Schemas and Models

David Jensen and Jennifer Neville

Computer Science Department, University of Massachusetts, Amherst, MA 01003
{jensen, jneville}@cs.umass.edu

Abstract. We propose the *Schema-Model Framework*, which characterizes algorithms that learn probabilistic models from relational data as having two parts: a schema that identifies sets of related data items and groups them into relevant categories; and a model that allows probabilistic inference about those data items. The framework highlights how relational learning techniques must structure their own learning tasks in ways that propositional learners do not. The framework also highlights interesting directions for future research in relational learning.

1 Introduction

Several techniques have been developed that learn probabilistic models from data with complex relational structure. These include techniques for learning probabilistic relational models (Friedman, Getoor, Koller, and Pfeffer 1999), first-order Bayesian classifiers (Flach and Lachiche 1999), and relational probability trees (Neville and Jensen 2002). The number of techniques has grown to the point that they provide a sufficient basis for generalizations about some aspects of learning probabilistic models from relational data.

In this paper, we do three things. First, we describe the general characteristics of the learning task addressed by these techniques. We introduce this characterization only to provide a foundation for the second portion of the paper. In this second portion, we propose the *Schema-Model Framework*, which characterizes algorithms that learn probabilistic models from relational data as having two parts: a *schema* that identifies sets of related data items and groups them into relevant categories; and a *model* that allows probabilistic inference about those data items. Third, we discuss several new research problems that arise in the context of schemas and models.

We propose the Schema-Model Framework (SMF) to highlight some of the unique challenges and opportunities of learning from relational data, rather than to provide an overarching characterization of all aspects of these learning algorithms. The SMF embodies a relatively simple point: techniques that learn from relational data must structure their own learning tasks in ways that propositional learners do not. The SMF helps identify: 1) new representational and algorithmic elements that are needed to learn from relational data; 2) alternative methods used by learning techniques to implement these elements; and 3) new directions for research in relational learning.

D. Jensen and J. Neville (2002). Schemas and models. *Papers of the KDD 2002 Workshop on Multi-Relational Data Mining*.

The next two sections lay the groundwork for discussing the SMF. These sections can be skimmed or skipped entirely by readers familiar with techniques for learning from relational data. Section 4 introduces the framework, and section 5 presents some new learning problems that it highlights. The final section discusses some related and future work.

2 Probabilistic relational learning

In this section, we attempt to provide a general characterization of learning probabilistic models from relational data. For the remainder of the paper, we shorten this to "relational learning" where the reference to probabilistic learning is clear from context.

2.1 Data

A data set D consists of a set of *objects* and *links*, $D = \{O,L\}$. Objects $o_i \in O$ generally represent people, places, things, and events. For example, in the domain of movies, a data set might contain objects representing movies, actors, directors, producers, studios, and movie releases.[1] Links $l_i \in L$ represent relations between two or more objects. For example, in the movie domain, a link might represent the relation *Directed(Director,Movie)*, *Produced (Producer,Movie)*, or *Awarded(Award,Actor,Movie)*. As we will discuss below, links need not be binary, though binary links are common. The term *item* refers to objects and links, and is denoted e.

An object or link contains a set A of *attributes*, where $|A| \geq 0$. In other work, attributes are called *variables* or *features* of an item. For example, attributes on objects might denote the age of an actor, the genre of a movie, or the location of a studio. Attributes on links might denote the salary an actor was paid for acting in a given movie or the role which an actor played in a movie. For a given item e_i, the jth attribute $a_{i,j}$ has a name $name(a_{i,j})$ and a value $value(a_{i,j})$. Items can be characterized by their attribute vector $<value(a_{i,1}), value(a_{i,2}),...value(a_{i,n})>$. For convenience, $name(A_i)$ and $value(A_i)$ represent sets of names and values, respectively, on the ith item. Attributes with identical names are assumed to have values of comparable data types. For example, all values of *age* are assumed to be integers, and all values of *release date* are assumed to be dates.

