Learning Probabilistic Relational Models: A Simplified Framework, a Case Study, and a Package

L. H. Mormille, F. G. Cozman

Universidade de São Paulo, Brazil
luiz.mormille@usp.br — fgcozman@usp.br

Abstract. While most statistical learning methods are designed to work with data stored in a single table, many large datasets are stored in relational database systems. Probabilistic Relational Models (PRM) extend Bayesian networks by introducing relations and individuals, thus making it possible to represent information in a relational database. However, existing methods that learn PRMs from data face some nontrivial challenges in the way relations are extracted. We propose a novel approach to learn the structure of a PRM. We also describe a package in the R language to support our learning framework, and we apply it to a real, large scale scenario combining citizens, companies and location data.

Categories and Subject Descriptors: G.3 [**Probability and Statistic**]: Probabilistic algorithms; I.2.6 [**Artificial Intelligence**]: Learning

Keywords: relational models, PRM, Bayesian networks, machine learning

1. INTRODUCTION

Most large data collections are stored in relational database systems consisting of multiple tables; however, most data mining techniques are only applicable to a single table. Multi-relational data mining (MRDM) focuses on the search for patterns across multiple tables (relations) of a database [Džeroski 2003], and many techniques have been proposed in that context. Indeed, relational models lead to a deeper understanding of the relations held within domains, and can be used for exploratory analysis, predictions and complex inferences.

Despite the success of Bayesian networks in a wide variety of real-world and research applications, they cannot be used to model domains where we might encounter several entities in different configurations [Koller 1999]. Probabilistic Relational Models (PRMs) are an extension of Bayesian Networks, introducing the concepts of objects and its properties, and the relations held between them, specifying a template for a probability distribution [Getoor and Taskar 2007]. Thus, PRMs offer a rich relational structure, allowing a property of an object to depend probabilistically, not only on the properties of that given object, but also on properties of other related objects [Getoor et al. 1999].

However, learning a PRM from relational data is a more complex task than learning a Bayesian Network from "flat" data. There are three main difficulties that arises while learning a PRM. The first one is establishing what are the legal dependency structures for a given domain: we must avoid cycles in the structure. The second difficulty is how to score the possible legal structures. And the third challenge is to search for possible structures [Friedman et al. 1999].

Given the complications often faced while learning a PRM, this paper aims to: (1) propose a new method for learning probabilistic relational models; (2) apply it in a real large-scale problem; and (3) report on a software package we have built to apply our method. Taken together, these contributions

should be valuable to researchers and practitioners interested in dealing with large relational datasets.

Section 2 presents a quick summary on probabilistic relational models. Section 3 explains the novel method for PRM learning. Section 4 reports on the case study and an R package produced by the authors. And Section 5 briefly presents some conclusions.

2. THE RELATIONAL MODEL FRAMEWORK

This section offers a review of PRMs, their semantics and the challenges in learning them. We use, as a running example, the case study we examine later.

A relational domain is usually represented by distinct tables in a database containing attributes and entities. For instance, consider the city of Atibaia, a relatively small town in the state of São Paulo, Brazil. Suppose we are interested in information about its citizens, its companies and its census sectors (territorial units, in which the city is geographically divided), and that we have three different tables, each representing one of the three different *classes* in our domain, **Person**, **Company** and **Census_Sector**. As we indicate later, the goal of our case study is to learn a PRM structure and parameters with classes in our domain, so as to predict the social class of people in Atibaia.

