FACTORBASE : SQL for Multi-Relational Model Learning

Zhensong Qian*
School of Computing Science
Simon Fraser University
Vancouver-Burnaby, Canada
zqian@sfu.ca

Oliver Schulte
School of Computing Science
Simon Fraser University
Vancouver-Burnaby, Canada
oschulte.ca

Abstract

We describe FACTORBASE , a new framework that leverages a relational database management system (RDBMS) to support multi-relational graphical model learning. The basic insight behind our approach is that an RDBMS can be leveraged to manage not only big data, but also to manage big models [1, 2]: First, model structure and model parameters can be managed efficiently without having to be stored in main memory. Second, the Structured Query Language (SQL) supports constructing, storing, and transforming structured statistical objects. The FACTORBASE system uses SQL as a high-level scripting language for statistical-relational learning of a graphical model structure. Our implementation shows how the SQL constructs in FACTORBASE facilitate fast, modular, and reliable program development. Empirical evidence from six benchmark databases indicates that leveraging database system capabilities achieves scalable model structure learning.

1 Introduction

Multi-relational data have a complex structure, that integrates heterogeneous information about different types of entities (customers, products, factories etc.) and different types of relationships among these entities. A statistical-relational model provides an integrated statistical analysis of the heterogeneous and interdependent complex data resources maintained by the database system. Machine learning researchers have developed a number of formalisms for statistical-relational models that combine ideas from graphical models with first-order logic. These include Probabilistic Relational Models, Markov Logic Networks, and Probabilistic Soft Logic [3]. Statistical-relational models have achieved state-of-the-art performance in a number of application domains, such as natural language processing, ontology matching, information extraction, entity resolution, link-based clustering etc. [4, 2]. Database researchers have noted the usefulness of statistical-relational models for knowledge discovery and for representing uncertainty in (probabilistic) databases [5, 1, 6].

Multi-relational graphical model learning raises new challenges that require system support, such as:

1. A description language for specifying metadata about structured random variables.
2. Efficient mechanisms for constructing, storing, and transforming complex statistical objects, such as cross-table sufficient statistics, parameter estimates, and model selection scores.
3. Computing model prediction scores for relational test instances.

*This paper is based on material presented at the DSAA 2015 conference in October. Supported by a Discovery grant to Oliver Schulte from the Natural Sciences and Engineering Research Council of Canada, and by a grant to Zhensong Qian from the China Scholarship Council.

We describe the FACTORBASE system that leverages RDBMS capabilities to solve these system challenges. The name FACTORBASE indicates that our system supports learning a set of (par)-factors for a log-linear multi-relational model, typically represented in a graphical model [3]. FACTORBASE adopts a system architecture, developed by database researchers, where statistical models are stored as first-class citizens inside a relational database management system [1, 2]. Our system applies SQL as a scripting language to implement database services that provide system capabilities for relational learning. The theoretical basis for SQL is relational algebra [7]; FACTORBASE shows that relational algebra provides a unified language for both representing and computing with statistical-relational objects, much as linear algebra does for traditional single-table machine learning.

1.1 Related Work

Qian and Schulte discuss related work in detail [8]. The idea of managing big structured models inside an RDBMS, in addition to managing big data, has been developed in the database community [1, 2]. These systems focus on inference *given* a statistical-relational model, not on *learning* the model from the data stored in the RDBMS. Our FACTORBASE system complements the in-database probabilistic inference systems with an in-database probabilistic model learning system.

There are several previous systems that leverage RDBMS support for learning [9, 10], but they apply to traditional learning where the data are represented in a *single* table or data matrix. The novel contribution of FACTORBASE is supporting graphical model learning for *multi-relational* data stored in different interrelated tables. The Sindbad system [11] provides support for some multi-relational knowledge discovery tasks in an inductive database, but not for graphical model construction.