Items typically can be grouped into one or more disjoint subsets of homogeneous type. For example, objects in the domain of movies could be grouped into persons, movies, studios, and awards. We do not assume that there is only one "correct" typing system over

[1] The movie examples presented throughout this paper are drawn from our work with the Internet Movie Database (www.imdb.com). The IMDb is a large relational datatset that catalogues diverse information about movies, containing data for 300,000 movies, 700,000 people (e.g. actors, directors, producers), and 12,000 studios.

the objects and links represented in the data. Indeed, one system we discuss below implicitly attempt to learn a useful typing system. That said, databases often have an initial typing system, and items of a single type within that system often have equivalent attribute sets, $name(A_i) = name(A_j)$ for all i,j. Typing systems for items are often called an *ontology*, particularly when types are organized into a hierarchical structure that allows generalization about the lower-level entities (e.g., a director is a person and thus has birth date).

Several nearly equivalent formalisms exist for representing these types of data sets. For example, data sets could be represented as:

- *Graphs* — A data set D can be thought of as a directed hypergraph with vertices O and hyperedges L. The hypergraph may have only one connected component or several connected components.
- *Database tables* — A data set D can also be thought of as a relational database with entities O and relations L. Items of each type would be stored in a separate table with one field for each attribute.
- *First-order logic statements* — D can be thought of as a collection of statements in first-order logic.

For purposes of this paper, we use the formalisms of graph theory and use binary links, though no restriction to binary relations is implied.[2]

In contrast to the assumptions underlying of much of propositional learning, attribute vectors characterizing objects in D are not assumed to be statistically independent or identically distributed. For example, two movies made by the same director are likely to be of similar genres and made in similar years.

2.2 Models

Algorithms for probabilistic relational learning have been developed that learn probability distributions over possible attribute values, possible links, or possible objects. A model over possible attribute values might predict the box-office receipts of a movie based on the success of previous movies made by the movie's director, producer, and studio. A model over possible links might predict the probability that a given actor will star in a future movie, represented as an *acted-in* link between the movie object and that actor. A model over possible objects might predict the probability that a movie will be released in a particular country, represented as a new *release* object for that movie (with an appropriate link to the movie itself).

We focus on models that estimate probability distributions over possible attribute values, though much of our discussion is also relevant to prediction of links and objects. The task of estimating probability distributions over the values of a given attribute would appear to differ little from traditional propositional learning. However, algorithms for

[2] The apparent limitations imposed by binary links are largely illusory and can usually be escaped by creating objects to represent more complex relations such as events (see Davidson 1967).

relational learning typically look beyond the item for which the attribute is defined, to consider the effect of related objects on the probability distribution. For example, in order to predict the box-office success of a movie, a relational model would consider not only the attributes of the movie, but attributes of the actors in the movie and the director, producer, and studio that made the movie. A model might go even further and consider attributes of much more "distant" objects (in the sense of a graph neighborhood).

Some algorithms for relational learning estimate a probability distribution over the values of one or more attributes of item e_i given the items in its neighborhood. That is, $P(value(a_{i,j})|neighborhood(e_i))$, where $neighborhood(e_i)$ is a set of objects and links reachable from e_i and it is typically less than the entire graph. Others estimate the full joint probability distribution over a large number of attributes of related objects. Below, we discuss both types of models.

We divide the set of all attributes A into fixed and inferred attributes. For each fixed attribute A_j, value($a_{i,j}$) is known a priori for each item e_i. No inference about the probability distribution of a fixed attribute is necessary. In contrast, for each inferred attribute A_j, value($a_{i,j}$) is missing for one or more items e_i, and the goal of modeling is to estimate a probability distribution for these missing values. Friedman et al. call these "fixed" and "probabilistic" attributes, respectively.

