Why Collective Inference Improves Relational Classification

David Jensen, Jennifer Neville, and Brian Gallagher
Dept. of Computer Science
Univ. of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01003-9264
{jensen, jneville, bgallag}@cs.umass.edu

ABSTRACT

Procedures for *collective inference* make simultaneous statistical judgments about the same variables for a set of related data instances. For example, collective inference could be used to simultaneously classify a set of hyperlinked documents or infer the legitimacy of a set of related financial transactions. Several recent studies indicate that collective inference can significantly reduce classification error when compared with traditional inference techniques. We investigate the underlying mechanisms for this error reduction by reviewing past work on collective inference and characterizing different types of statistical models used for making inference in relational data. We show important differences among these models, and we characterize the necessary and sufficient conditions for reduced classification error based on experiments with real and simulated data.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.5.1 [**Pattern Recognition**]: Models.

General Terms

Algorithms, Performance, Design, Theory.

Keywords

Relational learning, probabilistic relational models, collective inference.

1. INTRODUCTION

Recent research in relational learning has produced several novel types of statistical models. These models estimate conditional and joint probability distributions for graph-structured data [1,4,12,15]. Researchers have evaluated their performance on several domains, including classifying web pages [1,15], tracking communicable diseases [6], and identifying topics in scientific literature [12].

Some of this work focuses on *collective inference* — procedures that make simultaneous statistical judgments about the same variables for a set of related data instances. Collective inference can exploit relational autocorrelation, a widely observed characteristic of relational data in which the value of a variable for one instance is highly correlated with the value of the same variable on another instance [7]. Several studies [1,10,16] have shown that, by making inferences about multiple data instances simultaneously, collective inference can significantly reduce classification error.

In this paper, we show that the reduced error attributed to collective inference results from a clever factoring of the space of possible statistical dependencies in relational data. This factoring produces relational models with a parameter space only incrementally larger that of their non-relational counterparts, and thus the variance component of their error is roughly equivalent. When relational information is not informative, the bias component of their error is identical to those of non-relational models, but when relational information *is* informative, bias is vastly reduced. Thus, the increased algorithmic complexity of collective inference purchases a large increase in representational power at minimum cost. Relational models that do not exploit collective inference generally have much larger parameter spaces and require much larger data samples to learn relational models reliably.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD '04, August 22–25, 2004, Seattle, Washington, USA.
Copyright 2004 ACM 1-58113-888-1/04/0008…$5.00.

2. PROBABILISTIC RELATIONAL MODELS

Traditional graphical models such as Bayesian networks and dependency networks assume that data consist of independent and identically distributed (i.i.d.) instances, and inference procedures for these models instantiate a separate network for each data instance. No dependencies run between the networks, because of the assumption that data instances are independent.

Algorithms for constructing probabilistic relational models (PRMs) [4,12,15] remove the independence assumption, and instantiate a single network that represents dependencies both within and between the data instances in a given test set. This procedure — often called *rollout* — is common to graphical models that assume some form of instance dependence, including PRMs, HMMs, and others. Examples of PRMs include relational Bayesian networks (RBNs) [4], relational Markov networks (RMNs) [15], relational dependency networks (RDNs) [12], and Bayesian logic programs (BLPs) [8].

2.1 Types of Relational Models

The process of rollout is sufficiently general that it can be applied to a wide variety of graphical models for relational data. Each type of model imposes different constraints on the

possible dependencies in the network. Only some of these types make collective inference possible.

For example, consider a data graph with a regular structure of n^2 objects arranged in an $n \times n$ lattice. Each object in the lattice links to each of its immediate neighbors. With the exception of objects along the outer boundary, each object links to four others positioned above, below, left, and right. All links are undirected. Each object is characterized by a set of variables that includes a single probabilistic variable C (a class label) and several other variables A_i (one or more attributes) whose values are known with certainty. The task is to construct a joint model of the probability distribution over all the values of the class labels.