1.2 Evaluation

Our system is fully implemented and source code is available on-line [12]. We summarize the evaluation of FACTORBASE on six benchmark databases. For each benchmark database, the system applies the learn-and-join algorithm, a state-of-the-art SRL algorithm that constructs a statistical-relational Bayesian network model [13]. The learned Bayes net structure can be converted to a Markov Logic network structure or a set of clauses [14]. The same SQL scripts work for all benchmark databases, which demonstrates the generality of our approach.

Our experiments show that FACTORBASE pushes the scalability boundary: Learning scales to databases with over 10^6 records, compared to less than 10^5 for previous systems. At the same time it is able to discover more complex cross-table correlations than previous SRL systems. The scalability improvement is mainly due to the efficient computation and caching of sufficient statistics supported by SQL. Our experiments focus on two key services for an SRL client: (1) Computing and caching sufficient statistics, (2) computing model predictions on test instances. The system can handle as many as 15M sufficient statistics. SQL facilitates block-prediction for a set of test instances, which leads to a 10 to 100-fold speedup compared to a simple loop over test instances.

1.3 Benefits

We advocate using SQL as a high-level scripting language for statistical-relational learning, because of the following advantages.

1. Extensibility and modularity, which support rapid prototyping. Algorithm development can focus on statistical issues and rely on the RDBMS for data access and processing.
2. Increased scalability, in terms of both the size and the complexity of the statistical objects that can be handled.
3. Generality and portability: standardized database operations support "out-of-the-box" learning with a minimal need for user configuration.

2 Log-linear Template Models for Relational Data

FACTORBASE supports learning log-linear multi-relational graphical models based on par-factors. We briefly describe this model class; for more details see the survey of Kimmig *et al.*[3]. Par-factor

stands for "parametrized factor". A par factor represents an interaction among parametrized random variables, or par-RVs for short. We employ the following notation for par-RVs [3, 2.2.5]. Constants are expressed in lower-case, e.g. joe, and are used to represent entities. A type is associated with each entity, e.g. joe is a person. A first-order variable is also typed, e.g. $Person$ denotes some member of the class of persons. A functor f maps a tuples of entities to a value. We assume that the range of possible values is finite. A *term* is an expression of the form $f(\tau_1, \ldots, \tau_a)$ where each τ_i is either a constant or a first-order variable. If all of $\tau_1, \ldots, \tau_a$ are constants, $f(\tau_1, \ldots, \tau_a)$ is a *ground term* or random variable (RV), otherwise a *first-order term* or a **par-RV**. A par-RV is instantiated to an RV by grounding, i.e. substituting a constant of the appropriate domain for each first-order variable.

Examples. The standard single-table attribute-value representation is a special case where all functors map exactly one entity to an attribute value. For instance, in a single table $Student$, an attribute $gender$ can be represented as a function that takes as input a single student and returns W or M. A ground term such as $gender(joe)$ represents the gender of a specific individuals. A parametrized random variable $gender(\mathbb{S})$ is a template for a set of ground terms, as shown in the plate notation of Figure 3 [3]. A characteristic of relational data is that they include functions of more than one entity. For instance, $grade(Student, Course)$ is a par-RV based on a function that takes as input a student and a course and returns the grade of the student in that course.

A **par-factor** is a pair $\Phi = (\mathsf{A}, \phi)$, where A is a set of par-RVs, and ϕ is a function from the values of the par-RVs to the non-negative real numbers.[1] Intuitively, a grounding of a par-factor represents a set of random variables that interact with each other locally. SRL models use *parameter tying*, meaning that if two groundings of the same par-factor are assigned the same values, they return the same factor value. A set of parfactors $\mathcal{F}$ defines a joint probability distribution over the ground par-RVs as follows. Let $\mathcal{I}(\Phi_i)$ denote the set of *all* ground par-RVs in par-factor Φ_i. Let $\mathbf{x}$ be a joint assignment of values to all ground random variables. Notice that this assignment determines the values of all ground atoms. An assignment $\mathbf{X} = \mathbf{x}$ is therefore *equivalent to a single database instance*. The probability of a database instance is given by the log-linear equation [3, Eq.7]:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{\Phi_i \in \mathcal{F}} \prod_{\mathbf{A} \in \mathcal{I}(\Phi_i)} \phi_i(\mathbf{x}_\mathbf{A}) \tag{1}$$