2.3 Related work

Probabilistic relational learning is distinct from a set of related learning tasks that appear superficially similar:

- *Traditional inductive logic programming* — Traditional ILP systems learn deterministic models rather than estimating probability distributions over attributes, links, or objects. Recent developments in stochastic logic programming (Muggleton 2000, Cussens 2001) and Bayesian logic programs (Kersting and De Raedt 2000) are interesting alternatives to these deterministic models, which adapt ILP techniques to learn probabilistic first-order models.
- *Propositionalization of relational data* — A common approach to learning from relational data is to "propositionalize" the data rather than retain its inherent structure. The attributes of an item e_i can be "pulled back" and recorded as a local attribute of e_i. Then a probabilistic model can be learned from the resulting feature-vector representation of the data. However, this approach requires foreknowledge of the correct relational features. It also assumes that the nature of statistical inference is not changed by the relational structure of the data, an assumption that has been challenged by recent work showing how the structure of relational data affects parameter estimates (Jensen and Neville 2002a) and evaluation of learned models (Jensen and Neville 2002b).
- *Learning propositional concepts in relational databases* — Much of the work on knowledge discovery in databases addresses how to use techniques from relational databases to facilitate data mining. This work deals with relational databases, but

relatively little of it deals with relational data. Much of this work assumes that data are propositionalized (see above) or that data are drawn from only a single relation.

- *Multiple instance learning* — Some recent work has focused on how to learn in tasks where each data item is a "bag" containing multiple instances and where a positive bag indicates that one or more of the items it contains is positive. For example, a credit card account may be a bag of transactions, where some of the transactions in a given bag may be fraudulent. In relational learning, a data instance can be thought of as an item surrounded by a neighborhood of other items. However, in relational learning, the other items only provide context for estimating the probability distribution of the attributes of a single item, and do not represent multiple instances within a bag.

3 Example Systems

3.1 Probabilistic Relational Models

Probabilistic relational models (PRMs) extend Bayesian networks to support reasoning in relational domains, allowing the rich relational information to be incorporated into traditional Bayes net dependency structures. PRMs specify a probability model over a relational database with a fixed schema. Given a set of objects and the links between them, a PRM defines a full joint probability distribution over the attribute values of the objects. Attributes of an object can depend probabilistically on other attributes of the object, as well as on attributes of objects in its relational neighborhood. PRMs can also model uncertainty over both object and link existence.

PRMs make use of pre-specified database schemas that describe a fixed set of object types, each with a set of attributes and an associated set of links. A PRM represents a probability distribution over possible instances of a given schema, where an instance of the schema specifies (1) the set of objects of each type, (2) the set of links between the objects, and (3) the values of all attributes. For a particular skeleton set of objects and links (with missing attribute information), instantiating a PRM induces an unrolled Bayesian network model where random variables represent the individual attribute values of all the objects in the skeleton.

Instead of defining the dependency structure over attributes, as in conventional Bayes nets, PRMs define a generic dependency structure at the level of object *types*. Each attribute A_i associated with object type X is linked to a set of parents that influence the value of A_i. Parents of A_i are either (1) other attributes associated with type X, or (2) attributes associated with objects of type Y where objects Y are linked to objects X. (PRMs can also represent dependencies along longer chains of relations but for simplicity we will limit discussion to direct relations.) For the latter type of dependency, if the relation between X and Y is one-to-many, the “parent” consists of a set of attribute values. In this situation, PRMs use standard database aggregation functions (e.g. *max*, *mode*, *average*) to

map sets of values into single values. Learning a PRM consists of two tasks: learning the dependency structure, and estimating the parameters of the conditional probability distributions used to specify the local probability models for an attribute given its parents.

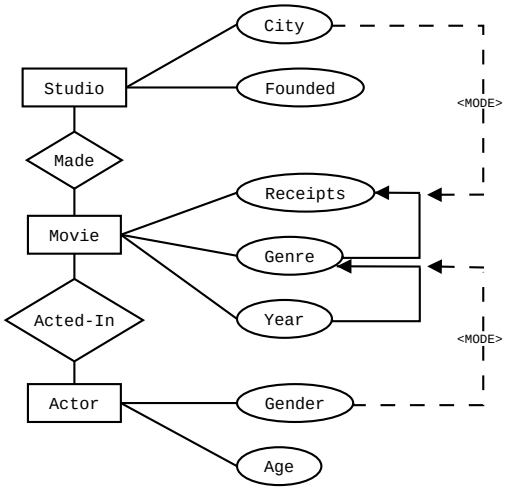

Fig. 1. The schema of a relational database in the movie domain and the dependencies among attributes encoded by a PRM.

