Dependency Networks for Relational Data

Dependency Networks for Relational Data

Jennifer Neville, David Jensen
Computer Science Department
University of Massachusetts Amherst
Amherst, MA 01003
{jneville|jensen}@cs.umass.edu

Abstract

Instance independence is a critical assumption of traditional machine learning methods contradicted by many relational datasets. For example, in scientific literature datasets there are dependencies among the references of a paper. Recent work on graphical models for relational data has demonstrated significant performance gains for models that exploit the dependencies among instances. In this paper, we present relational dependency networks (RDNs), a new form of graphical model capable of reasoning with such dependencies in a relational setting. We describe the details of RDN models and outline their strengths, most notably the ability to learn and reason with cyclic relational dependencies. We present RDN models learned on a number of real-world datasets, and evaluate the models in a classification context, showing significant performance improvements. In addition, we use synthetic data to evaluate the quality of model learning and inference procedures.

1. Introduction

Relational data pose a number of challenges for model learning and inference. The data have complex dependencies, both as a result of direct relations (e.g., research paper references) and through chaining multiple relations together (e.g., papers published in the same journal). The data also have varying structure (e.g., papers have different numbers of authors, references and citations). Traditional graphical models such as Bayesian networks and Markov networks assume that data consist of independent and identically distributed instances, which makes it difficult to use these techniques to model relational data that consist of non-independent and heterogeneous instances. Recent research in relational learning has produced several novel types of graphical models to address this issue. Probabilistic relational models (PRMs)[1] (e.g. [5, 15, 14]) estimate joint probability distributions of relational data and have been evaluated successfully in several domains, including the World Wide Web, genomic structures, and scientific literature.

In this paper, we present relational dependency networks (RDNs), an extension of dependency networks [6] for relational data[2]. RDN models are a new form of PRM that offer several advantages over the comparable joint models—relational Bayesian networks (RBNs) [5] and relational Markov networks (RMNs) [15]. Specifically, the strengths of RDNs include: (1) an interpretable representation that facilitates knowledge discovery in relational data, (2) the ability to represent arbitrary cyclic dependencies, including relational autocorrelation, and (3) simple and efficient methods for learning both model structure and parameters.

Graphical models are an attractive modeling tool for knowledge discovery because they are a compact representation of the joint distribution of a set of variables, which allows key dependencies to be expressed and irrelevancies to be ignored [2]. The qualitative properties of the model are encoded in the *structure* of the graph, while the quantitative properties are specified by the *parameters* of the associated probability distributions. The models are often easy to interpret because the graph structure can be used to infer dependencies among variables of interest. PRMs maintain this property as they extend conventional graphical models to relational settings. A compact representation is even more desirable for modeling relational data, because the enormous space of possible dependencies can overwhelm efforts to identify novel, and interesting patterns.

The ability to represent, and reason with, arbitrary cyclic dependencies is another important characteristic of relational models. Relational autocorrelation, a statistical dependency among values of the same variable on related entities [7], is a nearly ubiquitous phenomenon in relational datasets. For example, hyperlinked web pages are more

[1] Several previous papers (e.g., [5]) use the term PRM to refer to a specific type of model that the originators now call a relational Bayesian network (Koller, personal communication). In this paper, we use PRM in its more recent and general sense.

[2] This work generalizes our previous work on simple RDNs for classification [12].

likely to share the same topic than randomly selected pages, and proteins located in the same place in a cell are more likely to share the same function than randomly selected proteins. Recent work has shown that autocorrelation dependencies can be exploited to improve classification accuracy, if inferences about related data instances are made simultaneously (e.g., [3, 15, 12]). However, these relational autocorrelation dependencies are often cyclic in nature, making it difficult to encode these dependencies with directed graphical models such as RBNs unless the autocorrelation can be structured to be acyclic (e.g., with temporal constraints) [5]. In contrast, undirected graphical models such as RMNs, and RDNs, can represent arbitrary forms of relational autocorrelation.

During learning, relational models consider a large number of features, thus simple and efficient learning techniques are advantageous, particularly for joint models. The RDN learning algorithm is based on pseudolikehood techniques [1], which estimate a set of conditional distributions independently. This approach avoids the complexities of estimating a full joint distribution and can incorporate existing techniques for learning conditional probability distributions of relational data. Relatively efficient techniques exist for learning both the structure and parameters of RBN models but the acyclicity constraints of the model precludes the learning of arbitrary autocorrelation dependencies. On the other hand, while in principle it is possible for RMN techniques to learn cyclic autocorrelation dependencies, inefficiencies due to modeling the full joint distribution make this difficult in practice. The current implementation of RMNs is not capable of learning model structure automatically, nor can it automatically identify which features are most relevant to the task; research has focused primarily on parameter estimation and inference procedures. To our knowledge, RDNs are the first PRM capable of *learning* cyclic autocorrelation dependencies.

