Data Mining in Social Networks

David Jensen and Jennifer Neville

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

Abstract. Several techniques for learning statistical models have been developed recently by researchers in machine learning and data mining. All of these techniques must address a similar set of representational and algorithmic choices and must face a set of statistical challenges unique to learning from relational data.

Introduction

Recent research projects in two closely related areas of computer science — machine learning and data mining — have developed methods for constructing statistical models of network data. Examples of such data include social networks, networks of web pages, complex relational databases, and data on interrelated people, places, things, and events extracted from text documents. Such data sets are often called "relational" because the relations among entities are central (e.g., acquaintanceship ties between people, links between web pages, or organizational affiliations between people and organizations).[1]

These algorithms differ from a substantially older and more established set of data mining algorithms developed to analyze propositional data. Propositional data are individual records, each of which can be represented as an attribute vector and each of which are assumed to be statistically independent of any other. For example, a propositional data set for learning medical diagnostic rules might represent each patient as a vector of diagnostic test results, and analysis would assume that knowing the disease of one patient tells you nothing about another patient. In contrast, analysis of a relational representation of the same data would retract this latter assumption and add information about familial relationships, workplace contacts, and other relationships among patients that might influence their medical status.

The handful of data mining techniques that have been developed recently for relational data include probabilistic relational models (PRMs) (Friedman, Getoor, Koller, and Pfeffer 1999), Bayesian logic programs (BLPs) (Kersting and de Raedt 2000), first-order Bayesian classifiers (Flach and Lachiche 1999), and relational probability trees (RPTs) (Jensen and Neville 2002). In each of these cases, both the structure and the parameters of a statistical model can be learned directly from data, easing the job of data analysts, and greatly improving the fidelity of the resulting model. Older techniques include inductive logic programming (ILP) (Muggleton 1992; Dzeroski and Lavrac 2001) and social network analysis (Wasserman and Faust 1994).

For example, we have employed relational probability trees (RPTs) to learn models that predict the box office success of a movie based on attributes of the movie and related records,

[1] This meaning of "relational" should be distinguished from the more restrictive meaning of "data stored in relational databases." While relational databases can represent relational data, relational data can also be represented and accessed in other ways.

D. Jensen and J. Neville (to appear). Data mining in networks. *Papers of the Symposium on Dynamic Social Network Modeling and Analysis*. National Academy of Sciences. November 7-9, 2002. Washington, DC: National Academy Press.

including the movie's actors, directors, producers, and the studios that made the movie. We have also analyzed relational data in other ways to predict fraudulent cell phone use based on the calling patterns of individual phone numbers. Finally, we have produced models that predict the function and location of proteins in a cell based on network of interactions with other proteins.

Many of these techniques for relational learning share a common set of statistical challenges and design issues. In this paper, we survey these issues, using examples from our work on PROXIMITY, an integrated system for relational learning, and an algorithm for learning RPTs that we have incorporated into PROXIMITY. For each issue, we briefly discuss our design choices in PROXIMITY, and point to alternative approaches used by other systems.

We begin by describing a specific data set and an example analysis task — predicting the box-office receipts of movies — that we use throughout the remainder of the paper. Next, we describe some of the basic features of PROXIMITY and our approach to querying data and learning RPTs. The next two sections discuss a set of representational and algorithmic choices made by the different techniques and a set of statistical issues unique to relational data. We finish with some brief conclusions.

Example Data and Analysis Task

Consider the relational data shown schematically in Figure 1. The data consist of movies and associated objects including people (who act in, produce, or direct the movies), organizations (studios), events (releases of the movie), and other objects (awards). These objects are connected in the ways that you would expect (e.g., actors are linked to movies they act in) and in some occasionally unexpected ways (e.g., movies are linked directly to other movies that are remakes). In addition to the high-level structure of the database shown in Figure 1, the database contains attributes associated with each object, including the titles and genres of movies, the names and ages of persons, and the countries and box-office receipts of movie releases.