where $\mathbf{x}_\mathbf{A}$ represents the values of those variables in $\mathbf{A}$ that are necessary to compute ϕ_i. Equation 1 can be evaluated, without enumerating the ground par-factors, as follows. (1) For each par-factor, for each possible assignment of values, find the number of ground factors with that assignment of values. (2) Raise the factor value for that assignment to the number of instantiating factors. (3) Multiply the exponentiated factor values together. The number (2) of ground factors with the same assignment of values is known as a **sufficient statistic**.

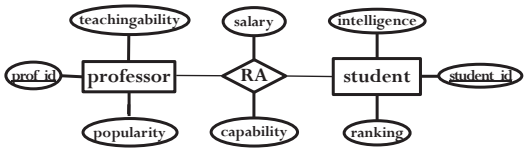


Figure 1: A relational ER Design for a university domain.

Student

s_id	intelligence	ranking
jack	3	1
kim	2	1
paul	1	2

(a)

RA

s_id	p_id	salary	capability
jack	oliver	high	3
kim	oliver	low	1
paul	jim	med	2
kim	david	high	2

(b)

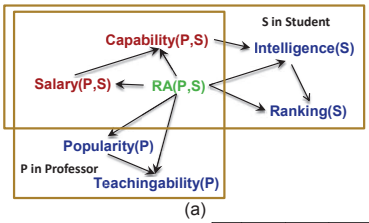
Professor

p_id	popularity	teachingability
jim	2	1
oliver	3	1
david	2	2

(c)

Figure 2: Database Table Instances: (a) $Student$, (b) RA, (c) $Professor$.

[1] A par-factor can also include constraints on possible groundings.



(a)

Capa(P,S)	RA(P,S)	Salary(P,S)	CP
n/a	F	n/a	1.00
4	T	high	0.45
5	T	high	0.36
3	T	high	0.18
3	T	low	0.20
2	T	low	0.40
1	T	low	0.40
2	T	med	0.22
3	T	med	0.44
1	T	med	0.33

(b)

Count	Capa(P,S)	RA(P,S)	Salary(P,S)
203	n/a	F	n/a
5	4	T	high
4	5	T	high
2	3	T	high
1	3	T	low
2	2	T	low
2	1	T	low
2	2	T	med
4	3	T	med
3	1	T	med

(c)

Figure 3: (a) Bayesian network for the University domain. We omit the *Registered* relationship for simplicity. The network was learned from the University dataset [12]. (b) Conditional Probability table $Capability(\mathbb{P},\mathbb{S})\_CPT$, for the node $Capability(\mathbb{P},\mathbb{S})$. Only value combinations that occur in the data are shown. This is an example of a factor table. (c) Contingency Table $Capability(\mathbb{P},\mathbb{S})\_CT$ for the node $Capability(\mathbb{P},\mathbb{S})$ and its parents. Both CP and CT tables are stored in an RDBMS.

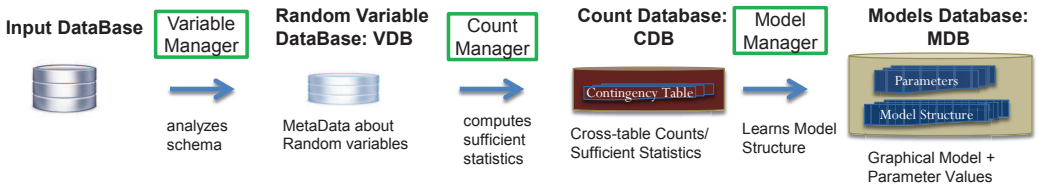


Figure 4: System Flow. All statistical objects are stored as first-class citizens in a DBMS. Objects on the left of an arrow are utilized for constructing objects on the right. Statistical objects are constructed and managed by different modules (boxes).