We begin by reviewing the details of dependency networks for propositional data and then describe how to extend them to a relational setting. Next, we present RDN models learned from four real-world datasets and evaluate the models in a classification context, demonstrating equivalent, or better, performance in comparison to conditional models. Finally, we report experiments on synthetic data that show model learning and inference is robust to varying levels of autocorrelation and that accurate models can be learned from small training set sizes.

2. Dependency Networks

Graphical models represent a joint distribution over a set of variables. The primary distinction between Bayesian networks, Markov networks and dependency networks (DNs) [6] is that dependency networks are an approximate representation. DNs approximate the joint distribution with a set of conditional probability distributions (CPDs), which are learned independently. This approach to learning is a relatively simple technique for parameter estimation and structure learning that results in significant efficiency gains over exact models. However, because the CPDs are learned independently, DN models are not guaranteed to specify a *consistent* joint distribution. This precludes DNs from being used to infer causal relationships and limits the applicability of exact inference techniques. Nevertheless, DNs can encode predictive relationships (i.e. dependence and independence) and Gibbs sampling (e.g. [11]) inference techniques can be used to recover a full joint distribution, regardless of the consistency of the local CPDs.

DN Representation. A DN encodes probabilistic relationships among a set of variables in a similar manner to Bayesian and Markov networks, combining characteristics of both undirected and directed models. DN models consists of a graph G, which encodes the *structure* of the model, and a set of probability distributions P, which encode the *parameters* of the model. Dependencies among variables are represented with a bidirected graph $G = (V, E)$, where conditional independence is interpreted using graph separation, as with undirected models. However, as with directed models, dependencies are quantified with a set of conditional probability distributions P. Consider the set of variables $\mathbf{X} = (X_1, ..., X_n)$ over which we'd like to model the joint distribution $p(\mathbf{x}) = p(x_1, ..., x_n)$. Each node $v_i \in V$ corresponds to an $X_i \in \mathbf{X}$ and is associated with a probability distribution conditioned on the other variables, $P(v_i) = p(x_i|\mathbf{x} - \{x_i\})$. The parents, pa_i, of node i are the set of variables that render X_i conditionally independent of the other variables ($p(x_i|pa_i) = p(x_i|\mathbf{x} - \{x_i\})$) and G contains a directed edge from each parent node v_j to each child node v_i ($e(v_j, v_i) \in E$ iff $x_j \in pa_i$).

DN Learning. Both the structure and parameters of DN models are determined through learning a set of local conditional probability distributions (CPDs). The DN learning algorithm learns a separate CPD for each variable X_i, conditioned on the other variables in the data ($\mathbf{X} - \{X_i\}$). Any conditional learner can be used for this task (e.g. logistic regression, decision trees). The learned CPD is included in the model as $P(v_i)$, and the variables selected by the conditional learner (e.g., $x_i = \alpha x_j + \beta x_k$) form the parents of X_i (e.g., $pa_i = \{x_j, x_k\}$), which is then reflected in the edges of G appropriately. If the conditional learner is not selective, the DN model will be fully connected (i.e., $pa_i = \mathbf{x} - \{x_i\}$). In order to build understandable DNs, it is desirable to use a selective learner that will learn CPDs that use a subset of all variables.

DN Inference. Although the DN approach to structure learning is simple and efficient, it can result in an inconsistent network, both structurally and numerically. In other

words, there may be no joint distribution from which each of the CPDs can be obtained using the rules of probability. For example, a network that contains a directed edge from X_i to X_j, but not from X_j to X_i, is inconsistent—X_i and X_j are dependent but X_j is not represented in the CPD for X_i. A DN is consistent if the conditional distributions in P factor the joint distribution—in this case we can compute the joint probability for a set of values $\mathbf{x}$ directly. In practice, [6] show that DNs will be nearly consistent if learned from large data sets, since the data serve a coordinating function that ensures consistency among the CPDs. If a DN is inconsistent, approximate inference techniques can be used to estimate the full joint distribution and extract probabilities of interest. Gibbs sampling (e.g., [11]) can be used to recover a full joint distribution for $\mathbf{X}$, regardless of the consistency of the local CPDs, provided that each X_i is discrete and each local CPD is positive [6].