Given this task, multiple models could be used to infer the values of C. We will focus on five such models in our experiments:

Intrinsic — For a given object, the *Intrinsic* model estimates the joint distribution of the class label and attributes on that object. It assumes that objects are i.i.d., and thus corresponds to traditional models used in many knowledge discovery applications. The model is depicted graphically in figure 1a using the plate notation common in the graphical modeling community. The inner box, along with the edge connecting A and C, indicates that m different versions of node A (corresponding to m attributes A_i) each depend on C. The outer box indicates that the model creates N different versions of the network, each containing a single node C. For example, this model would indicate that the words on a web page (the attributes A_i) depend only on the topic of that page (C) and are independent of the topic and words on any other page.

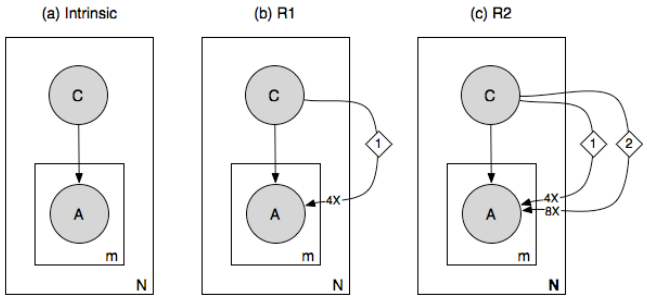

Figure 1: Relational models

Relational 1 (R1) — The model *R1* is a simple relational model indicating that the attributes of an object depend on the class label of that object as well as the class labels of objects one link away. Figure 1b shows this model using a modified plate notation in which the integer within the diamond-shaped annotation ("1") indicates the graph distance of neighboring objects and the multiplier on the edge ("4x") indicates the number of such neighboring objects. The path of the annotated edge outside the outer box emphasizes the dependence on the class labels of adjoining objects. Here, the value of each A_i depends on five different parents C, four of which are from neighboring objects. For example, this model would indicate that the words on a web page depend on the topic of that page and the topics of four adjoining pages.

Relational 2 (R2) — A somewhat more complex relational model *R2* indicates that the attributes of an object depend on the class label of that object and the class labels of objects up to two links away (Figure 1c).

None of these three models allows interdependence among class labels, which is a prerequisite for collective inference. We examine two additional models that do allow for such dependence:

Collective Inference (CI) — The model *CI*, shown in figure 2a, provides the same type of dependence as *Intrinsic*, but adds dependence between the class label of an object and the class label of adjoining objects. This is equivalent to specifying that the topics of web pages depend on those of adjoining pages (and also determine the words on the page).

Relational Collective Inference (RCI) — The model *RCI*, shown in figure 2b, extends the *R1* model by adding dependence among class labels of neighboring objects one link away.

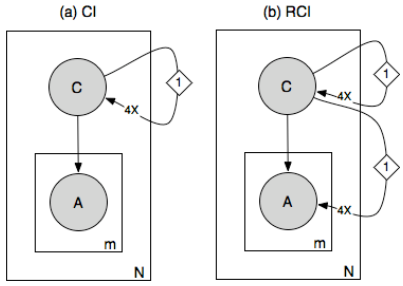

Figure 2: Collective models

These models are relatively simple because the example data are highly regular and contain only a single object and link type. More heterogeneous data might require models with longer and more complex paths among objects. For example, paths connecting autocorrelated objects might pass through one or more intervening objects of specified types. However, the simplicity of this example allows us to focus on the critical aspects of learning and inference in relational data.

2.2 Previous Performance Comparisons

Collective inference has been a small but active area of research in relational learning for at least six years, since the publication of Chakrabarti, Dom, and Indyk's detailed study of hypertext categorization strategies [1]. Several more recent studies of collective inference have extended and broadened this work [9,12,14,15,17]. Finally, some work has extended the basic paradigm of collective inference to incorporate selecting among a range of possible actions. For example, Domingos and Richardson's work on mining the network value of customers incorporates collective inference into a larger approach to "viral marketing" [3]. Table 1 summarizes the types of models evaluated in seven key papers.