For example, figure 1 shows an example of a PRM that could have be learned in the movie domain. Three types of objects in the schema are represented by boxes: studios, movies, and actors. Each type has a set of associated attributes represented by ovals. Links, represented by diamonds, relate studios and actors to movies. Directed arcs represent the learned dependency structure of a PRM. Dashed lines indicate an aggregated attribute – multiple actors may star in any particular movie and more than one studio may be involved in production of each movie. `<MODE>` annotations on the arcs indicate that the model uses the most prevalent attribute value from sets while reasoning. The model outlines the dependency of movie genre on both the year of the movie and the most prevalent gender of the actors in the movie. The model also shows that genre and most prevalent studio location influence movie success (box office receipts).

3.2 First-order Bayesian Classifiers

1BC is a simple Bayesian Classifier that builds discriminative models of attributes from relational data. Examples consist of objects and their relational neighborhood and first-order features evaluate attributes values of various items in the examples. A probability

density function is specified for the target class using first-order features, making the standard naïve Bayes assumption that features are conditionally independent given the class. As with PRMs, it is assumed that the domain provides a well-defined notion of object types (individuals), with associated sets of links (structural predicates) and attributes (property predicates).

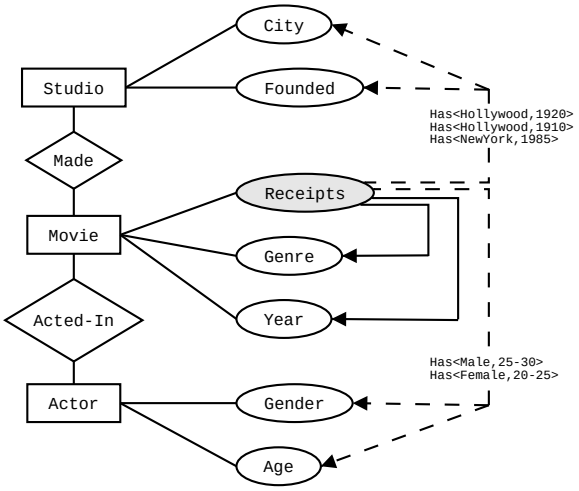

Fig. 2. The schema of a relational database in the movie domain and the dependencies among attributes encoded by a 1BC model.

1BC is designed to model at varying levels of decomposition. Level-0 does not decompose the examples at all and uses complex probability estimators over sets and multisets to model objects with heterogeneous relational neighborhoods in terms of their components. If there are too few examples to produce accurate probability estimates at this level, the examples can be decomposed into level-1 elementary features that consider the items of each example individually. For each original type X and its set of attributes A, new types are defined for combinations of attributes in A. Boolean features are constructed for these types which denote whether a particular example contains an item of the specified type. For example, the feature *HasActor<Female,20-25>* is true of a movie example is there exists a female actress in the age range (20,25) starring in the movie. A level-2 decomposition further subdivides these types by the attributes and constructs features that reference a single attribute of an item, such as *HasActor<Male>*.[3]

[3] Work by Craven and Slattery (2001) is related to 1BC, but it uses a complementary method. Rather than use an ILP system to construct features for a Bayesian classifier, Craven and Slattery use a Bayesian classifier to construct predicates for an ILP system.

The implementation of 1BC uses Tertius (Flach and Lachiche 2001) to return a set of interesting first-order features at each level of decomposition, given domain and background knowledge. These first-order features consist of zero or more links which reference specific items of an example and attributes of that item. Features are multi-valued if the links reference an attribute of a single item (e.g. movie) or boolean if the series of links refer to a set of items (e.g. actors).