3. Relational Dependency Networks

RDNs extend dependency networks to a relational setting. DNs have been shown to perform comparably to BNs for a number of propositional tasks [6], thus we expect they will achieve similar performance levels in relational settings. Also, several characteristics of DNs are particularly desirable for modeling relational data. First, learning a collection of conditional models offers significant efficiency gains over learning a full joint model—this is generally true, but is even more pertinent to relational settings where the feature space is very large. Second, networks that are easy to interpret and understand aid analysts' assessment of the utility of the relational information. Third, the ability to represent cycles in a network facilitates reasoning with relational autocorrelation, a common characteristic of relational data. Finally, while the need for approximate inference is a disadvantage of DNs for propositional data, due to the complexity of relational model graphs in practice, all PRMs use approximate inference.

RDNs extend DNs for relational data in the same way that RBNs [5] extend Bayesian networks and RMNs [15] extend Markov networks. We describe the general characteristics of PRMs and then discuss the details of RDNs.

3.1. Probabilistic Relational Models

PRMs represent a joint probability distribution over a relational dataset. When modeling attribute-value data with graphical models, there is a single graph G that is associated with the model M. In contrast, there are three graphs associated with models of relational data: the *data graph* G_D, the *model graph* G_M, and the *inference graph* G_I.

First, the relational dataset is represented as a typed, attributed graph $G_D = (V_D, E_D)$. For example, consider the data graph in figure 1a. The nodes V_D represent objects in the data (e.g., authors, papers) and the edges E_D represent relations among the objects (e.g., author-of, cites). We use rectangles to represent objects, circles to represent random variables, dashed lines to represent relations, and solid lines to represent probabilistic dependencies. Each node $v_i \in V_D$ is associated with a type $T(v_i) = t_{v_i}$ (e.g., *paper*). Each object type $t \in T$ has a number of associated attributes $\mathbf{X^t} = (X_1^t, ..., X_n^t)$ (e.g., topic, year). Consequently, each object v_i is associated with a set of attribute values determined by its type $\mathbf{X}_{\mathbf{v_i}}^{\mathbf{t_{v_i}}} = (X_{v_i 1}^{t_{v_i}}, ..., X_{v_i n}^{t_{v_i}})$. A PRM model represents a joint distribution over the values of the attributes throughout the data graph, $\mathbf{x} = \{\mathbf{x}_{\mathbf{v_i}}^{\mathbf{t_{v_i}}} : v_i \in V, t_{v_i} = T(v_i)\}$.

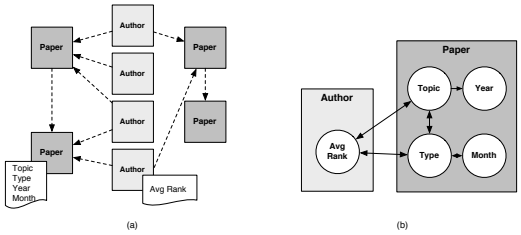

Figure 1. PRM (a) data, and (b) model graph.

The dependencies among attributes are represented in the model graph $G_M = (V_M, E_M)$. Attributes of an object can depend probabilistically on other attributes of the same object, as well as on attributes of other related objects in G_D. For example, the topic of a paper may be influenced by attributes of the authors that wrote the paper. Instead of defining the dependency structure over attributes of specific objects $\mathbf{X_v}$, PRMs define a generic dependency structure at the level of object types. The set of attributes $\mathbf{X_k^t} = (X_{v_i k}^t : v_i \in V,\ T(v_i) = t)$ is tied together and modeled as a single variable. Each node $v_i \in V_M$ corresponds to an $X_k^t, t \in T \wedge X_k^t \in \mathbf{X^t}$. As in conventional graphical models, each node is associated with a probability distribution conditioned on the other variables. Parents of X_k^t are either: (1) other attributes associated with type t_k (e.g., paper *topic* depends on paper *type*), or (2) attributes associated with objects of type t_j where objects t_j are related to objects t_k in G_D (e.g., paper *topic* depends on author *rank*). For the latter type of dependency, if the relation between t_k and t_j is one-to-many, the parent consists of a set of attribute values (e.g., author ranks). In this situation, PRMs use aggregation functions, either to map sets of values into single values, or to combine a set of probability distributions into a single distribution.

For example, consider the model graph in figure 1b. It models the data in figure 1a, which has two object types: paper and author. In G_M, each object type is represented by a plate, and each attribute of each object type is represented as a node. The edges of G_M characterize the dependencies among the attributes at the type level.