The data are drawn primarily from a large online resource, the Internet Movie Database (*www.imdb.com*) that makes its data public for research and other non-commercial purposes. In addition, we have added other data drawn from the Hollywood Stock Exchange (*www.hsx.com*), an artificial market where players trade in stocks that track the relative popularity of movie actors.

The data are voluminous, consisting of over 300,000 movies, 650,000 persons, and 11,000 studios. Those objects are connected by over 2.3 million acted-in links, 300,000 directed links, and 200,000 produced links. The available data on movies vary widely. For example, not all movies have releases, and HSX data are only available for a small percentage of actors in IMDb. Data are more complete for more recent movies and persons.

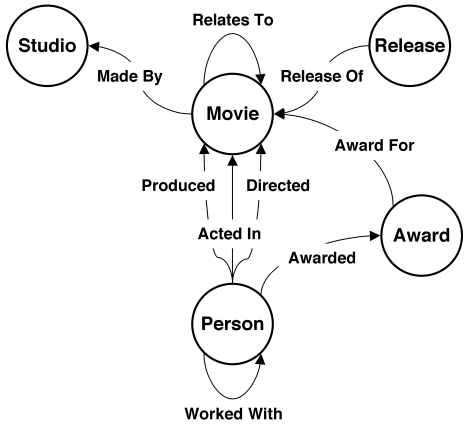

Figure 1: Example schema for data from the Internet Movie Database.

The movie data support a variety of interesting predictive modeling tasks. We have already mentioned one — predicting the opening weekend box office receipts of a movie — and we will use this task as an example throughout the paper. Specifically, we will focus on predicting a probability distribution over a simple binary attribute of movies — Does the movie make more than $2 million in its opening weekend? We will call this attribute *receipts*.

We could attempt to predict other attributes of objects (e.g., a movie's genre or an actor's gender) or attributes of links (e.g, the type of a link between a person and a movie) with PROXIMITY. In addition to these, other types of prediction tasks are certainly possible. One could attempt to learn models that predict missing links between objects. For example, reviewers sometimes call a movie a "crypto-sequel" when it stars the same actors and has a similar plot line as another recent movie, but does not explicitly tie the two storylines. For example, the 1998 movie "You've Got Mail" starring Tom Hanks and Meg Ryan was said to be a crypto-sequel to the 1993 movie "Sleepless in Seattle" (as well as a remake of the 1940 movie "Shop Around The Corner" starring James Stewart and Margaret Sullavan). Given enough examples of crypto-sequels, a data mining algorithm could learn a predictive model from the movie data. Recent work by Getoor, Friedman, Koller, and Taskar (2001) has created models that predict the existence of missing links.

One could also attempt to learn models that predict an attribute of a subgraph, rather than only a single object or link. For example, the emergence of a highly paid Hollywood movie star may consist of a set of successful movies in which the actor had a starring role and one or more awards. Models of this pattern would consist of many objects and links, combined in a particular temporal sequence.

In this paper, we will focus almost exclusively on the task of learning probability distributions over the values of attributes of objects and links. While predicting link existence and classifying subgraphs are extremely interesting problems, the techniques learning probabilistic models for these tasks are much less numerous and much less well-developed than for simple attribute modeling.

One important input to relational learning algorithms is a *schema* or interpretation of the data that specifies a type system over the objects and links in the data. For example, Figure 1 above specifies one schema for the movie data, but others are possible. For example, an alternative schema might specify people as either actors, directors, or producers. Figure 2 provides a hierarchy of possible object types as well as two possible families of schemas constructed from those object types (a full schema would also specify a set of link types). Such a hierarchy is sometimes called an *ontology* (Gruber 1993).

