Automated Probabilistic Modeling for Relational Data

Sameer Singh
School of Computer Science
University of Massachusetts, Amherst MA
sameer@cs.umass.edu

Thore Graepel
Microsoft Research
Cambridge UK
thoreg@microsoft.com

ABSTRACT

Probabilistic graphical model representations of relational data provide a number of desired features, such as inference of missing values, detection of errors, visualization of data, and probabilistic answers to relational queries. However, adoption has been slow due to the high level of expertise expected both in probability and in the domain from the user. Instead of requiring a domain expert to specify the probabilistic dependencies of the data, we present an approach that uses the relational DB schema to automatically construct a Bayesian graphical model for a database. This resulting model contains customized distributions for the attributes, latent variables that cluster the records, and factors that reflect and represent the foreign key links, whilst allowing efficient inference. Experiments demonstrate the accuracy of the model and scalability of inference on synthetic and real-world data.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous

Keywords

Bayesian models;Probabilistic Modeling;Databases

1. INTRODUCTION

Relational databases have been the primary choice for management of structured knowledge for a majority of scientific and commercial applications, such as medicine, bioinformatics (protein interactions), commercial transactions, paper citation records, product ratings, and many more. These databases are often large, noisy, and contain missing values for many cells. Incorporating machine learning techniques into such databases can help in a number of ways: inference of missing values, detection of errors in the database (outliers), probabilistic responses to queries (taking into account the noisy and missing values), and visualization of the data.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2507828.

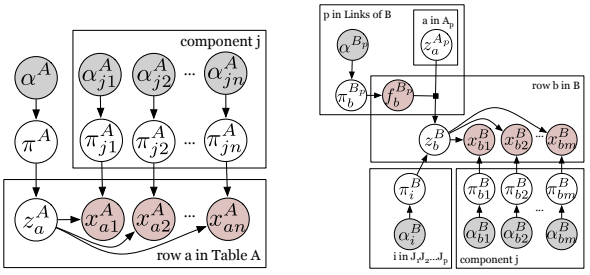
Traditional machine learning abstractions, such as classification and clustering, cannot be applied directly to achieve these goals in the structured data setting. However, an accurate probabilistic graphical model that represents the relational structure is appropriate; many of the desired objectives can be framed as inference (missing values), outlier detection (error checking), conditional marginalization (queries), and latent variables (visualization). To this end, there has been work along a number of avenues. Statistical relational learning (SRL) has focused on construction of graphical models specific to relational data[1]. The database community has proposed *probabilistic databases* as an alternative approach, i.e. creating tools to manage imprecise and uncertain data in a scalable manner, and to support efficient probabilistic inference for queries[2].

Unfortunately, a number of shortcomings restrict the application of these approaches in practice. First, these approaches are not fully-automated and require intervention by a domain expert. Second, many of these approaches assume that the domain expert also has a background in basic machine learning (probability distributions, features and sufficient statistics, graphical models, etc.). Further, approaches that do not require a user to specify the model (such as many probabilistic databases) often use simple models that fail to represent complex dependencies in the data. Third, many of these approaches only support a small subset from our desiderata, for example approaches in missing value prediction often do not support error-checking and visualization.

In this work, we propose an approach that automatically creates a probabilistic graphical model for any given database. Our approach is based on the observation that although the domain experts have trouble specifying the probabilistic model, they do spend considerable effort and expertise designing and normalizing the database schemata, providing foreign key relations that specify the dependencies. Using this schema, we automatically generate a customized fully-Bayesian, generative graphical model. Each table is modeled as a mixture consisting of a latent component variable for each row. We include variables and factors that cross multiple tables to represent dependencies between the latent components according to the foreign key relationships.

This approach provides a number of advantages. The underlying model allows prediction of foreign keys by including foreign key attributes as random variables (referential uncertainty). The latent variables represent clusterings over

[1]see Getoor and Taskar [4] for a comprehensive survey.
[2]see Dalvi et al. [2] for a comprehensive survey



(a) Table without Links (b) Table with Links

Figure 1: Building blocks of the graphical model: *red* variables may be observed or unobserved.

the rows of each table in a manner that is *aware* of the clusterings of linked tables; these clusterings can be used for data exploration and visualization. Inference is used to learn the parameters of the model, allowing predicted distributions over the missing values and error-checking via outlier detection (low-probability observations). Most importantly, since the approach is fully-automated, we enable the construction of probabilistic models automatically for a large number of existing databases without any manual intervention.