During inference, a PRM uses the G_M and G_D to in-

stantiate an inference graph $G_I = (V_I, V_E)$ in a process sometimes called "rollout". The rollout procedure used by PRMs to produce the G_I is nearly identical to the process used to instantiate models such as hidden Markov models (HMMs), and conditional random fields (CRFs) [9]. G_I represents the probabilistic dependencies among all the object variables in a single test set (here G_D is different from the G'_D used for training). The structure of G_I is determined by both G_D and G_M—each object-attribute pair in G_D gets a separate, local copy of the appropriate CPD from G_M. The relations in G_D constrain the way that G_M is rolled out to form G_I. PRMs can produce inference graphs with wide variation in overall and local structure, because the structure of G_I is determined by the specific data graph, which typically has non-uniform structure. For example, figure 2 shows the PRM from figure 1b rolled out over a data set of three authors and three papers, where P_1 is authored by A_1 and A_2, P_2 is authored by A_2 and A_3, and P_3 is authored by A_3. Notice that there is a variable number of authors per paper. This illustrates why PRM CPDs must aggregate—for example, the CPD for paper-type must be able to deal with a variable number of author ranks.

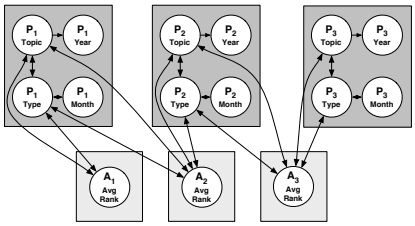

Figure 2. Example PRM inference graph.

3.2. RDN Models

An RDN model encodes probabilistic relationships in a similar manner to DN models, extending the representation to a relational setting. RDNs use a bidirected model graph G_M with a set of conditional probability distributions P. Each node $v_i \in V_M$ corresponds to an $X_k^t \in \mathbf{X^t}$, $t \in T$ and is associated with a conditional distribution $p(x_k^t \mid pa_{x_k^t})$. Figure 1b illustrates an example RDN model graph for the data graph in figure 1a. The graphical representation illustrates the qualitative component (G_D) of the RDN—it does not depict the quantitative component (P) of the model, which includes aggregation functions. The representation uses a modified plate notation; dependencies among attributes of the same object are contained inside the rectangle and arcs that cross the boundary of the rectangle represent dependencies among attributes of related objects. For example, *month*$_i$ depends on *type*$_i$, while *avgrank*$_j$ depends on the *type*$_k$ and *topic*$_k$ for all papers k related to author j in G_D. Although conditional independence is infered using an undirected view of the graph, bidirected edges are useful for representing the set of variables in each CPD. For example, in figure 1b, the CPD for *year* contains *topic* but the CPD for *topic* does not contain *type*. This shows inconsistencies that may result from the RDN learning technique.

Learning. The RDN learning algorithm is much like the DN learning algorithm, except we use a selective relational classification algorithm to learn a set of conditional models. The algorithm input consists of: (1) a data graph G_D, with a set of types T and attributes X, (2) a conditional relational learner R, and (3) a search limit c, which limits the length of paths in G_D that are considered in R. For each $t \in T$, and each $X_k^t \in \mathbf{X^t}$, the algorithm uses R to learn a CPD for X_k^t given the set of attributes $\{X_{k' \neq k}^t\} \cup \mathbf{X^{t' \neq t}}$, where t' is up to c links away from t in G_D. The resulting CPDs are included in P and are used to form G_M.

We use relational probability trees (RPTs) [13] for the CPD components of the RDN. The RPT learning algorithm adjusts for biases towards particular features due to degree disparity and autocorrelation in relational data [7, 8]. We have shown that RPTs build significantly smaller trees than other conditional models and achieve equivalent, or better, performance. This characteristic of the RPTs is crucial for learning understandable RDN models. The collection of RPTs will be used during inference so the size of the models also has a direct impact on efficiency. We expect that the general properties of RDNs would be retained if other approaches to learning conditional probability distributions were used instead, given that those approaches are both selective and accurate.

RPTs extend probability estimation trees to a relational setting. RPT models estimate probability distributions over class label values in the same manner as conventional classification trees, but the algorithm looks beyond the attributes of the item for which the class label is defined and considers the effects of attributes in the local relational neighborhood ($\leq c$ links away) on the probability distribution. The RPT algorithm automatically constructs and searches over aggregated relational features to model the distribution of the target variable—for example, to predict the value of an attribute (e.g., paper topic) based on the attributes of related objects (e.g., characteristics of the paper's references), a relational feature may ask whether the oldest reference was written before 1980.