The vocabulary of a relational model consists of a set of *classes* $X_1, ..., X_n$, and a set of *relations* $R_1, ..., R_m$. Every class in the domain has a set of *attributes* $\mathcal{A}(X_i)$, and every attribute $A_j \in \mathcal{A}(X_i)$ has a space of possible values $V(A_j)$, being that this vocabulary defines a *schema* for the relational model. One attribute A of a class X, is referred to as $X.A$. If $X.A$ has a value that is fully determined, such as a name or and identification number, it is labeled as a *fixed* attribute. The other attributes are *probabilistic* ones [Friedman et al. 1999]. The logical description of the domain is called *relational schema*, and it shows how different classes relate to each other, through what is called *reference slots*. The set of reference slots of X is denoted $R(X)$, while $X.\rho$ is used to refer to the reference slot ρ of X. A reference slot ρ in X can be interpreted as an attribute of X that is also a foreign key for another class. And an *instance* I of a schema is an interpretation of the relations between the classes in the domain, and it specifies, for a set of objects, a class, and a value for each of it's attributes. To continue our running example, Figure 1(a) shows the schema for our Atibaia domain.

A central concept in a relational model is the *relational skeleton* σ, defined as a partial specification of an instance of a schema [Friedman et al. 1999]. It specifies for a set of *objects* $O^{\sigma}(X_i)$, a class, the value of the fixed attributes within this objects, and the relations held between them, leaving only the probabilistic attributes unspecified. Figure 1(b) represents an example of a relational skeleton for our Atibaia domain.

In our domain of study, the **Person** class has, among others, the attribute *Social_class*, and that the value space for **Person**.*Social_class* is {*A, B, C, D*}. Also, the **Person**.*Census_Sector_ID* attribute is a reference slot of the **Person** class, with *range type* **Census_Sector**. The same applies for the class **Company**. Every object in **Census_Sector** is associated with n objects in **Person**, that are the citizens who live in the sector, and m objects in **Company**, that are the companies located inside the area comprised by the census sector.

A PRM consists of a dependency structure S, and the conditional probability distribution θ_S associated with the dependency structure. Just like a Bayesian network, the dependency structure of a PRM is defined by associating a set of parents $\text{Pa}(X.A)$ with each attribute $X.A$. However, in a PRM, an attribute $X.A$ can have as parent, either an intra-class attribute, denoted $X.B$, or an inter-class attribute, denoted as $X.\tau.B$, where τ is a *slot chain*, representing the set of objects that are τ-relatives of an object $x \in X$. When an object x relates with an object of another class through a slot chain, unless the relation is guaranteed to be singled-valued, $x.a$ depends on a set of objects $X.\tau.B$, and, when that occurs, a method for representing these complex dependencies is necessary. This is attained using an *aggregation* function. Some notable aggregation functions are the *mode*, if the attribute is

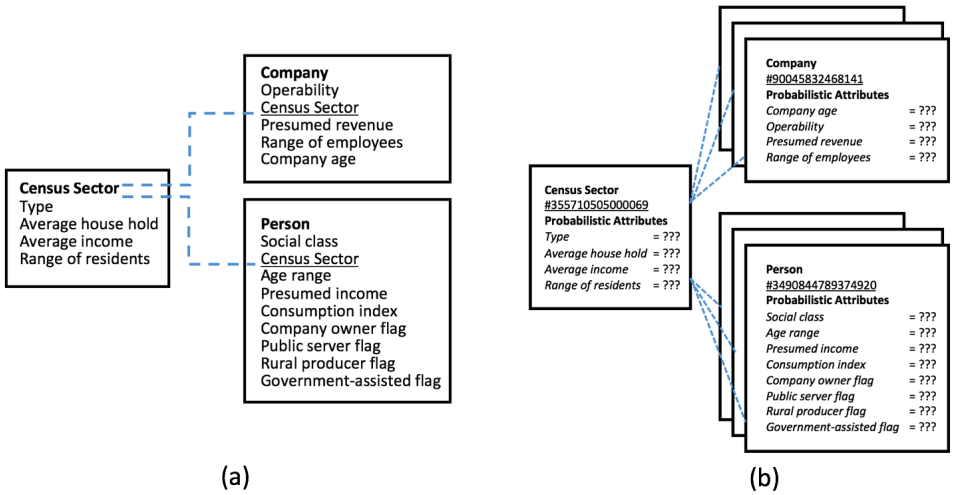

Fig. 1. (a) The relational schema for the Atibaia domain. (b) The relational skeleton for the Atibaia domain.

categorical, the *mean*, if the attribute is continuous, the *median*, *maximum* or *minimum*, if the attribute is ordered, or the *cardinality*. By using such functions, it is possible to return a summary of a multiset of values [Getoor and Taskar 2007]. An aggregation function γ takes a multiset of values, and returns a summary of it, allowing $X.A$ to have $\gamma(X.\tau.B)$ as a parent.