Figure 2 shows an example a 1BC model that could be used to classify movie success. First-order features use the relational context to predict a movie's box office receipts – attributes of the movie itself are considered (e.g. genre, year) as well as attributes of related objects (e.g. studio city and founding year, actor age and gender). Each feature is used as an attribute in the naïve Bayes formula and the conditional probabilities of features given class are estimated from the training data.

3.3 Relational Probability Trees

Relational Probability Trees (RPTs) extend standard probability estimation trees to a relational setting. RPTs are used in PROXIMITY, a relational knowledge discovery system designed to operate on graph databases.[4] The RPT algorithm takes a collection of subgraphs as input and constructs a probability estimation tree to predict the target class label. Each subgraph in the collection contains one target object to be classified; the other items in the subgraph form its relational neighborhood. An RPT encodes a probability distribution over the class label given attributes of both the target objects and of other items in the subgraphs.

The subgraphs in the input collection specify the relational neighborhood that will be considered by the model and their structure defines a typing over items in the collection. Subgraphs are extracted from a larger graph database using the visual query language QGRAPH (Blau, Immerman, and Jensen 2001). Queries in the language allow for variation in the number and types of objects and links that form the subgraphs and return collections of all matching subgraphs from the database. Objects and links in the returned collection are named for particular roles that they serve in the matched subgraphs and may be duplicated across subgraphs if they match in multiple ways. The returned collection is a view of the graph — subgraph membership defines the relevant local relational context and names serve to dynamically assign object and link types. After querying, the user specifies a set of attributes for each type that will be available to the model during the learning phase.

The RPT induction algorithm searches over a space of relational features involving attributes of items in the input subgraphs. Relational features are similar to traditional propositional features in that they identify both an attribute and a way of testing the values of the attribute. However, relational features may also identify a particular relation (e.g. *ActedIn*) that links a single object (e.g. movie) to a set of other objects (e.g. actors). If this

[4] For additional details on PROXIMITY, see <http://kdl.cs.umass.edu>.

is the case, the attribute referenced by the feature may belong to the linked objects (e.g. actor age), and the test is conducted on the set of attribute values on the linked objects. The algorithm searches over a space of possible item types (e.g. actor), attributes (e.g. age), and aggregation functions (e.g. count). Each node in an RPT tests a binary relational feature — for example, whether at least two actors in a movie are over 65.

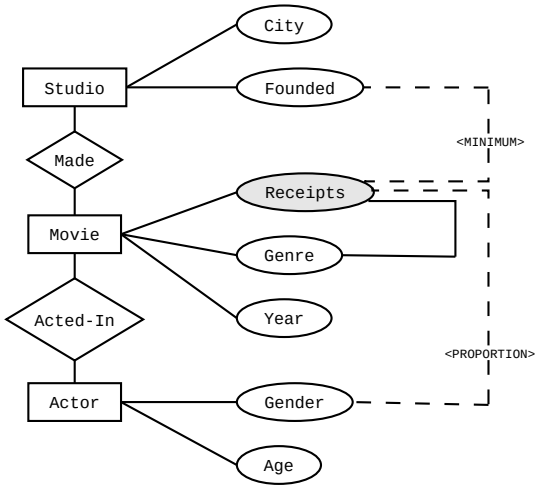

Fig. 3. The schema of a relational database in the movie domain and the dependencies among attributes encoded by a RPT.

Figure 3 shows the dependencies of an RPT for classifying movie success. The model has selected three attributes relevant to predicting a movie's box office receipts. Features of the model test attributes of the movie (e.g. genre) and attributes of related objects (e.g. studio founding year, actor gender). Annotations on the arcs indicate the aggregations used in the features. More specifically, the tree could test whether the earliest founding year of studios is below a threshold (e.g. <1910) and whether the proportion of actor genders satisfies some constraint (e.g. females<20%).