Many studies of collective inference have reported large reductions in error when the method is applied. For example, Chakrabarti et al [1] report large reductions in classification error, including one drop in error of over 70% (from 68% to 21%). In previous work, two of the authors reported significant accuracy gains from a relatively simple technique for collective inference [10]. Macskassy and Provost show how models that consider only autocorrelation in class labels (equivalent to *CI* without attributes) can perform very well when only a small fraction of the class labels are known [9].

Several studies have also pointed out that collective inference of various types can also reduce accuracy. For example, Chakrabarti et al. [1] discuss an experiment where including rela-

tional information about web pages actually reduces accuracy. They hypothesize that the additional features meant that the learning and inference scheme was "overwhelmed by the signal to noise ratio".

Table 1: Previous Work Categorization.

Paper	INTR	R1	CI	R2	RCI
Chakrabarti et al. [1]	√	√	√		√
Slattery & Mitchell [14]		√			√
Neville & Jensen [10]		√			√
Taskar et al. [16]	√	√	√		√
Taskar et al. [15]	√		√		
Macskassy & Provost [9]		√	√		√
Neville & Jensen [12]		√			√

2.3 Why Collective Models Work Well

Based on these results, it appears clear that collective inference is capable of significantly improving probabilistic inferences in relational data. Important questions remain, however: *why* and *under what circumstances* does collective inference improve the accuracy of relational models?

One reasonable explanation is that the power of collective inference lies merely in the larger feature-space provided by models such as *CI*. These models consider features that their less expressive cousins (e.g., *R1*) do not. In experiments below, we will show that this explanation is inadequate to explain the power of collective inference.

Instead, we show that methods for collective inference benefit from a *clever factoring of the space of dependencies*. The models *CI* and *RCI* have substantially smaller parameter spaces than the model *R2*, yet they can benefit from information propagated from outside of their local neighborhood. Predictions about the class label *C* on other objects essentially "bundle information" about the graph beyond the immediate neighborhood. In addition, collective models can *make use of known class labels* (e.g., known topics of web pages) to improve inferences about unknown labels. This provides a new, and often highly reliable, additional feature for learning and inference.

This increased representational power is purchased with only an incremental increase in the parameter space. In this way, *CI* and *RCI* emulate other robust techniques such as simple Bayesian classifiers and linear regression models. Even when their assumptions are violated, *CI* and *RCI* often perform well.

3. METHODS

To evaluate different models and inference methods, we conducted experiments with both real and synthetic data.

3.1 Yeast Protein Experiments

Our empirical experiments considered relational data about the yeast genome, containing information about 1,243 genes and 1,734 interactions among their associated proteins (http://www.cs.wisc.edu/~dpage/kddcup2001/). Both gene location and function are autocorrelated in this dataset [11] so we expect it to be a good testbed for investigating the relative performance of the various relational models.

The learning task was to predict gene localization in the cell. There are 15 locations ranging from mitochondria to plasma membrane, with a default error rate of 0.57. In addition to gene location, each gene has 13 boolean attributes indicating gene function. Each gene may have as many as six functions.

For non-collective models, we used relational Bayesian classifiers (RBCs) [13] to predict gene location given the function attributes. The *Intrinsic* model considered the 13 function attributes on the genes themselves. The *R1* model added another 13 function attributes for genes one link away (through interactions) for a total of 26 attributes. The *R2* model then added another 13 attributes for genes two links away, for a total of 39 attributes. For collective models, we used relational dependency networks (RDNs) [12] with RBCs to represent the component conditional probability distributions. The *CI* model considered the location attribute of genes one link away, in addition to the 13 function attributes of the genes in isolation. The *RCI* model added in the 13 function attributes for genes one link away for a total of 27 attributes. The RDNs used 250 Gibbs iterations and all models used Laplace correction for zero-values.

To compare the five approaches, we evaluated zero-one loss over ten-fold cross validation trials. We report average error over the ten folds and use two-tailed, paired t-tests to assess the significance of the results.