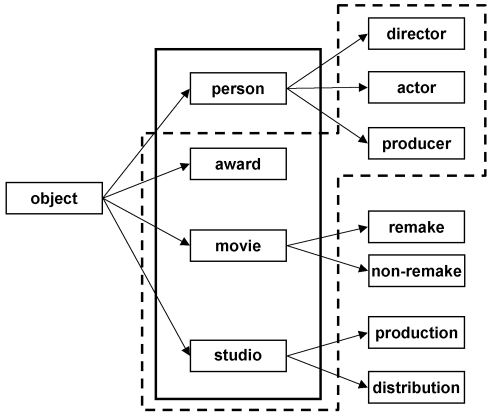

Figure 2: An example ontology of movie objects.

Querying and Learning

To address learning tasks of this kind, our research group is constructing PROXIMITY — a system for machine learning and data mining in relational data. The system is designed as a framework within which a variety of analysis tools can be used in combination. At the foundation of PROXIMITY is a graph database for storing semi-structured data that can be represented as a graph. The database can be accessed by tools for querying data, sampling data, and calculating attributes that depend partially or entirely on network structure (e.g., measures drawn from social network analysis). Sampled data can then be analyzed with tools that construct statistical models. Finally, all these tools can be called from a scripting language interface. In addition to these components, we are developing additional components for clustering, graph partitioning, and additional types of statistical modeling.

In this paper, we will focus on a relatively simple combination of two tools — our query language and one of our learning algorithm. The query language is a visual language for expressing queries to the graph database. The learning algorithm constructs relational probability trees (RPTs), a type of probabilistic classifier for relational data. The two components work in concert. The query language is used to extract subgraphs from a large network of data; the RPT algorithm is used to learn a model that estimates a conditional probability distribution for the

value of an attribute of a class of objects or links represented in all those subgraphs. That estimate is conditioned on the attributes of other objects and links in the subgraph.

For example, a query might extract subgraphs consisting of a movie and all directly related actors, producers, directors, studios, and awards. An RPT could then be constructed to estimate the probability that a movie makes more than $2 million in its opening weekend (*receipts* = *True*), given attributes of the actors, producers, directors, studios, and awards. Note that different movies will have different numbers of related objects such as actors and awards. Thus, the subgraphs could not be represented directly as simple attribute vectors.

Our query language, QGraph, represents queries as graphs with associated attribute constraints and annotations on vertices and edges (Blau, Immerman, and Jensen 2002). For example, Figure 3 shows the query described above with a movie and all its related objects. The numeric annotation [1..] on the actor vertex specifies that a match must have one or more actors, and that all associated actors should be returned as part of each matching subgraph. Some object types and link types are left unspecified because of known connectivity constraints in the data. Matches to the query are shown in Figure 4. Actual names of people, studios, and movies are left out for simplicity. The first match has three actors and no award; the second has four actors and no award, and shares an actor and a studio with the first match; the third match has only a single actor, but won an award. The fact that entire subgraphs are returned as part of a match is a subtle, yet vital, feature of the language for our purposes. Other languages such as SQL, for example, can only return a single record as a match, not a record of variable size, such as a subgraph.

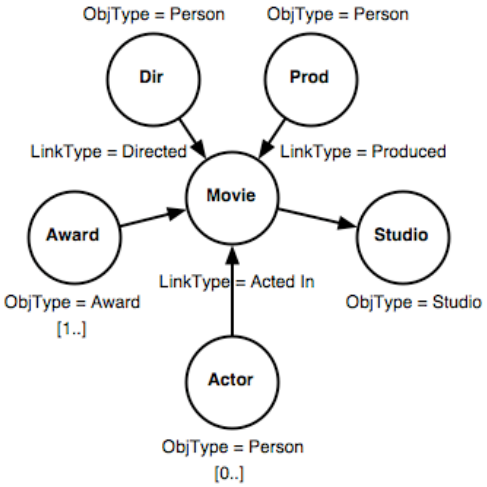

Figure 3: QGraph query for IMDb data.