3 System Overview

Our system design represents statistical objects as relational tables, on a par with the original data tables, so that SQL can be used to manage them. Figure 4 represents key system components. The starting point is a multi-relational database containing the input data.

3.1 The Schema Analyzer

The schema analyzer is an SQL script that queries the system catalog table to define a default set of relational random variables (par-RVs) for statistical analysis [3]. The metadata include the domain of the par-RVs (possible values), and type information (possible arguments). The schema analyzer extracts metadata about the random variables from the database system catalog. The random variables and associated metadata are stored in the **random variable database** VDB. We highlight two features of the VDB component.

(i) The set of par-RVs and the associated metadata is constructed *automatically* from the input database. Thus FACTORBASE utilizes the data description resources of SQL to faciliate the "setup task" for relational learning [15].

(ii) Representing metadata explicitly offers two advantages. First, a user can easily edit the *VDB* to customize the learning behavior, for instance by deleting irrelevant par-RVs. Second, it is possible to export metadata from other formats to the *VDB* format. In this way FACTORBASE can serve as a structure learning backend to expressive specification languages for other relational models [16, 17].

3.2 The Count Manager

A key service for statistical-relational learning is counting how many times a given relational pattern (par-RV) is instantiated in the data. Such counts are known as *sufficient statistics*. Accessing sufficient statistics is often the main scalability bottleneck. The access patterns of a model search procedure are inherently sequential and random [2], and therefore it is important to cache sufficient statistics. Caching is even more important if the data is stored on disk in an RDBMS, rather than in main-memory. There are several reasons for employing an RDBMS for gathering sufficient statistics. (1) The machine learning application saves expensive data transfer by executing count operations in the database server space rather than local main memory. (2) SQL optimizations for aggregate functions such as SUM and COUNT can be leveraged. (3) Sufficient statistics can be stored in the RDBMS. For many datasets, the number of sufficient statistics runs in the millions and is too big for main memory. A novel aspect of FACTORBASE is managing *multi-relational sufficient statistics* that combine information *across* different tables in the relational database. This requires combining SQL aggregate functions with table joins [18].

3.3 The Model Manager

The Model Manager supports the construction and querying of large structured statistical models, which are stored in the **Model Database** MDB. Services provided by the Model Manager include the following. (1) Compute parameter estimates for the model using the sufficient statistics in the Count Database. (2) Computing model characteristics such as the number of parameters or degrees of freedom in a model. (3) Computing a model selection score that quantifies how well the model fits the multi-relational data. Model selection scores are usually functions of the number of parameters and the parameter estimates. By employing the SQL view mechanism, parameter estimates and model selection scores are updated automatically during the model search.

4 Conclusion

Compared to traditional learning with a single data table, learning for multi-relational data requires new system capabilities. In this paper we described FACTORBASE , a system that leverages the existing capabilities of an SQL-based RDBMS to support statistical-relational learning. Representational tasks include specifying metadata about structured first-order random variables, and storing the structure of a learned model. Computational tasks include storing and constructing sufficient statistics, and computing parameter estimates and model selection scores. We showed that SQL scripts can be used to implement these capabilities, with multiple advantages. These advantages include: 1) Fast program development through high-level SQL constructs for complex table and count operations. 2) Managing large and complex statistical objects that are too big to fit in main memory. For instance, some of our benchmark databases require storing and querying millions of sufficient statistics. While FACTORBASE provides good solutions for each of these system capabilities in isolation, the ease with which large complex statistical-relational objects can be integrated via SQL queries is a key feature.