3.2 Synthetic Experiments

To generate synthetic data, we extended the example presented in section 2.1. We generated data with a regular two-dimensional lattice structure. The first and last two rows and columns make up the "frame" of the lattice. Objects in the frame are not used to train models, and objects in the frame are not used for loss estimates, although inference is performed over all objects in the lattice, including the frame. Thus, training or test sets of size S^2 correspond to a lattice of $(S+4) \times (S+4)$ objects, and models are trained or evaluated on the S^2 objects in the core of the lattice.

Each object in a given dataset contains the same set of attributes. In every dataset, objects contain a class label *C* and a single attribute A_1, that is correlated with *C*. Depending on dataset generation parameters, objects may also contain up to 14 additional attributes, none of which are correlated with *C*.

We generated the values of attributes and class labels in two ways, which we label "relational" and "collective". Both use parameters given in Table 2. For *collective data generation*, we begin by assigning each object in the lattice an initial class label with $P(C=1) = 0.5$. We then perform Gibbs sampling over the entire lattice. The class labels assigned to each object after 200 iterations are used as the final labels. To assign class labels during Gibbs sampling, we use a manually specified model that assigns class labels to each object based on the class values of neighboring objects one link away. The parameters of this model are varied to produce different levels of autocorrelation among neighboring class labels. Once class labels are assigned, a value for the A_1 attribute is randomly drawn from a distribution conditioned on the class label of the object — derived from the $P(C|A_1)$ and $P(A_1)$ data generation parameters. Finally, random values are assigned to all other attributes with $P(A_i) = 0.5$. Once a dataset is generated, we measure the proportion of objects with positive class labels, and any dataset with a value outside the range [0.4, 0.6] is discarded and replaced with a new dataset. This ensures consistency in $P(C)$ across datasets and reduces variance in estimated model performance.

Table 2: Data Generation Parameters.
Bold numbers in the value column indicate default values.

Name	Description	Values
TrainSize	Number of core objects in training set.	25, 49, 100, 225, 484, **1024**, 5041
TestSize	Number of core objects in test set.	**484**
NumAttrs	Number of Attributes	1,**3**,5,10,15
PropLabel	Proportion of objects with known class labels	**0.0**, 0.1, 0.3, 0.5, 0.7, 0.9
Autocorr	Autocorrelation of class labels for neighboring object *(see text)*.	0.01, 0.27, **0.49**, 0.75, 0.98
$P(A_1)$	Prior probability of $P(A_1=1)$.	0.1, **0.3**, 0.5
$P(C\|A_1)$.	Conditional probability of $P(C=1\|A_1=1)$.	0.6, 0.75, **0.9**

For *relational data generation*, we begin by training the parameters of an *R2* model on a large dataset consisting of 100 lattices of 1000 objects each. The attributes and class labels on the objects of each lattice are determined by the collective data generation method described above. We also train a univariate model of $P(A_i)$ for each attribute A_i. We create a lattice of objects in the usual way, assign attribute values randomly to each object based on the learned model of $P(A_i)$, and assign class labels to each object based on the attribute values of itself and all neighboring objects up to two links away using the learned *R2* model.

We measured bias and variance for each model using the decomposition defined for squared-loss by Domingos [2]. Loss is decomposed into three factors: bias, variance and noise. Although calculation of variance is straightforward for relational data, calculation of bias is not. Fortunately for the synthetic data experiments, we know the probabilities from the generative model and can use these as the optimal predictions. Bias and variance estimates are calculated for each test example using 10 different training sets and averaged over the entire test set. This was repeated for 20 test sets to calculate average test set bias and variance.