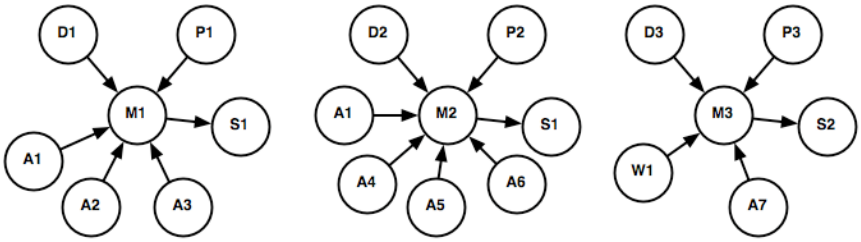

Figure 4: Matches to the query in Figure 3.

Our learning algorithm for relational probability trees constructs trees such as the one shown in Figure 5. The tree represents a series of questions to ask about any subgraph returned by the corresponding query. In this tree, the root node asks whether the movie has more than five actors born after 1943. If so, the subgraph travels down the left-hand branch to a node asking whether the movie at the center of the subgraph is a drama. The subgraph continues moving down appropriate branches of the tree until a leaf node is reached. The leaf nodes contain probability distributions over the values of the *receipts* attribute. Leaf nodes in Figure 5 shows the number of movie subgraphs of each class that reach the leaf, as well as their respective probabilities. The leftmost pair of numbers indicate the number and probability of movies with opening weekend box office receipts exceeding $2 million (*receipts* = *True*). The second numbers indicate the converse (*receipts* = *False*).

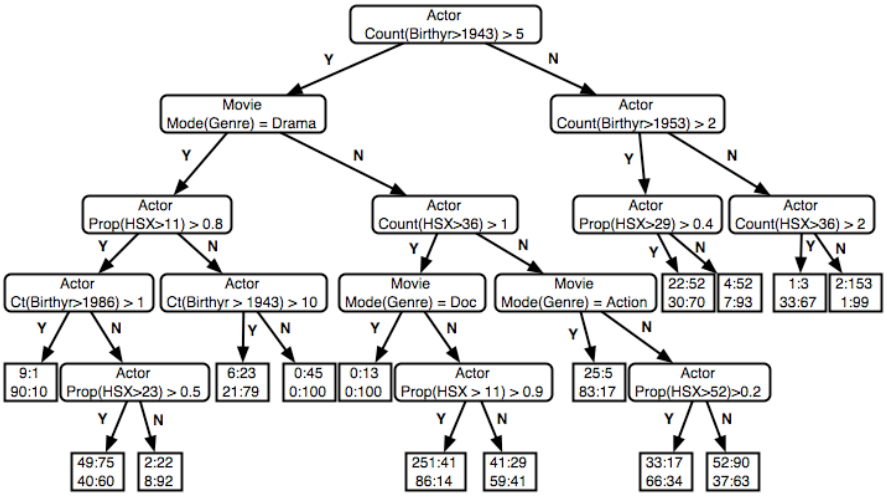

Figure 5: An example relational probability tree (RPT)

Our construction algorithm for RPTs is a recursive partitioning algorithm similar in spirit to CART (Breiman, Friedman, Olshen and Stone 1984), C4.5 (Quinlan 1993), and CHAID (Kass 1980). However, the RPT algorithm searches over the attributes of different object types in the subgraph and multiple methods of aggregating the values of those attributes and creating binary

splits on those aggregated values. For example, for a numeric attribute such as birth year, it searches over splits such as MEAN(birthyr) > x, PROPORTION(birthyr > x) > y, MAXIMUM(birthyr) > y, MINIMUM(birthyr) > x, and COUNT(birthyr > x) > y. Our current approach continues partitioning the training data until a stopping criteria is reached. Our current stopping criteria uses a Bonferroni-adjusted chi-square test analogous to that used in CHAID. However, such methods face a variety of problems due to multiple comparison effects (Jensen and Cohen 2000), and we are exploring the use of randomization tests (Jensen 1992) to better adjust for such effects.