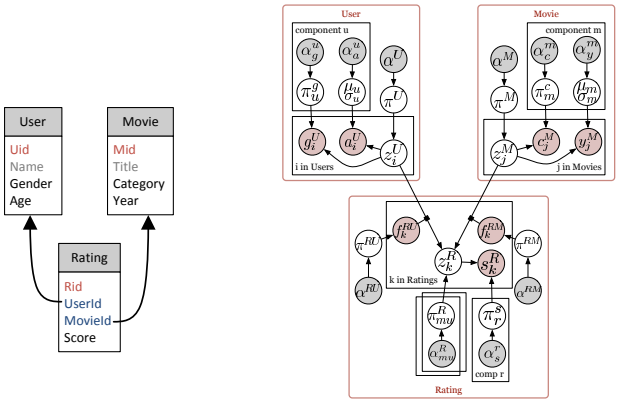
2. GRAPHICAL MODEL CONSTRUCTION

In this section we describe how we automatically create a Bayesian graphical model from a database schema.

Schema Description: Formally, a schema $\mathcal{S}$ consists of linked tables $A, B, C, \ldots$, where each table A contains n_A *value* attributes for each row a, denoted by $x^A_{a1}, x^A_{a2}, \ldots, x^A_{an_A}$. These attributes are typed by the standard data types (integer, double, categorical, string, etc.), and may be missing or observed. Each table A may also contain m_A foreign links to other tables $B_1, B_2, \ldots, B_{m_A}$; the p^{th} link in row a in A to a row in B_p is represented by $f^{AB_p}_a$. The links in the schema, when represented as edges $A \rightarrow B_p$, should form an acyclic graph[3]. Further, we filter out the attributes that should not be modeled, for example we automatically exclude string attributes that contain many unique values.

Single Table: We begin the description of the model by examining a single table A with attributes $\mathbf{x}^A$. We employ a *mixture model* for each table, wherein a mixture component is used to generate all the attributes $\mathbf{x}^A_a$ of row a. Specifically, the model for each table consists of J_A components, and each component j represents the distributions $\pi^A_{j1}, \pi^A_{j2}, \ldots, \pi^A_{jn_A}$ that correspond to each attribute of A. Each row also contains a latent variable $z^A_a \in \{1 \ldots J_A\}$ that indicates the component row a belongs to. We use this latent variable to select the distribution to generate the attributes of the row, i.e. we use π^A_{ji} to generate x^A_{ai} where $j = z^A_a$. The type of the latent distributions π^A_{ji} depends on the data type of the attribute (Gaussian for real-valued, Discrete for categorical-valued, and Bernoulli for Boolean-valued attributes), and each distribution is generated from its observed, conjugate prior α^A_{ji} (Gaussian for the mean, Gamma for the precision, Dirichlet for Discrete, and Beta for Bernoulli). If the table

[3]The violation to this (*circular reference*) is often discouraged, if not disallowed, in most DBMS. Further, such references can often be normalized by including additional tables.



(a) Schema (b) Generated Model

Figure 2: Example DB. Gray and red in (a) are not modeled. Data (black) is represented by g^U, a^U, c^M, y^M, s^R, and links (blue) by f^{RM}_k and f^{RU}_k.

A does not contain any links, the latent components z^A_a are generated from a latent Discrete distribution π^A of dimensionality J_A, with its observed Dirichlet prior α^A. We show such a model using the plate notation in Figure 1(a).