For our Atibaia city domain, a legal PRM structure, with the required aggregations indicated can be seen in Figure 2. Note that, since objects from either class **Company** and **Person** can only be associated with one object from the **Census_sector** class, no aggregations are required when an attribute on **Census_sector** is the parent of an attribute from either **Company** or **Person**.

Learning a PRM from data can be quite difficult. The most imposing challenge is to learn the dependency structure. First, making sure that all the dependencies established are acyclic is a nontrivial task. The best scenario is when the skeleton σ is acyclic. However, in some cases, if there is a cycle in the skeleton σ, one needs to guarantee that cycles do not happen at an object level. Also, evaluating and scoring concurrent structures is also challenging; the usual approach it to adapt Bayesian model selection. And finally, even after choosing a method to assure that a structure is legal and score it, one must go over candidate models, comparing them. The number of possible structures may be quite large, and the cost of the search operations is also high, making it necessary to associate the search method with an heuristic approach, limiting the space of possible structures, usually by associating a limited set o possible parents for each attribute.

3. THE PROPOSED METHOD

Our goal here is to search for a PRM, using some score typically used to learn Bayesian networks. As noted already, the search space is too big, so the process is quite complex and costly in practice. We propose an alternative approach to learning PRMs, by restricting the space of possible structures in ways that make sense for practical problems.

First, we do not allow the dependency structure to have cycles, not only at the attribute level, but at class level as well. For instance, structures such as $X.A \rightarrow Y.B \rightarrow X.C$ are not allowed. A second assumption is that we have *attributes of interest* that belong to a distinguished class, that we refer to as the *main class*. Note that, whenever an object $x \in X$ is linked to a set of objects $\{y_1, ..., y_i\} \in Y$ from another class, an attribute of the type $X.A$ can only have as parent an attribute of the type $Y.B$ by using an aggregation function γ. The space of possible structures will be then further reduced by not allowing any attributes from classes, other than the main class, to have inter-class attributes whose

slot chain is not singled-valued as parents, since it would require an aggregation. That determines that the only edges from aggregation functions will be directed to the main class.

Given a relational skeleton σ, all the required aggregations for our restricted space of structures will be computed in advance, and stored in a table denoted *master table*. At first, the master table will be a copy of the main class table. Then, for every object x_i in the main class X, the attributes of its τ-relative objects, whose links are given by the value of the reference slot $X.\rho$, will be aggregated using a proper aggregation function γ, and the result, will be returned to it's respective row in the master table. However, if the slot chain is guaranteed to be singled-valued, that is, if every object x_i in the main class relates to only one object of another class, the use of an aggregation function γ is unnecessary.

For instance, in the Atibaia domain, the main class is **Person**, because **Person**.*Social_class* is the variable of interest. That means the attributes from objects in **Company** and **Census_sector** that are τ-relatives to an object in **Person** will be aggregated. However, every object in the class **Person**, is associated with one, and only one, object in the class **Census_sector**, meaning that the slot chain that links them is singled-valued and the use a an aggregation function γ is unnecessary in that particular case. The values of attributes on **Census_sector** are simply replicated on the rows where their corresponding keys appears in the master table.

Now that the data is "flattened" into a master table, the challenges of defining a proper structure scoring function, choosing a search method for possible legal structures, and comparing them via score, can be solved by traditional Bayesian network approaches, and there are several popular algorithms that addresses these issues.