This two-step approach of querying and then learning is necessary because of the semi-structured data model that underlies Proximity. In Proximity's graph database, objects and links are not created with strong type information. Rather, data about each object or link is stored in zero or more attributes, name-value pairs such as <age, 54> or <genre, comedy>. Even type information (e.g., person or movie) is stored as an ordinary attribute without privledged status. As a result, attributes are not constrained to occur in particular combinations, in contrast to more conventional relational databases, where a static schema defines both type information and the fields (attributes) corresponding to each entity or relation type. If such structure is needed in Proximity, it can be imposed by a QGraph query. The labels in a query (e.g., the "movie", "actor", and other labels in Figure 3) are assigned to the matching portions of a subquery and remain on those elements for use by other algorithms such as the RPT construction algorithm. Similarly, we often employ particular schemas (such as the one shown in Figure 1) to aid communication, but this is a convenience, not a necessity.

This high degree of flexibility imposes a performance penalty for querying. However, such flexibility is essential for effective machine learning and data mining. First, practical data mining often involves the creation of many new attributes as a human data analyst tries alternative methods for understanding and modeling the data. Adding many attributes to a conventional database would require constant updates to its schema, a costly operation for traditional relational databases. Second, a particular schema is just one way of interpreting a given data set, and it can bias analysis in important ways. To enable truly effective data mining, analysts must be able to change the schema easily, and thus reconceptualize the domain (Jensen & Neville 2002b; Neville & Jensen 2002).

Comparison and Contrast

Techniques for relational learning can be better understood by examining them in the context of a set of design choices and statistical issues. This section describes several decision choices and the next section covers a small set of unique statistical issues facing relational learning algorithms.

Data characteristics

- *Network size* — Raw size is one of the most obvious methods of characterizing a relational data set. PROXIMITY has been constructed and evaluated on relatively large networks. The

largest data set we have analyzed (on wireless phone fraud) contains nearly 2 million objects and 7 million links. The complete IMDb data set contains over 1.1 million objects and over 3.1 million links. These fairly large data sets contrast with the relatively small networks typically examined by work in social network analysis and inductive logic programming.

- *Connectivity* — The degree of connectivity among different portions of the data graph is another important characteristic of relational data sets. Our work focuses on networks consisting of a small number of large connected components. In contrast, much of the work in ILP and SNA has focused on many small disconnected components, each of which can be considered a data instance. For example, some work in ILP has analyzed the relational structure of molecules to predict their mutagenicity (Srinivasan, Muggleton, Sternberg, and King 1996). Each molecule is considered a single instance for purposes of learning.
- *Homogeneity* — Many techniques that analyze relational data assume the data consist of homogeneous objects. Such networks include sets of web pages, phone numbers, or persons within an organization. In contrast, several recently developed techniques, including our work on RPTs, can analyze sets of relational data with heterogenous objects, such as movies, people, and studios that make up the IMDb data.

Task

- *Level of relational dependence* — The most commonly used modeling techniques from machine learning, data mining, and statistics analyze independent attribute vectors, thus assuming that relational dependencies are unimportant, or at least beyond the scope of analysis. Specialized techniques for spatial and temporal data have been developed that assume a highly regular type of relational dependence. In contrast, the work discussed here addresses relational data sets with potentially irregular relational structure, with variation in the number and type of links among objects, and these variations are assumed to have significance for modeling.
- *Type of task* — Nearly all the algorithms discussed here focus on *supervised learning*. That is, they attempt to predict the value of some attribute whose true value is known in the data set. In contrast, some approaches focus on *unsupervised learning*, where the task is to discern some unknown structure in the data. Clustering algorithms are a form of unsupervised learning, and similar work has recently been undertaken for relational data (e.g., Taskar, Segal, and Koller 2001).
- *Level of determinism* — RPTs, PRMs, and many of the other approaches discussed here attempt to learn *probabilistic* models of relational data. However, some techniques are specially adapted to learning in deterministic domains. For example, such techniques have been applied to chess, learning grammars for artificial and natural languages, and inducing computer programs from examples. Most work in inductive logic programming is focused on deterministic domains, though some recent work extends this work into probabilistic domains (Dzeroski and Lavrac).