4. ANALYZING COLLECTIVE INFERENCE

4.1 Baseline Inference Accuracy

Figure 3a shows squared loss as a function of training set size for all models. Data for figure 3a were produced using the collective generator, so it is not unexpected that *CI* rapidly converges to a relatively low loss. *R2* continues to reduce its loss as *TrainSize* increases. At *TrainSize=5000*, none of the models achieve minimum error, corresponding to the *CI* model provided with perfect class information, but *R2* continues to reduce loss at a steady rate. For all points with *TrainSize* > *100*, the 95% confidence intervals are less than 0.004

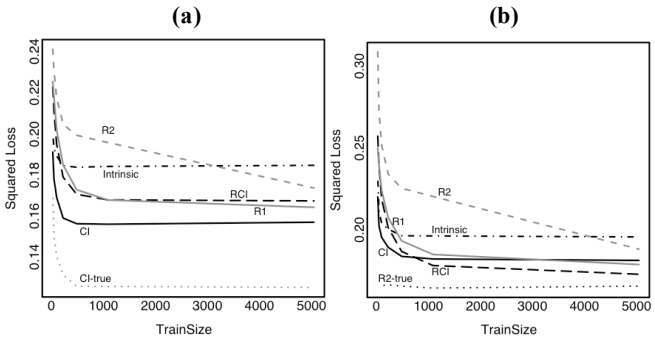

Figure 3: Squared loss for (a) collective data generation, and (b) relational data generation.

Figure 3b shows the same type of results for the relational data generator. Even though they do not match the data generator, *CI* and *RCI* outperform other models when *TrainSize* is small. *R2* continues to have corresponding high loss, though it will eventually drop below *CI* and *RCI* as it declines to the optimal value of loss at high *TrainSize*. *R1* performs similarly, dropping below *CI* at around *TrainSize*=3000. For *TrainSize* > *100*, the 95% confidence intervals are less than 0.004.

We obtained similar results with experiments on the yeast protein data, shown in Table 3. *CI* resulted in the lowest zero-one loss and its loss was significantly different than the zero-one loss of all other models.

Table 3: Yeast protein data zero-one loss results.

Model	Attributes	Zero-one loss	p-value
Intrinsic	13	0.446	0.000
R1	26	0.439	0.000
R2	39	0.455	0.000
CI	14	0.306	--
RCI	27	0.337	0.009

What is responsible for the low error of *CI* models? We measured bias and variance for the probability estimates of each model to compare their decomposed loss as a function of *TrainSize*. Figures 4a and b show the results for the relational generator. For the collective generator, the variance results were qualitatively similar and the bias varied only slightly across the range of *TrainSize*.

For all models except *R2*, bias quickly becomes nearly level. The bias of *R2* continues to decline as data slowly accumulate in joint distributions and overcome the prior (an initial Laplace correction to prevent zero probabilities). Variance is level for *Intrinsic* and *CI*, but continues to decline for *R2*, and less so for *R1* and *RCI*, between TrainSize values of 1000 and 5000. For all points with *TrainSize* > *100*, the 95% confidence intervals are less than 0.002.

Thus, the large initial gap in loss between *CI* and *R2* appears directly attributable to the size of *R2*'s parameter space, and the difficulty of making good estimates with sparse data. This difference between *CI* and *R2* is pronounced even though objects in the data contain only three attributes. If the number of attributes on each object is increased, as shown in figure 5, the loss of *R2* soars compared to other models, including *CI*, whose loss remains nearly constant. Even *R1* and *RCI* show marked increases in loss, though to a much smaller extent than *R2*. The results for data produced by collective generation are qualitatively similar.

This growth in the number of attributes is modest compared to some of the most common applications of relational learning algorithms, such as classifying web pages, in which objects have hundreds or thousands of attributes (e.g., words on a web page). For these applications, the ability of *CI* to provide a built-in factoring of the feature space may be almost essential.

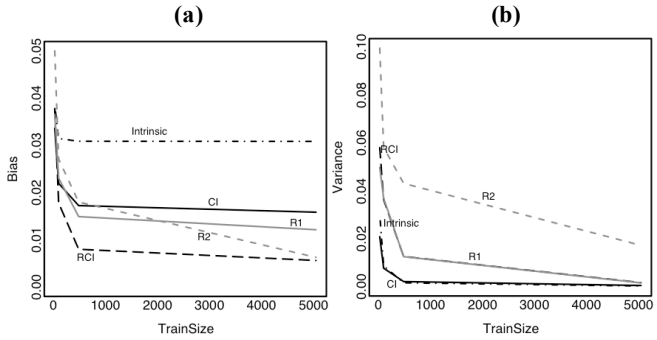

Figure 4: Loss decomposition into (a) bias, and (b) variance, for relational data generation.