To make sure all adopted restrictions are enforced during learning, a *black list* of edges must be assembled. That list will specify which arrows cannot exist in the model, maintaining a coherent structure. Popular algorithms for Bayesian network learning typically allow for such lists of forbidden edges. For instance, in our Atibaia domain the restriction over the set of structures forbids attributes from class **Person** to parent attributes from **Census_sector** and **Company**, and also forbids attributes from class **Company** to parent any attribute on **Census_sector**.

Then, a Bayesian network structure, whose representation is a *directed acyclic graph* (DAG) describing a set of independencies [Koller and Friedman 2009], can be learned with any package of choice. For our domain of study, we used the *bnlearn* package; the search algorithm selected to complete this task was the *Hill Climbing*, and the selected scoring function was the *Dirichlet posterior density* based on *Jeffrey's prior*, returning a DAG analogous to a legal PRM structure.

For instance, in the Atibaia domain it is possible to observe on the resulting dependency structure that the only inter-class arrows respect those restrictions, thus generating a network isomorphic to a legal PRM structure for the domain, as shown in Figure 2. It is worth noting that, in this example, on the best structure learned by this method, all parents of our target variable are from foreign classes.

The PRM, just like a Bayesian network, is associated with a *conditional probability distribution* (CPD) or local probabilistic model. The CPD for X_i, given its parents $Pa(X_i)$ in the graph, is $P(X_i|Pa(X_i))$, and it captures the conditional probability of the random variable, given its parents in the graph [Koller et al. 2007].

As described by Getoor [2001], in the same way as with Bayesian networks, the joint distribution over these assignments can be factored by taking 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. The formal expression can written as follows:

$$P(I|\sigma, S, \theta_s) = \prod_{x \in \sigma} \prod_{A \in \mathcal{A}(x)} P(I_{x.A}|I_{Pa(x.A)}) = \prod_{X_i} \prod_{A \in \mathcal{A}(x)} \prod_{x \in \sigma(X_i)} P(I_{x.A}|I_{Pa(x.A)}).$$

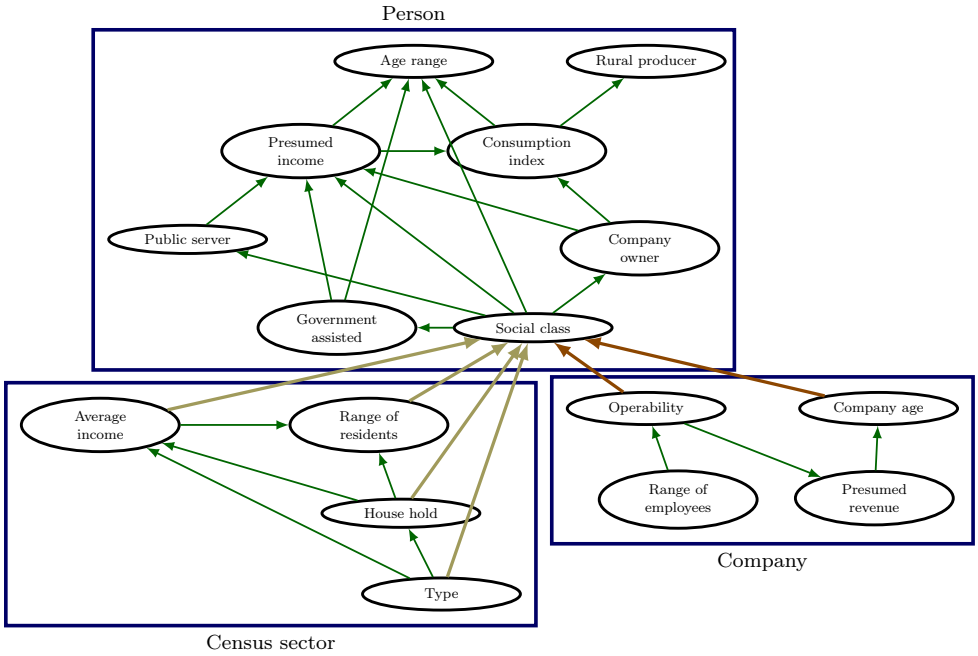

Fig. 2. The PRM structure, learned using the proposed method. The green arrows represent intra-class relations. The yellow arrows represent inter-class relations where the slot chain is singled valued. And the orange arrows represents the inter-class relations where an aggregation function was used (in our case, the mode).