- *Locality of inference* — PROXIMITY's combination of querying for subgraphs and learning based on those subgraphs assumes that all relevant relational information is preserved in the portion of the entire data set represented in the subgraph. If important information resides on elements outside the matched subgraph, then the RPT cannot capture it. The subgraph is assumed to represents the relevant "local neighborhood" of an object (e.g., a movie), and more global features of the graph are assumed to be unimportant. Similar locality constraints apply explicitly or implicitly for most techniques, but the degree of these constraints can vary considerably.

Model Representation and Learning

- *Type of model* — To date, we have incorporated modeling algorithms into PROXIMITY that construct conditional or discriminative models. This contrasts with other work focused on constructing generative models. Generative models define a probability distribution over the entire space of data instances. For example, for the problem of predicting the receipts of movies, a generative model would define the probability of all possible movie subgraphs along with a probability distribution over possible values of the receipts attribute. In contrast, a discriminative model defines a probability distribution over the values of receipts, given a particular subgraph. As with other types of Bayesian network models, PRMs are generative models. As with other types of tree-based models, RPTs are discriminative models. Generative models have a wider range of uses (such as detecting anomalies in a data set), provide a more complete description of the dependencies in a data set, and allow for more robust inference in the presence of missing data. However, their accuracy on purely discriminative tasks is often lower than models explicitly learned for that purpose, and they can be more difficult to learn.
- *Search over model structures* — The RPT learning algorithm searches over a wide range of possible structures for the tree and for the attributes included in the tree. In contrast, some approaches to relational learning, including first-order Bayesian networks, PROXIMITY's own relational Bayesian classifer, and other techniques in social network analysis only learn the parameters for a model with fixed structure and attributes.
- *Attribute construction* — RPT learning involves a limited form of attribute construction. Aggregate attributes (e.g., average actor age) are constructed and evaluated when constructing the tree. Some techniques such as ILP offer far more extensive search of such "constructed" attributes, greatly expanding the set of possible models that can be learned (Silverstein and Pazzani 1991). Other techniques do no search whatsoever, relying on the existing attributes on objects and links.
- *Use of background knowledge* — Data analysts often have substantial background knowledge that can greatly assist model construction. Some techniques can used encoded background knowledge in the learning process. For example, background knowledge in first-order logic can be used by ILP approaches to speed and improve learning. Similarly, prior probability distributions can be used in Bayesian learning techniques. To date, PROXIMITY does not employ any explicit form of background knowledge in its learning algorithms.

Statistical Issues

Our recent work on relational learning has concentrated on the unique challenges of learning probabilistic models in relational data. Specifically, we are examining how particular characteristics of relational data affect the statistical inferences necessary for accurate learning. We have identified three features of relational data — concentrated linkage, degree disparity, and relational autocorrelation — and shown how they lead to two pathological behaviors in learning algorithms.

To explain more fully, the relevant features of relational data are:

- *Concentrated linkage* — Real relational data sets can show striking non-uniformities in the concentration of linkage between different types of objects. For example, in our IMDb data, movies are linked to only a single primary studio, and each such studio is typically linked to many movies. We refer to this as *concentrated linkage* (Jensen and Neville 2002a). It contrasts with other situations where a smaller number of movies link to a single object (e.g., directors) or where many movies link to many objects of the same type simultaneously (e.g., actors). Figure 6 shows a schematic of the two situations. We have found concentrated linkage in many relational data sets. Perhaps the best example is publicly traded companies that each link to a single accounting firm, of which there are only a very small number.