Inference. The RDN inference graph G_I is potentially much larger than the original data graph. To model the full joint distribution there must be a separate node (and CPD) for each attribute value in G_D. To construct G_I, the set of template CPDs in P is rolled-out over the data graph. Each object-attribute pair gets a separate, local copy of the appropriate CPD. Consequently, the total number of nodes in the inference graph will be $\sum_{v \in V_D} |\mathbf{X^{T(v)}}|$. Rollout facilitates generalization across data graphs of varying size—we can learn the CPD templates from one data graph and apply

the model to a second data graph with a different number of objects by rolling out more CPD copies.

We use Gibbs sampling (e.g. [11]) for inference in RDN models. To estimate a joint distribution, the inference graph consists of a rolled-out network with unobserved variables. The values of all unobserved variables are initialized to values drawn from their prior distributions. Gibbs sampling then iteratively relabels each unobserved variable by drawing from its local conditional distribution, given the current state of the rest of the graph. After a sufficient number of iterations, the values will be drawn from a stationary distribution and we can use the samples to estimate probabilities of interest. For the experiments reported in this paper we use a fixed-length chain of 2000 samples (each iteration relabels every value sequentially), with a burn-in of 200.

4. Experiments

The experiments in this section are intended to demonstrate the utility of RDNs as a joint model of relational data. We learn RDN models of four real world datasets to illustrate the types of domain knowledge that can be garnered. In addition, we evaluate the models in a classification context, where only a single attribute is unobserved in the test set, and report significant performance gains compared to a conditional model. Finally, we use synthetic data to assess the impact of training set size and autocorrelation on RDN learning and inference, showing that accurate models can be learned at small data set sizes and that the model is robust to all but extreme levels of autocorrelation. For these experiments, we used the parameters $R = RPT$ and $c = 2$. The RPT algorithm used *MODE*, *COUNT* and *PROPORTION* features with 10 thresholds per attribute.

The RDN models in figures 3-5 continue with the RDN representation introduced in figure 1. Each object type is represented in a separate plate, arcs inside a plate indicate dependencies among the attributes of a single object and arcs crossing the boundaries of plates indicate dependencies among attributes of related objects. An arc from x to y indicates the presence of one or more features of x in the RPT learned for y.

When the dependency is on attributes of objects more than a single link away, the arc is labeled with small rectangle to indicate the intervening object type. For example, movie genre is influenced by the genres of other movies made by the movie's director, so the arc would be labeled with a small *D* rectangle.

In addition to dependencies among attribute values, RPTs also learn dependencies between the structure of the relations (edges in G_D) and the attribute values. This *degree* relationship is represented by a small black circle in the corner of each plate, arcs from this circle indicate a dependency between the number of related objects and an attribute value of a related object. For example, movie receipts is influenced by the number of actors in the movie.

4.1. RDN Models

The first data set is drawn from the Internet Movie Database (www.imdb.com). We collected a sample of 1,382 movies released in the United States between 1996 and 2001. In addition to movies, the data set contains objects representing actors, directors, and studios. In total, this sample contains approximately 42,000 objects and 61,000 links. We learned a RDN model for ten discrete attributes including actor gender and movie opening weekend receipts (*>$2million*). Figure 3 shows the resulting RDN model. Four of the attributes, movie receipts, movie genre, actor birth year, and director 1^{st} movie year, exhibit autocorrelation dependencies. Exploiting this type of dependency has been shown to significantly improve classification accuracy of RMNs compared to RBNs which cannot model cyclic dependencies [15]. However, to exploit autocorrelation the RMN must be instantiated with a corresponding clique template—the dependency must be pre-specified by the user. To our knowledge, RDNs are the first PRM capable of *learning* this type of autocorrelation dependency.

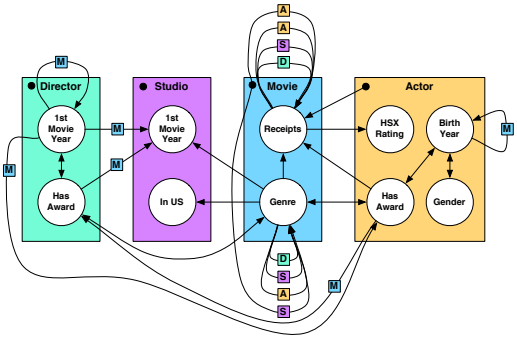

Figure 3. Internet Movie database RDN.

