[12] S.H. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, Amsterdam, 1996.

[13] S.H. Muggleton. Learning stochastic logic programs. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 36–41. AAAI Press, 2000.

[14] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147– 177, 1996.

[15] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

[16] M.P. Wellman, J.S. Breese, and R.P. Goldman. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35–53, 1992.

[17] M. Wilson, J. DeRisi, H. Kristensen, P. Imboden, S. Rane, P. Brown, and G. Schoolnik. Exploring drug-induced alterations in gene expression in Mycobacterium tuberculosis by microarray hybridization. In *Proceedings of the National Academy of Sciences*, 2000.