4 Schema-model Framework

Each of the learning techniques outlined above has roots in techniques for learning in propositional data. PRMs extend traditional Bayesian networks, first-order Bayesian classifiers extend traditional Bayesian classifiers, and RPTs extend techniques from

probability estimation trees (Provost and Domingos 2000). However, each technique also requires substantial modification to learn effectively from relational data.

We group these modifications into changes relevant to a *schema* and those relevant to a *model*. The *schema* addresses how the technique structures relational data for the modeling algorithm. The *model* addresses alterations to the modeling technique that handle the more complex structure of relational data.

4.1 Schema

In contrast to the highly specified task of propositional learning, relational learning raises new questions. For example, compare learning a model of the probability distribution of a movie's box-office receipts from propositional and relational data, respectively. In propositional data, the learner is given a fixed set of attributes intrinsic to each movie. In relational data, how much of the relational neighborhood around a movie should influence the probability distribution of a movie's box-office receipts? In propositional data, each attribute is considered as a separate entity. In relational data, should the attributes of the movie's director, producer, and actors contribute independently to predicting box-office receipts, or should we consider some aggregate attributes of all people connected with the movie? We call the collective answers to these questions a *schema*.

The schema provides access to an appropriate neighborhood of items that influence the probability distribution of inferred attributes. Either this neighborhood should include all items that influence the inferred attributes, or the influence of those items should be carried through an appropriate inference chain (see below). Second, the schema groups items into equivalence classes that can be used by the model.

In PRMs, the schema is a provided by a combination of the dependency structure of the PRM itself, which provides a set of parents for each probabilistic attribute, and the *database schema*, which groups objects and links into mutually exclusive and collectively exhaustive types.[5] For the purposes of this discussion, two features of PRMs are particularly noteworthy. First, the typing system is given by the underlying database, and cannot be altered without changing the structure of the database.[6] Second, the probability distribution of a given attribute appears to depend only on a given set of parents, but the inference procedure employed by PRMs allows probabilistic attributes to depend on other probabilistic attributes whose values are inferred by the model. Such an *inference chain* allows the inferred distribution of an attribute to be influenced by attributes beyond its immediate parents.

In 1BC, the schema is provided by a combination of the structure of the data graph, which is assumed to consist of disjoint subgraphs representing individual instances, and

---

[5] Note that the database schema is related to, but different than the schema for the learning technique.

[6] Other work related to PRMs has dealt with inducing relevant types from relational data (Taskar, Segal, and Koller 2001)

first-order features learned by an ILP system. Alternatively, the instance subgraphs could be formed by some sort of query mechanism, rather than relying on natural divisions of the data, although this option is not discussed in the literature on 1BC. Note that the first-order features are not learned with direct reference to their utility to the probabilistic model, though they are learned rather than being provided *a priori*.

In RPTs, the schema is provided by the results of a graph query executed prior to learning the model. The query returns a collection of subgraphs, each of which contains one target item with the attribute to be modeled. The query also associates names with items in each subgraph, which can be interpreted as types (e.g., movie, actor, producer, director, and studio). This approach to providing a schema sits midway between PRMs and 1BC; the schema is not fixed, but neither is it learned.

4.2 Model

A schema provides substantial additional structure to relational learning tasks, but creating a probabilistic model in the presence of a schema still raises unique questions. Again, compare learning a probability distribution of a movie's box-office receipts in the propositional and relational cases. In propositional data, every instance has an identical and fixed structure. In relational data, how should we deal with the varying size of the neighborhood surrounding a given movie? A model of relational data must accurately estimate probability distributions in the face of this varying structure.

PRMs address this challenge by aggregating over the attribute values of multiple parents of the same type. For example, if a movie's genre depends on the gender of the movie's actors, and a movie has five actors, then the PRM would aggregate the five values of gender with a function such as *mode*. These aggregation functions appear to be selected *a priori*, rather than selected by the learning algorithm itself.

1BC addresses the challenge of variable-sized neighborhoods by using first-order features that produce boolean values that characterize the neighborhood. For example, a first-order feature might determine whether an actor with a particular gender and age acted in the movie. The attribute would be true if one or more actors met the gender and age criteria. The first-order features are not formed during model learning. Instead, they are formed prior to learning the Bayes classifier.