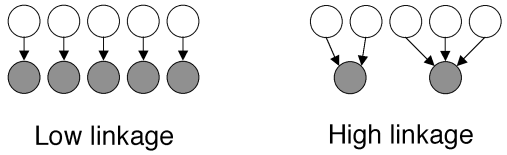

Figure 6: Concentrated linkage

- *Degree disparity* — Another characteristic that occurs in some relational data sets is *degree disparity*. This condition arises when objects of different classes have widely different distributions of degree (the number links to objects of a particular type). For example, in IMDb, we found that US-based studios were systematically linked to a larger number of movies than foreign studios ($p<0.0001$). Figure 7 shows degree disparity schematically. We have found similar degree disparity in other data sets. For example, the number of owners differs systematically among publicly traded companies in different industries and the number of hyperlinks differs systematically among different classes of web pages at university web sites.

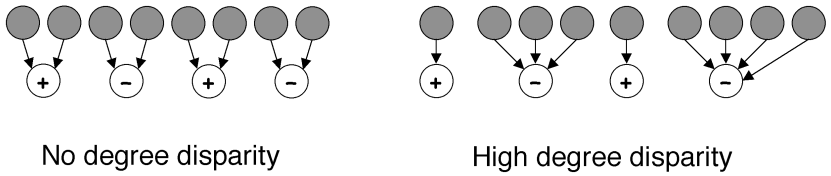

Figure 7: Degree disparity

- *Relational autocorrelation* — Autocorrelation is the correlation among values of the same attribute for related objects. For example, temporal autocorrelation occurs when values of a given attribute (e.g., stock price) at time t tend to correlate highly with the value of the same attribute at time t+1. By analogy, we define *relational autocorrelation* as the correlation among values of given variable on objects that are nearby in graph space (Jensen and Neville 2002a). For example, the box office receipts of a movie tend to be highly correlated with the receipts of other movies made by the same director (correlation coefficient = 0.65) but not for movies starring the same actors (correlation coefficient = 0.17). Figure 8 shows autocorrelation schematically. We have found many other examples of autocorrelation, including correlation of the fraud status of interconnected wireless phone numbers and topics of interconnected web pages.

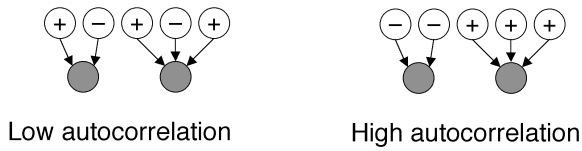

Figure 8: Relational autocorrelation

These three characteristics of relational data can greatly complicate efforts to construct good statistical models. Specifically, they can lead to:

- *Biased feature selection* — Our recent work has shown that high levels of concentrated linkage and relational autocorrelation can cause data mining algorithms to select models that have the weakest, rather than the strongest, statistical support from the data (Jensen and Neville 2002a). This pathology occurs because linkage and autocorrelation combine to reduce the effective sample size of the data, thus increasing the variance of statistics used to assess the relatively utility of different components in learned models. Given that learning algorithms select the best component among many options, they can often select components with high variance, but low true utility, thus reducing the overall accuracy of the resulting model.
- *Spurious correlation* — In other work, we demonstrate a pathology associated with building models that aggregate the values of many objects (e.g., the ages of many actors associated with a movie). This is a common method for simplifying relational data, and it is used in both RPTs

and PRMs. When aggregation is used on data with degree disparity and autocorrelation, it can lead data mining algorithms to include completely spurious elements in their models (Type I errors) and to completely miss very useful elements (Type II errors) (Jensen and Neville, in preparation). These errors occur with degree disparity because many aggregation functions (e.g., Max) will produce apparent correlation between the aggregated values (e.g., maximum movie receipts) and a class label (e.g., studio location) whenever degree disparity occurs, regardless of whether movie receipts has any correlation with studio location.