The second data set is drawn from Cora, a database of computer science research papers extracted automatically from the web using machine learning techniques [10]. We selected the set of 4,330 machine-learning papers along with associated authors, cited papers, and journals. The resulting collection contains approximately 13,000 objects and 26,000 links. We learned an RDN model for seven attributes including paper topic (e.g., neural networks) and journal name prefix (e.g., IEEE). Figure 4 shows the resulting RDN model. Again we see that four of the attributes exhibit autocorrelation. In particular, notice that the topic of a paper depends not only on the topics of other papers that it cites, but also on the topics of other papers written by the authors. This model is a good reflection of our domain knowledge about machine learning papers.

The third data set was collected by the WebKB Project [4]. The data consist of a set of 3,877 web pages

from four computer science departments. The web pages have been manually labeled with the categories: course, faculty, staff, student, research project, or other. The collection contains approximately 4,000 web pages and 8,000 hyperlinks among those pages. We learned an RDN model for four attributes of the web pages including school and page label. Figure 5a shows the resulting RDN model.

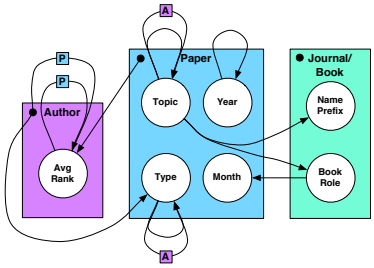

Figure 4. Cora machine-learning papers RDN.

The fourth data set is a relational data set containing information about the yeast genome at the gene and the protein level (www.cs.wisc.edu/∼dpage/kddcup2001/). The data set contains information about 1,243 genes and 1,734 interactions among their associated proteins. We learned an RDN model for seven attributes. The attributes of the genes included protein localization and function, and the attributes on the interactions included type and level of expression. Figure 5b shows the resulting RDN model.

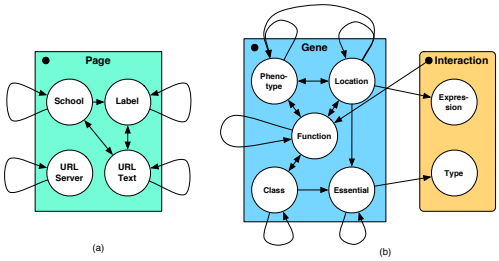

Figure 5. (a) WebKB, and (b) gene data RDNs.

4.2. Classification Experiments

We evaluate the learned models on classification tasks in order to assess (1) whether autocorrelation dependencies among instances can be used to improve model accuracy, and (2) whether the RDN models, using Gibbs sampling, can effectively infer labels for a network of instances. To do this, we compare three models. The first model is a conventional RPT model—an individual classification model that reasons about each instance independently from other instances and thus does not use the class labels of related instances. The second model is a RDN model that exploits additional information available in labels of related instances and reasons about networks of instances collectively. The third model is a probabilistic ceiling for the RDN model. We use the RDN model but allow the true labels of related instances to be used during inference. This model shows the level of performance possible if the RDN model could infer the true labels of related instances with perfect accuracy.

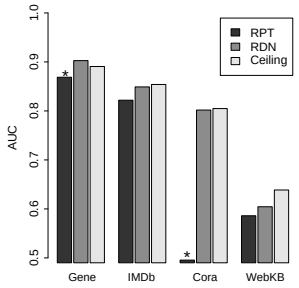

Figure 6. AUC results for classification tasks.

Figure 6 shows area under the ROC curve (AUC) results for each of the three models on four classification tasks. We used the following prediction tasks: IMDb: movie receipts, Cora: paper topic, WebKB: page label, Gene: gene location. The graph shows AUC for the most prevalent class, averaged over a number of training/test splits. For IMDb and Cora, we used 4-5 temporal samples where we learned models on one year of data and applied the model to the subsequent year. For WebKB, we used cross-validation by department, learning on three departments and testing on pages from the fourth held out department. For Gene there was no clear sampling choice, so we used ten-fold cross validation on random samples of the genes. We used two-tailed, paired t-tests to assess the significance of the AUC results obtained from the trials. The t-tests compare the RDN results to each of the other two models. The null hypothesis is that there is no difference in the AUC results of the two models; the alternative is that there is a difference. The differences in performance that are significant at a $p < 0.05$ level are reported in the graph with asterisks.