Foreign Links: Consider table B that contains foreign key attributes to tables $A_1, A_2, \ldots, A_{m_B}$. The data attributes of each row b of the table $\mathbf{x}^B_b$ are modeled as above using latent component variables z^B_b. The foreign key for each row b in table B to a table A_p is represented by $f^{BA_p}_b$ that indexes into a row in table A_p. Since we would like the links between rows to reflect the dependencies between the tables, we make the component indicator z^B_b depend on the component indicators of the foreign rows it links to $(z^{A_p}_k; k = f^{BA_p}_b)$. Specifically, instead of generating z^B from a single distribution as in the previous section, we use as many discrete distributions π^B_i as the product of the number of components in the linked tables $(J_{A_1} \times J_{A_2} \times \ldots J_{A_{m_B}})$, and *select* the corresponding distribution: $z^B_b \leftarrow \pi^B_i$, where $i \equiv \left\langle z^{A_1}_{(f^{BA_1}_b)}, z^{A_2}_{(f^{BA_2}_b)}, \ldots, z^{A_n}_{(f^{BA_n}_b)} \right\rangle$[4]. The uncertainty in foreign keys $\mathbf{f}^{B_p}$ is modeled as discrete distributions π^{B_p} (with conjugate Dirichlet prior α^{B_p}) to predict missing and incorrect foreign links. We show such a model in Figure 1(b).

Database Schema: An input schema $\mathcal{S}$ consists of tables, their attributes, and acyclic foreign key relations. We use the building blocks above to iteratively construct a model for a schema by applying the single-table models to all value attributes, simple priors for tables without any foreign keys, and using the foreign links to define the dependencies between the components for the tables with foreign keys. For an example, consider a simple schema consisting of three tables as shown in Figure 2(a). Figure 2(b) shows the generated model, where we use mixture models for Users and Movies, while the Rating table consists of additional dependencies of the component indicators across tables. Note that we use a slightly different notation in the example for clarity.

[4]As a consequence, the complexity of the model (number of parameters) scales only with the number of components, not with the number of rows in the parent tables.

Model Hyper-Parameters: A number of priors in the models need to be specified. We set these priors to be *uninformative*[5], however specifying the number of components for each table is crucial. Too many components result in slower inference, while too few components produce inaccurate models. We set these components heuristically, and will explore non-parametric approaches in future work.

3. INFERENCE

The model is generated as source code for Infer.NET [8], however other graphical model toolkits may also be used. Inference is performed using variational message passing [11].

Training and Predictions: During training, we *learn* the parameters of the model and use it to predict missing values. The complexity of each iteration of message passing is linear in the number of rows when all foreign keys are observed, since estimating the priors for foreign links is equivalent to counting. When not all of the links are observed, the complexity remains linear in the number of rows in tables that contain links, which are often much bigger than the tables they point to. Although the complexity is proportional to the product of the number of components in tables that are linked to (dimensionality of π_i^B in Figure 1(b)), in practice it is rare to find more than 3 tables linked from a single table. Marginal distributions computed during inference for each cell can be directly used to predict missing values (and confidence) and detect outliers in the observed cells. Further, latent variables $\mathbf{z}$ assign a component/cluster to each row; predictions include probabilistic assignments of each row to these clusters.

Querying the Model: The approach also supports a restricted set of queries over the trained model. Queries take the form of a set of rows for each table with missing values or foreign links. Using the learned distributions, inference estimates marginal posteriors over the query rows that are used to predict missing values and detect outliers in the query. Inference also provides clusters for each query row that may be used to discover similar rows in the existing data (such as other users that rate similarly). Query inference is efficient: linear in query size if links to the existing data are observed.

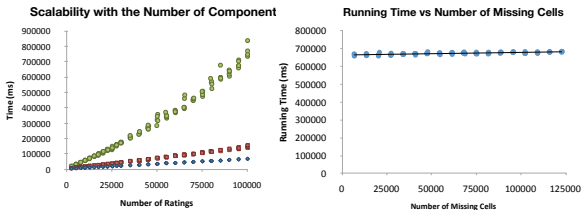
4. EXPERIMENTS

In this section, we present experiments to evaluate the accuracy, clustering quality, and scalability of the approach.