Both of these effects show the problems associated with violating the assumption of independence among data instances that underlies so many of the techniques common to machine learning, data mining, and statistical modeling techniques. These results imply that new approachhes are necessary to extend current techniques for data mining to relational data. We are developing one potentially promising class of techniques, based on randomization tests and resampling-based methods. We expect that these computationally intensive statistical procedures will allow us to adjust for the unique characteristics of a given relational data set, and make accurate parameter estimates and hypothesis tests. We are incorporating these approaches into our algorithm for constructing relational probability trees. We conjecture that similar approaches will need to be incorporated into all accurate techniques for building statistical models from relational data.

Conclusions

Recent work in machine learning and data mining has made impressive strides toward learning highly accurate models of relational data. However, little of this work has made good use of research in other areas, such as social network analysis and statistics. Cross-disciplinary efforts and joint research efforts should be encouraged to promote rapid development and dissemination of useful algorithms and data representations. In particular, this work should focus on the unique statistical challenges raised by relational data.

References

Blau, H., N. Immerman, and D. Jensen (2002). A Visual Language for Querying and Updating Graphs. University of Massachusetts Amherst Computer Science Technical Report 2002-037.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group.

Dzeroski, S. and N. Lavrac, (Eds.) (2001). *Relational Data Mining.* Berlin: Springer.

Flach, P. and N. Lachiche (1999). 1BC: A first-order Bayesian classifier. Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP'99). S. Dzeroski and Peter Flach (Eds.). Springer. 92-103.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning Probabilistic Relational Models. In IJCAI'99. 1300-1309.

Getoor, L., N. Friedman, D. Koller, and B. Taskar (2001). Learning probabilistic models of relational structure. Proceedings of the Eighteenth International Conference on Machine Learning (ICML).

Gruber, T. (1993). Towards principles for the design of ontologies used for knowledge sharing. *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers. N. Guarino and R. Poli (Eds.).

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* 29:119-127.

Kersting, K. and Luc De Raedt (2000). Bayesian logic programs. Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming. J. Cussens and A. Frisch (Eds.). 138-155.

Jensen, D. (1992). Induction with Randomization Testing. PhD thesis. Washington University. St. Louis, Missouri.

Jensen, D. and P. Cohen (2000). Multiple comparisons in induction algorithms. *Machine Learning* 38: 309-338.

Jensen, D. and J. Neville (2002a). Linkage and autocorrelation cause feature selection bias in relational learning. Proceedings of the 19th International Conference on Machine Learning.

Jensen, D. and J. Neville (2002b). Schemas and models. Proceedings of the Multi-Relational Data Mining Workshop, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Jensen, D. and J. Neville (in preparation). The effect of degree disparity on feature selection in relational learning.

Muggleton, S. (Ed.) (1992). *Inductive Logic Programming.* San Diego, CA: Academic Press.

Neville, J. and D. Jensen (2002). Supporting relational knowledge discovery: Lessons in architecture and algorithm design. Proceedings of the Data Mining Lessons Learned Workshop, 19th International Conference on Machine Learning.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Silverstein, G. and Pazzani, M. (1991). Relational cliches: Constraining constructive induction during relational learning. Proceedings of the Eighth International Workshop on Machine Learning.

Srinivasan, A., S. Muggleton, M. Sternberg, R. King (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence* 85: 277-299.

Taskar, B., E. Segal, and Daphne Koller (2001). Probabilistic classification and clustering in relational data. Proceeding of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01). 870-878.

Wassermann, S. and Faust, K (1994). *Social Network Analysis: Methods and Applications*. Cambridge: Cambridge University Press.

Acknowledgments

This research is supported by the Defense Advanced Research Projects Agency under contract numbers F30602-01-2-0566 and F30602-00-2-0597, respectively. 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 DARPA or the U.S. Government.