4. A CASE STUDY, AND A PACKAGE FOR PRM LEARNING

In this section we apply our proposed method to a real large-scale problem. To do so, we have implemented a package in the R language. After we present some details of our case study, we describe our freely available package and apply it.

4.1 The case study

The domain of our case study is a small town named Atibaia, in the state of São Paulo, Brazil, and data was gathered from different types of objects within the city. The first class of objects is **Person**, and every object in that class, represents a citizen of Atibaia and its attributes. The second class in our domain is **Company**, and the objects of that class, are the business located in Atibaia. And the third class represents small territorial units that comprise the city of Atibaia, named **Census_sector**. It can be noted that, every object in **Census_ sector** can have n people related to it, which are the citizens who lives inside that sector, and m related companies, which are the business located inside that sector. However, an object in **Person** and an object in **Company** can only be associated with one, and only one, census sector. The goal is to explore the relations between those classes, so as to create a model that uses their attributes and to infer the social class of objects in **Person**.

The data of the first two classes (**Person** and **Company**) was kindly provided by *Serasa Experian*. All the entities on both classes are fully anonymised, and no personally identifiable attribute was provided, only a *hashed key*, for every person or company on the tables. Another attribute, present in both these classes, is the Census Sector where a person lives, or a company is situated. That attribute is also a foreign key for the third class in our domain, the **Census_Sector**. The remaining attributes will be *probabilistic attributes*, and express information about distinct aspects of a given person or company. The data of the **Census_Sector** can be found at the *IBGE* (Brazilian Institute

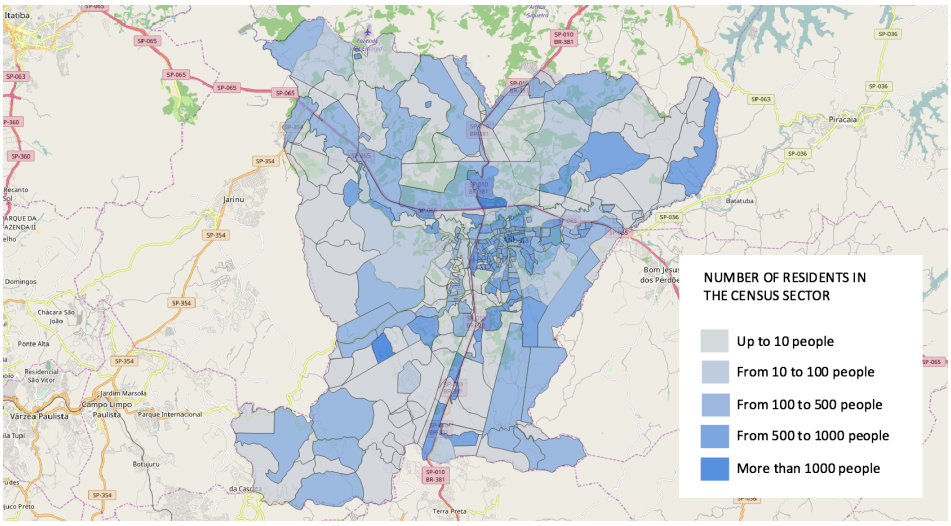

Fig. 3. All census sectors in Atibaia plotted in a map using the software *QGIS 2.18.7 Las Palmas*. The color of each sector represents the range of residents inside the sector.

of Geography and Statistics) website (`http://www.ibge.gov.br/home/`). More than a thousand different variables produced on the last census realized in Brazil, on year 2010, are available to the general public. The **Person** class, originally contained 110816 observations and 27 variables. However, a previous analysis detect that some of variables had high collinearity, so we discarded some of these variables. Other variables were also dropped, for they had a very high incidence of missing values. In the end, 10 variables of the class **Person** were selected for this study, and 8 of them were probabilistic attributes. The **Company** class, originally had 20162 observations and 9 variables. For the same issue of the **Person** class, only 6 variables were included in the analysis, and 4 of them were probabilistic attributes.