Synthetic User-Movies-Ratings: One typical approach to modeling values in a database is to use a single-table mixture model on the result of a join over all the tables. Unlike in our relational model, the dependencies across rows are lost in the join operation[6]. To evaluate its effect on accuracy, we compare these models by hiding a proportion of cells before performing the join. We create synthetic data for the schema in Figure 2(a), and perform inference to predict the values of the hidden cells. The prediction error for real-valued attributes is shown in Figure 3, demonstrating that the schema-based model is consistently more accurate and robust in the presence of missing cells. In particular, the rating scores are accurate even when half of the values are missing, while the competing approach (in *blue*) suffers from a high error even when only a few cells are missing.

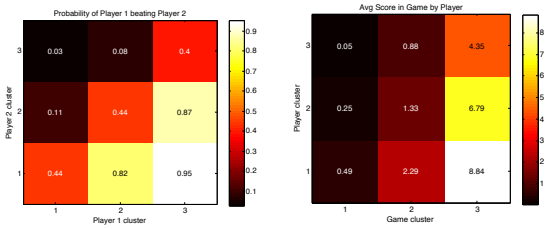
[5]We use $(\mu = 0, \sigma = 10^8)$ for Gaussian, $(k = 1, \theta = 10)$ for Gamma, $\alpha = 1$ for Discrete, and $(\alpha = 1, \beta = 1)$ for Beta.

[6]For example, age of a user may appear in multiple cells in the joined table, and thus the estimates may be different.



(a) Number of ratings (b) Number of missing cells

Figure 4: Time for Inference for MovieLens data

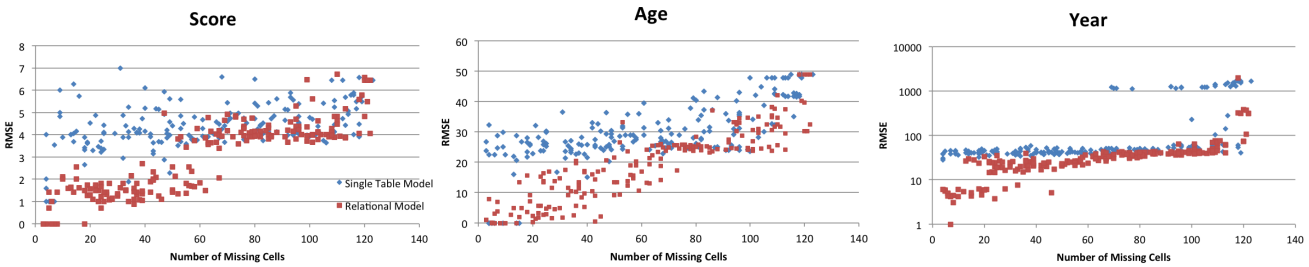


(a) 1 vs 1 (12k games) (b) Team (14k games)

Figure 5: Xbox Data: Visualizing players and games

MovieLens: We evaluate scalability on MovieLens (`http://www.grouplens.org/node/73`). The data is similar to User-Movie-Rating example with additional attributes, and consists of 943 users, 1682 movies, and 10^5 ratings. Since the number of rows in the *leaf* table is much greater than in other tables, we examine the scalability in terms of its size. The running time of inference shows a linear trend with data size in Figure 4(a). The trend in Figure 4(b) demonstrates that the number of missing values in the database has little effect on the running time, and thus can be applied to large databases with noisy and missing attributes.

Xbox: To perform a qualitative evaluation of the predicted clustering, we use Halo2 Xbox data [6]. The first dataset, 1-vs-1, consists of player Id table (with no attributes), and a table of match results that consists of links to two players and a Boolean result that is true if the first player was the winner. The generated model assigns each player to one of three components/clusters. We bin each of the matches (pairs of players) according to the clusters that the players belong to, and include the average result for each pair of clusters (winning probability), in Figure 5(a). The clusters separate the players fairly discriminatively in terms of winning probabilities. Further, the clusters correspond to bad, good, and excellent players respectively, demonstrating the utility of the latent clustering for predicting player skills without any domain-specific assumptions. We also apply our approach to the Xbox team games dataset that contains player, game (both only Ids) and score tables that represent the score attained by the player in a game. The clustering of players and games with the average scores in Figure 5(b) shows that the clusters can be viewed as (Good, Normal, Bad) over the players and (Difficult, Normal, Easy) over the games, respectively.