The RPT induction algorithm also forms first-order features, but it forms these features as part of the same process that constructs the tree. The RPT induction algorithm searches the a space of possible types, attributes, aggregation functions, and thresholds to form binary predicates such as *average actor age < 30*. Thus, RPT induction creates similar types of first-order features as 1BC. RPT induction searches over a larger space of aggregation functions and thresholds than either PRMs or 1BC, but RPT induction lacks the ability of 1BC to automatically create long first-order chains of conditions (e.g., whether an actor over 30 also starred in an action adventure movie).

5 New problems in relational KD

5.1 Learning schemas

The quality of any probabilistic model of relational data depends on the features available to the modeling technique. In the language of schemas and models, model quality depends on schema quality. Ideally, the schema should be learned such that it provides the necessary "raw materials" for constructing an accurate probabilistic model.

First, the schema should provide access to an appropriate neighborhood of items that influence the probability distribution of inferred attributes. If the neighborhood is too small, and the influence of other relevant items is not carried through an inference chain, we conjecture that the model will have increased error due to *bias* (Friedman 1997) because it cannot represent the influence of some relevant items. If the neighborhood is too large, we conjecture that the model will have increased errors due to *variance* (Friedman 1997), because the data used for probability estimates will include irrelevant items that merely introduce "noise" that masks the "signal" provided by relevant items.

Second, the schema groups items into equivalence classes that can be used by the model to form relational features. We conjecture that this component of a schema must balance two types of variance errors. A schema might group items at a level of detail that is either too fine or too course. For example, in the movie domain, a schema might designate actors, directors, and producers as different types of objects, when the aggregate level of experience of all persons associated with a movie is the best predictor of the box-office success of a movie. Alternatively, a schema could designate actors, directors, and producers as "people" when the experience of directors and producers alone are the best predictors of success.

None of the techniques discussed above provides an entirely satisfactory solution to determining the correct schema. PRMs take the schema as given, based on the structure of the database. 1BC learns a set of relational features, but without regard to the effect of those features on the quality of the probabilistic model learned. RPTs assume a pre-specified schema, allowing alternative queries to be used, but not supporting any automated exploration of the space of possible schemas.

5.2 Alternative schemas

The relatively simple schemas used by current approaches do not exhaust the potential types of schemas that might be useful for learning. For example, PRMs and RPTs currently assume a single set of mutually exclusive and collectively exhaustive types. Alternatives to this approach include: 1) *probabilistic schemas*, where each item is characterized by a probability distribution over several possible types; 2) *overlapping schemas*, where each item can have multiple types; and 3) *multi-resolution schemas*, where each item is in a set of types of increasing generality. Further, it is not clear that one

schema is appropriate for learning all probability distributions. This is particularly relevant to PRMs, which might benefit from learning different conditional probability distributions with different schemas.

5.3 Learning models

Finally, a variety of challenges remain in learning accurate probabilistic models from relational data. In related work (Jensen and Neville 2002a, 2002b), we have shown how common characteristics of relational data can complicate learning and evaluating relational models. Specifically, high levels of linkage and autocorrelation in relational data can cause learning algorithms to systematically prefer attributes with the *least* supporting evidence. Similarly, these characteristics of relational data can cause evaluations of learned models to vastly overestimate the utility of those models. In addition to these, we suspect that other statistical challenges will emerge in the next few years, as we gain experience with learning probabilistic models from relational data.