Future Work. Further potential application areas for FACTORBASE include managing massive numbers of aggregate features for classification [19], and collective matrix factorization [20]. An important goal is a single RDBMS package for both learning and inference that integrates FACTORBASE with inference systems such as BayesStore and Tuffy.

There are opportunities for optimizing RDBMS operations for the workloads required by statistical-relational structure learning. These include view materialization and the key scalability bottleneck of computing multi-relational sufficient statistics. There are several fundamental system design choices whose trade-offs for SRL warrant exploration. These include the choice between pre-counting and post-counting sufficient statistics, and using main memory vs. RDBMS disk storage. For instance, model selection scores can be cached in either main memory or the database. Our SQL-based approach facilitates using distributed computing systems such as SparkSQL [21], which have shown great potential for scalability.

References

[1] Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakis, and Joseph M Hellerstein. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*, volume 1, pages 340–351, 2008.

[2] Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.

[3] Angelika Kimmig, Lilyana Mihalkova, and Lise Getoor. Lifted graphical models: a survey. *Machine Learning*, 99(1):1–45, 2015.

[4] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.

[5] Sameer Singh and Thore Graepel. Automated probabilistic modeling for relational data. In *CIKM*, pages 1497–1500. ACM, 2013.

[6] Sameer Singh and Thore Graepel. Automated probabilistic modelling for relational data. 2013.

[7] J. D. Ullman. *Principles of Database S¡ystems*. W. H. Freeman & Co., 2 edition, 1982.

[8] Zhensong Qian and Oliver Schulte. Factorbase: Multi-relational model learning with sql all the way. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015.

[9] Joseph M. Hellerstein, Christoper Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar. The MADlib analytics library: Or MAD skills, the SQL. *PVLDB*, 5(12):1700–1711, August 2012.

[10] Tim Kraska, Ameet Talwalkar, John C. Duchi, Rean Griffith, Michael J. Franklin, and Michael I. Jordan. MLbase: A distributed machine-learning system. In *CIDR*, 2013.

[11] Jörg Wicker, Lothar Richter, and Stefan Kramer. SINDBAD and SiQL: Overview, applications and future developments. In Savso Dvzeroski, Bart Goethals, and Panvce Panov, editors, *Inductive Databases and Constraint-Based Data Mining*, pages 289–309. Springer New York, 2010.

[12] Zhensong Qian and Oliver Schulte. The BayesBase system, 2015. `www.cs.sfu.ca/~oschulte/BayesBase/BayesBase.html`.

[13] Oliver Schulte and Hassan Khosravi. Learning graphical models for relational data via lattice search. *Machine Learning*, 88(3):331–368, 2012.

[14] Hassan Khosravi, Oliver Schulte, Tong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, pages 487–493, 2010.

[15] Trevor Walker, Ciaran O'Reilly, Gautam Kunapuli, Sriraam Natarajan, Richard Maclin, David Page, and Jude W. Shavlik. Automating the ILP setup task: Converting user advice about specific examples into general background knowledge. In *ILP*, pages 253–268, 2010.

[16] Alex Guazzelli, Michael Zeller, Wen-Ching Lin, and Graham Williams. Pmml: An open standard for sharing models. *The R Journal*, 1(1):60–65, 2009.

[17] Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: probabilistic models with unknown objects. In *IJCAI-05*, pages 1352–1359, 2005.

[18] Zhensong Qian, Oliver Schulte, and Yan Sun. Computing multi-relational sufficient statistics for large databases. In *CIKM*, pages 1249–1258. ACM, 2014.

[19] Alexandrin Popescul and Lyle Ungar. Feature generation and selection in multi-relational learning. In *Introduction to Statistical Relational Learning*, chapter 16, pages 453–476. MIT Press, 2007.

[20] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *SIGKDD*, pages 650–658. ACM, 2008.

[21] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Hua, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. Spark SQL: Relational data processing in Spark. In *SIGMOD Conference, To Appear*, 2015.