The city of Atibaia contains 327 census sectors, and a dataset with 5 variables was assembled with data available at the *IBGE* website. A map with the census sectors in Atibaia can be seen in Figure 3; the color of the layer represents the number of citizens living inside a given sector.

4.2 The package

Vanilla.prm is a package based on the above described method. The package supports domains with up to three classes with categorical attributes, and it performs all required aggregations using the function *mode*. The package is available in *github*, and install as any regular R package (using *devtools*).[1]

For processing, the user must ensure that all the variables on the tables are categorical, and that all *keys* and *foreign keys* are named the same way on every table on which it appears. If any variable does not fit this criteria, the R language provides tools to manipulate the data, in order to make it compliant with the requirements of the package.

Once the criteria for using the package are attended, the user should store the *column names* of all keys and foreign keys in a vector, which can be done, as an example, using the command:

```
R> key.names <- c("key1", "key2", "key3")
```

The result will be stored in the object *key.names*, and will be used as an argument of the following

[1]Further information about the package is available with its documentation, which can be found at `https://github.com/mormille/vanilla.prm`.

two functions. The master table for our domain is given by applying the function `relational.skeleton` on the three classes previously loaded, and the keys stored on the key.names vector. The first class to be assigned as an argument on the function, is the main class, and all the aggregations will be done on its direction. As result from running this function, the master table for our domain will be returned:

```
R> master_table <- relational.skeleton(main.class, class2, class3, key.names)
```

With the master table, resultant from the `relational.skeleton` function, the structure of a flat Bayesian network can be learned. And, for it to be a network analogous to a PRM, the black-list of arrows must also determined. The function `structure.learn` automatically generates the black-list of arrows. Then, it applies a search algorithm (hill climbing as default) to find optimal structures through a scoring function, and it can be run with the following command:

```
R> dag <- structure.learn(master_table, key.names, main.class, class2, class3)
```

The Bayesian information criterion (BIC) was set as the default scoring function, since it has performed better than other scores. Learning a Bayesian network is known to be NP-hard, with the number of possible structures growing exponentially with the number of variables, making it advantageous to use BIC [Liu et al. 2012]. The user, however, may choose other search algorithms and scoring functions, and transmit it as an argument of the `structure.learn` function.

By running the above function, a direct acyclic graph will be returned and plotted, respecting the restrictions that will allow it to be isomorphic to a PRM. In practice, it is also possible for the user of this method, to elicit a Bayesian network structure based on specialist knowledge over the domain of study, as long as the edges in the black list are not used.

The parameters over the dependency structure can be learned by using the function `parameters.learn`. It takes two arguments, the DAG, learned on the previous step, and the master table. It can be run with the following command:

```
R> fit <- parameter.learn(dag, master_table)
```

After these steps, a Bayesian network isomorphic to a PRM structure for a relational domain is completed, and can be used for inference on any attribute of the main class.

4.3 Applying the package to the case study

The performance of our approach in predicting the value of our target variable, **Person**.*Social_ class*, was tested by dividing the dataset in two parts. One part is the training set, used to fit the parameters of our model, and the remaining rows of the dataset are used for validation. To evaluate the performance of our model, a Bayesian network was leaned, using only the attributes in the **Person** class. To learn this Bayesian network, we also used *Hill climbing* as the search method and a *Dirichlet posterior density* based on *Jeffrey's prior* as the scoring function. The resulting dependency structure can be seen in Figure 4(a).

The Bayesian network learned from the class **Person** was also tested by dividing the data in two parts, one for training, and other for validation. The figure of merit used was the misclassification rate; the results displayed in Figure 4(b) show that, on average, the model learned from our method, correctly classifies 12% more cases than the Bayesian network from class **Person**.