(a) Rating Score of a Movie (b) Age of a User (c) Release year of a Movie

Figure 3: Results on Synthetic Data: Comparison of the relational model (*in red*) with a single table model generated using a join (*in blue*) using RMSE on three real-valued attributes.

5. RELATED WORK

Relational Latent Variable Models: Our underlying model is most similar to the latent variable models for relational data [12, 7]. These approaches construct a generative model over *entities* and *relations*, where each member of these classes is associated with attributes and latent component indicators. A similar approach was also proposed for graphical data [1]. Although we have been inspired by these models, the relations that they support are restricted to presence/absence, while our approach, by representing uncertainty at the (finer) level of foreign key relationships, is not restricted to Boolean links. Even though these approaches are non-parametric, inference speed is impractical; inference for our model scales linearly in the size of the data.

Statistical Relational Learning (SRL): SRL has made significant strides in representing uncertain entity-relation data [4]. Friedman et al. [3] and Heckerman et al. [5] introduce Bayesian models for SRL, allowing informed domain experts to create probabilistic models. Taskar et al. [10] instead use undirected models, for which Neville and Jensen [9] introduce tractable learning. Our work differs in a number of aspects. First, we directly represent the schema and foreign keys, providing the flexibility to represent data that existing approaches cannot. Second, we create a joint model and use a well-understood inference algorithm that is accompanied by certain guarantees. We prefer Bayesian generative modeling due to its elegant adaptation to different domains; undirected models are often unreliable on small and/or sparse data. Third, since the schema captures much of the dependency structure, by using latent foreign links we create a customized model without making any other assumptions.

Probabilistic Databases: In recent years, there has been significant progress in *probabilistic databases*, i.e. tools to manage imprecise and uncertain data scalably, and to support efficient probabilistic inference for arbitrary SQL-like queries [2]. To attain this goal, many of these approaches make strong independence assumptions in the probabilistic model, resulting in lower accuracy. Others require the user to specify the underlying model, and perform efficient inference and learning over this model. We instead focus on automatically creating a joint, expressive model over all the data, and perform queries on a subset of SQL. Although filling missing values is related to uncertainty representation (as both require an underlying probabilistic model), by restricting our queries to a reasonable subset (that we feel is important), we are able to use the power of Bayesian models and inference without sacrificing efficiency or making strong independence assumptions. Further, we directly represents referential uncertainty (foreign key prediction), which is uncommon amongst the research in this area.

6. CONCLUSIONS

We suggest automatically compiling probabilistic graphical models from database schemata. This approach allows us to utilize the domain knowledge that went into the design of the database schema and potentially makes probabilistic graphical models directly available for a large fraction of the world's data. Inference on the compiled joint Bayesian model allows the prediction of the values of missing cells in the database, detect outliers, visualize clustering of the data, and to answer basic probabilistic relational queries.

References

[1] C. Böhm, G. Kasneci, and F. Naumann. Latent topics in graph-structured data. In *International Conference on Information and Knowledge Management (CIKM)*, 2012.

[2] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Comm. of the ACM*, 52(7):86–94, 2009.

[3] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[4] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[5] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research, 2004.

[6] R. Herbrich, T. Minka, and T. Graepel. Trueskill^{TM}: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems (NIPS)*, pages 569–576, 2007.

[7] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *American Association of Artificial intelligence (AAAI)*, pages 381–388, 2006.

[8] T. Minka, J. M. Winn, J. P. Guiver, and D. A. Knowles. Infer.NET 2.4, 2010. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

[9] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

[10] B. Taskar, A. Pieter, and D. Koller. Discriminative probabilistic models for relational data. In *Uncertainty in Artificial Intelligence (UAI)*, 2002.

[11] J. M. Winn. *Variational Message Passing and its Applications*. PhD thesis, Dept of Physics, University of Cambridge, 2003.

[12] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Infinite hidden relational models. In *Uncertainty in Artificial Intelligence (UAI)*, pages 544–551, 2006.