6 Related and Future work

Related work on Bayesian logic programs (BLPs) (Kersting and De Raedt 2000) has characterized the expressivity of several probabilistic models, including Probabilistic relational models (PRMs) (Freidman et. al 1999), relational Bayesian nets (RBNs) (Jaeger 1997), and probabilistic logic programs (PLPs) (Ngo and Haddawy 1997). Kersting and De Raedt compared these three types of models with BLPs and investigated the relationships among the various knowledge representations. They outlined a chain of positive inclusion for the four types of models, from PRMs (least expressive) to RBNs, to PLPs, to BLPs (most expressive). Instead of focusing on model representation, the SMF relates systems based on aspects of model learning. At the outset of our work on the SMF, PRMs were the only type of model considered by Kersting and De Raedt for which learning had been investigated. Consequently, we limit the initial discussion in this paper to PRMs. However, recent work on Bayesian logic programs (Kersting and De Raedt 2002) has outlined methods to learn both the parameters and structure. In future work, we hope to incorporate BLPs into the SMF.

For simplicity, our work was limited to systems that model degrees of belief over individuals (Halpern 1990). Halpern outlines two types of probabilistic structures in his analysis of first order logics of probability. The first type of structures represent subjective probabilities concerning particular individuals (e.g. any given bird is likely but not certain to fly), modeling probabilities over possible worlds. The second type of structures represent qualitative statistical statements (e.g. most birds fly), modeling probabilities over the domain. The systems described in this paper are, in Halpern's notation, Type I models. However, there is no reason for the schema-model framework to be limited to

Type I models. Work in Type II probabilistic models includes stochastic logics programs (Muggleton 2000, Cussens 2001) and PRISMs (Sato and Kameya 2001). In future work, we hope to incorporate these systems into the SMF.

Acknowledgment

This effort is supported by DARPA and AFRL under contract numbers F30602-00-2-0597 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA, the AFRL, or the U.S. Government.

References

1. Blau, H., N. Immerman, and D. Jensen. (2001). A Visual Query Language for Relational Knowledge Discovery. University of Massachusetts Amherst, Department of Computer Science. Technical Report 01-28.
2. Craven, M. and S. Slattery (2001). Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning* 43: 97-119.
3. Cussens, J. (2001). Parameter estimation in stochastic logic programs. *Machine Learning* 43:245-271.
4. Davidson, D. (1967). The logical form of action sentences. In *The Logic of Decision and Action*. N. Rescher (Ed.). University of Pittsburgh Press.
5. Flach, P. and N. Lachiche (1999). 1BC: A first-order Bayesian classifier. In *ILP'99*. 93-103.
6. Flach, P. and N. Lachiche (2001). Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning* 42:61-95.
7. Friedman, N., L. Getoor, D. Koller and A. Pfeffer. (1999). Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. 1300-1309.
8. Friedman, J. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* 1:55-77.
9. Halpern, J. Y. (1990). An analysis of first-order logics of probability. *Artificial Intelligence* 46:311-350.
10. Jaeger, M. (1997). Relational Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*. 266-273.
11. Jensen, D. and J. Neville (2002a). Linkage and autocorrelation cause feature selection bias in relational learning. To appear in *ICML2002*.
12. Jensen, D. and J. Neville (2002b). Linkage and autocorrelation cause bias in evaluation of relational learners. To appear in *ILP2002*.
13. Kersting, K. and L. De Raedt. (2000). Bayesian logic programs. In *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming* 138-155.

14. Kersting, K. and L. De Raedt. (2002). Basic principles of learning Bayesian logic programs. Technical Report No. 174, Institute for Computer Science, University of Freiburg, Germany.
15. Muggleton, S. (2000). Learning stochastic logic programs. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*. AAAI Press.
16. Neville, J. and D. Jensen (2000). Iterative classification in relational data. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data.* AAAI Press. 13-20.
17. Neville, J. and D. Jensen (2002). Learning relational probability trees. University of Massachusetts Amherst, Department of Computer Science, Technical report.
18. Ngo, L. and P. Haddawy (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science* 171:147-177.
19. Provost, F. and P. Domingos (2000). "Well-trained PETs: Improving probability estimation trees." CeDER Working Paper #IS-00-04. Stern School of Business, New York University.
20. Sato, T. and Y. Kameya. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research* 15:391-454.
21. Taskar, B., E. Segal, and D. Koller (2001). Probabilistic clustering in relational data. In *AAAI-2001*. 870-876.