On three of the tasks, the RDNs models achieve comparable performance to the ceiling models and on the fourth (WebKB) the difference is not statistically significant. This indicates that the RDN model realized its full potential, reaching the same level of performance as if it had access to the true labels of related movies. On the Gene data, the RDN surpasses the performance of the ceiling model, but is only a probabilistic ceiling—the RDN may perform better if an incorrect prediction for one object improves the classification of related objects. Also, the performance of the RDN models is superior to RPT models on all four tasks. This indicates that autocorrelation is both present in the data and identified by the RDN models. The performance improvement over RPTs is due to successful exploitation of this autocorrelation. On the Cora data, the RPT model performance is no better than random because autocorrelation is the only predictor of paper topic (see figure 4).

4.3. Synthetic Data Experiments

To explore the effects of training set size and autocorrelation on RDN learning and inference, we generated homogeneous data graphs with a regular square lattice structure. With the exception of objects along the outer boundary, each object in the lattice links to four immediate neighbors positioned above, below, left, and right. The first and last row and column make up the *frame* of the lattice. In order to control for effects of varying structure, objects in the frame are not used during learning and inference, although their attribute values are available to objects in the core for learning and inference. Thus, training or test sets of size N correspond to a lattice of $(\sqrt{N}+2)^2$ objects, and models are trained or evaluated on the N objects in the core of the lattice. Each object has four boolean attributes X_1, X_2, X_3 and X_4. We use a simple RDN where X_1 is autocorrelated (through objects one link away), X_2 depends on X_1, and the other two attribute have no dependencies.

We generated the values of attributes using the RDN in the following way. We begin by assigning each object in the lattice an initial value for X_1 with $P(x_1 = 1) = 0.5$. We then perform Gibbs sampling over the entire lattice to estimate the values of X_1 conditioned on neighboring values. The values assigned to each object after 200 iterations are used as the final labels. We use a manually specified RPT that assigns X_1 values to each object based on the X_1 values of objects one link away in G_D. The parameters of this model are varied to produce different levels of autocorrelation in X_1. Once X_1 values are assigned, values for X_2 are randomly drawn from a distribution conditioned on objects' X_1 values. We used the parameters $p(x_2 = 1) = 0.3$ and $p(x_1 = 1|x_2 = 1) = 1 - p(x_1 = 0|x_2 = 1) = 0.9$. Finally, random values are assigned to the two other attributes with $p(x_3 = 1) = p(x_4 = 1) = 0.5$. Once a dataset is generated, we measure the proportion of objects with $X_1 = 1$, and any dataset with a value outside the range $[0.4, 0.6]$ is discarded and replaced by a new dataset. This ensures consistency in the distribution of X_1 across datasets and reduces variance in estimated model performance.

The first set of synthetic experiments examines the effectiveness of the RDN learning algorithm. Figure 7a graphs the log-likelihood of learned models as a function of training set size. Training set size was varied at the following levels $\{25, 49, 100, 225, 484, 1024, 5041\}$. Figure 7b graphs log-likelihood as a function of autocorrelation. Autocorrelation was varied to approximate the following levels $\{0.0, 0.25, 0.50, 0.75, 1.0\}$. (We graph the average autocorrelation for each set of trials, which is within 0.02 of these numbers.) At each data set size (autocorrelation level), we generated 25 training sets and learned RDNs. Using each learned model, we measured the average log-likelihood of another 25 test sets (size 225). Figure 7 plots these measurements as well as the log-likelihood of the test data from the RDN used for data generation. These experiments show that the learned models are a good approximation to the true model by training set size 1000, and that RDN learning is robust with respect to varying levels of autocorrelation.

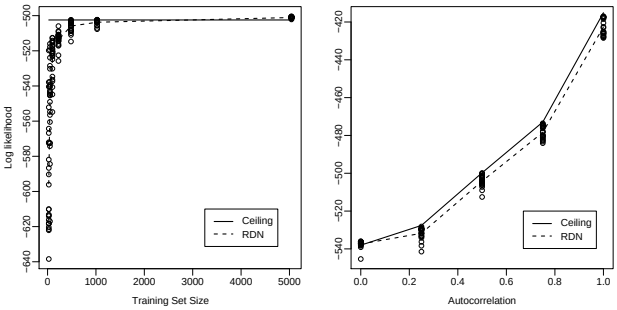

Figure 7. Evaluation of RDN learning.

The second set of synthetic experiments evaluates the RDN inference procedure in a classification context, where only a single attribute is unobserved in the test set. We generated data in the manner described above, learned an RDN for X_1, used the learned models to infer the class labels of unseen test sets, and measured AUC to evaluate the predictions. These experiments compared the same three models as section 4.2, and used the same training set sizes and autocorrelation levels outlined above. At each data set size (autocorrelation level), we generated 25 training and test set pairs, learned the model on the training set, and inferred labels for the test set.