5. CONCLUSIONS

Exploring the relational structure of databases allows a better understanding of the domain it represents. Reducing the attention to a subset of PRMs that are sensible in such a scenario could result in the restriction of good candidates. However, even though this approach may not cover all possible

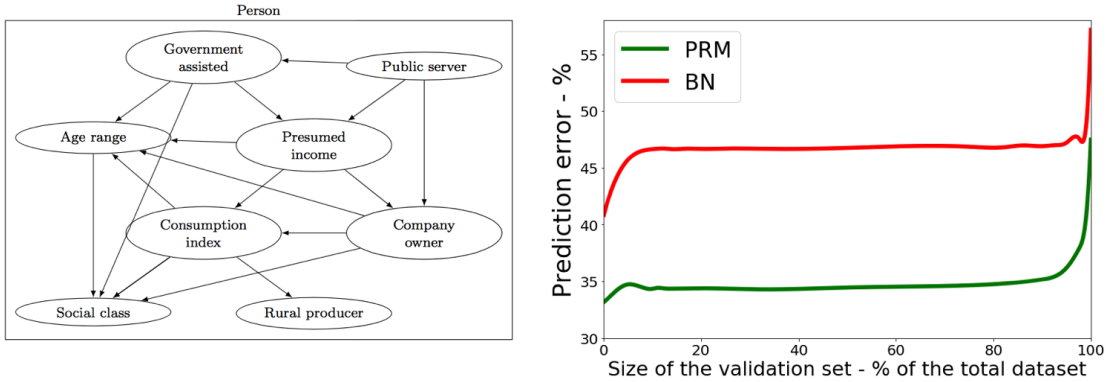

Fig. 4. (a) The resulting Bayesian network learned with the attributes from class **Person** (b) Misclassification rate of both models, given the size of the validation set.

cases, it will be useful whenever the variables of interest (for prediction) are in the same class of objects. In this work we have proposed a novel method for PRM learning, and we have applied it to a real large-scale problem, using a freely available R package that we have developed. The relational schema and skeleton for our case study are displayed in Fig. 1. The attribute of interest our model predicts is the **Person**.*Social_class*, and the value space for **Person**.*Social_class* is {*A, B, C, D*}. Using the proposed method, a PRM structure was learned (shown in Figure 2). The resulting relational model was confronted with a Bayesian network learned using attributes from **Person**, outperforming it, when comparing the misclassification rate of both models.

ACKNOWLEDGMENT

Most of the data used in this work was kindly provided by *Serasa Experian*, specially through the Latam Experian Datalab and the Marketing Services department. We thank Prof. Renato Vicente (*Instituto de Matemática e Estatística* of Universidade de São Paulo), and Glauber de Bona (*Escola Politécnica* of Universidade de São Paulo) for their help.

REFERENCES

Džeroski, S. Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter* 5 (1): 1–16, 2003.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. Learning probabilistic relational models. In *IJCAI*. Vol. 99. pp. 1300–1309, 1999.

Getoor, L. Multi-relational data mining using probabilistic relational models: research summary. In *In Proceedings of the First Workshop in Multi-relational Data Mining*. Citeseer, 2001.

Getoor, L., Sahami, M., et al. Using probabilistic relational models for collaborative filtering. In *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, 1999.

Getoor, L. and Taskar, B. *Introduction to statistical relational learning*. MIT press, 2007.

Koller, D. Probabilistic relational models. In *International Conference on Inductive Logic Programming*. Springer, pp. 3–13, 1999.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Koller, D., Friedman, N., Getoor, L., and Taskar, B. Graphical models in a nutshell. *Introduction to statistical relational learning*, 2007.

Liu, Z., Malone, B., and Yuan, C. Empirical evaluation of scoring functions for bayesian network model selection. *BMC bioinformatics* 13 (15): S14, 2012.