Figure 8a graphs AUC as a function of training set size for RDNs compared to RPTs and the ceiling, plotting the average AUC for each model type. Even at small data set sizes the RDN performance is close to optimal and significantly higher than the performance of the RPTs. Surprisingly, the RPTs are able to achieve moderately good results even without the class labels of related instances. This is because the RPTs are able to use the attribute values of related instances as a surrogate for autocorrelation.

Figure 8b plots average AUC as a function of autocorrelation for RDNs compared to RPTs and the ceiling. When there is no autocorrelation, the RPT models perform optimally. In this case, the RDNs are slightly biased due to excess structure. However, as soon as there is minimal autocorrelation, the RDN models start to outperform the RPTs. At the other extreme, when autocorrelation is almost perfect the RDNs experience a large drop in performance. At this level of autocorrelation, the Gibbs sampling procedure can easily converge to a labeling that is "correctly" autocorrelated but with opposite labels. Although all 25 trials appeared to converge, half performed optimally and the other half performed randomly (AUC ≈ 0.50). Future work will explore ways to offset this drop in performance. However, the utility of RDNs for classification is clear in the

range of autocorrelations that have been observed empirically [0.25,0.75].

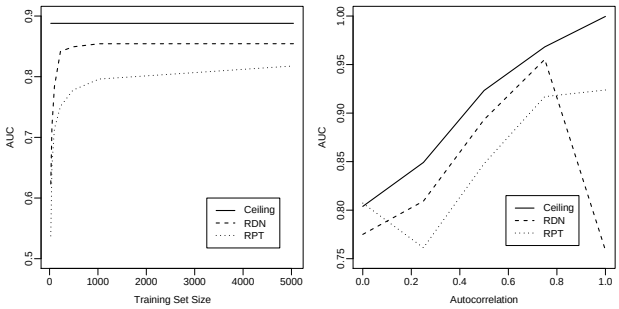

Figure 8. Evaluation of RDN inference.

5. Discussion and Conclusions

In this paper we presented RDNs, a new form of PRM. The primary advantage of RDN models is the ability to learn and reason with relational autocorrelation. We showed the RDN learning algorithm to be a relatively simple method for learning the structure and parameters of a probabilistic graphical model. In addition, RDNs allow us to exploit existing techniques for learning conditional probability distributions. Here we have chosen to exploit our prior work on RPTs, which constructs parsimonious models of relational data, but we expect that the general properties of RDNs would be retained if other approaches to learning conditional probability distributions were used, given that those approaches are both selective and accurate.

The results of the real and synthetic data experiments indicate that collective classification with RDNs can offer significant improvement over conditional approaches to classification when autocorrelation is present in the data—a nearly ubiquitous characteristic of relational datasets. The performance of RDNs also approaches the performance that would be possible if all the class labels of related instances were known. Future work will compare RDN models to RMN models in order to better assess the quality of the pseudolikelihood approximation of the joint distribution. In addition, we are exploring improved inference procedures that consider the autocorrelation dependencies in the data in order to improve inference accuracy and efficiency.

Acknowledgments

The authors acknowledge the invaluable assistance of M. Cornell, A. Shapira, M. Rattigan, M. Hay, B. Gallagher, and helpful comments from A. McGovern, A. Fast, C. Loiselle, and S. Macskassy. This research is supported under a AT&T Labs Graduate Research Fellowship and by NSF and AFRL under contract numbers F30602-00-2-0597, EIA9983215 and F30602-01-2-0566.

References

[1] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:3:179–195, 1975.

[2] W. Buntine. Graphical models for discovering knowledge. In *Advances In Knowledge Discovery And Data Mining*. AAAI Press/The MIT Press, 1996.

[3] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[4] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 509–516, 1998.

[5] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Relational Data Mining*. Springer-Verlag, 2001.

[6] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[7] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 259–266, 2002.

[8] D. Jensen and J. Neville. Avoiding bias when aggregating relational data with degree disparity. In *Proceedings of the 20th International Conference on Machine Learning*, pages 274–281, 2003.

[9] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[10] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 662–667, 1999.

[11] R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Dept of Computer Science, University of Toronto, 1993.

[12] J. Neville and D. Jensen. Collective classification with relational dependency networks. In *Proceedings of the 2nd Multi-Relational Data Mining Workshop, KDD2003*, pages 77–91, 2003.

[13] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, 2003.

[14] S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 992–1002, 2003.

[15] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.