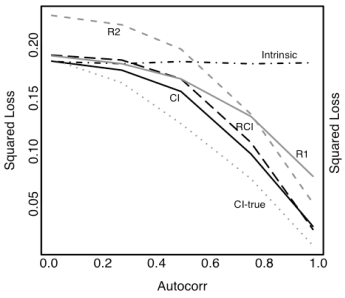

Figure 7: Loss as a function of autocorrelation.

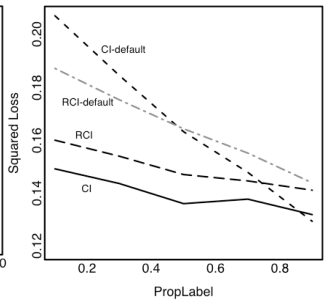

Figure 8: Loss as a function of percentage of data labeled.

The negative effect of large parameter spaces on R2 and RCI is a plausible explanation for the results of our experiments with the yeast protein data. Both *R2* and *RCI* perform worse than *CI*.

4.2 Strength of Probabilistic Dependence

Our data generation procedures use two parameters $P(A_1)$ and $P(C|A_1)$ to determine the strength of probabilistic dependence between the attribute A_1 and the class label C. The relative performance of models differs based on the strength of this dependence. Figure 6 depicts how the quantity *loss(R2)–loss(CI)* varies as a function of $P(A_1)$ and $P(C|A_1)$.

The largest difference between the two models occurs when $P(A_1)$ is uniformly distributed and the dependence of A and C is weakest. That is, *CI* performs best, in relative terms, when few correlations exist other than autocorrelation of class labels. The relative advantage of *CI* disappears as more information is available to *R2*. However, if no attributes are useful then only *CI* would be able to attain non-random performance.

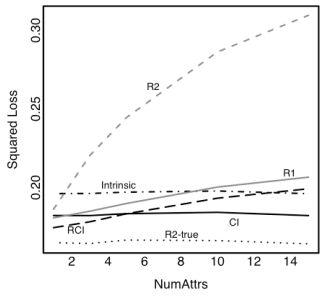

Figure 5: Loss as a function of *AttrCount*.

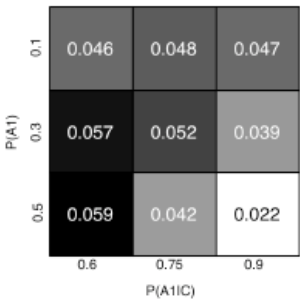

Figure 6: Relative error of *R2* over *CI* as correlation varies.

4.3 Strength of Autocorrelation

Relational autocorrelation refers to the correlation among the values of the same variable on several related objects. The widespread occurrence of autocorrelation is one of the strongest motivations for relational inference of any kind. Its effects have been noted and explored by several researchers in collective inference, including Macskassy & Provost [9], Taskar et al. [15], and Yang et al. [17].

Figure 7 shows the effect of increasing levels of autocorrelation on the relative performance of different models. All models we consider, except *Intrinsic*, are greatly aided by autocorrelation, though their relative ordering changes slightly. *R1* is aided least by autocorrelation while *R2* is aided most.

These results reveal another advantage of *CI*. Even when autocorrelation is entirely absent, *CI*'s performance is equal to that of *Intrinsic*. *CI* can exploit autocorrelation when present, but is not significantly impaired by its absence.

4.4 Proportion of Known Values

The core of collective inference is that inferences about one object can inform inferences about another. This capability is particularly useful when some values are known with certainty. For example, predictions about the topic of a previously unvisited webpage may be aided by considering the known topics of previously visited pages.

Figure 8 shows how varying the proportion of labeled data affects *CI* and *RCI*, and how their performance compares to an alternative inference scheme for these models (labeled *default*). Rather than conducting full collective inference, *default* models terminate inference after the first round of Gibbs sampling. These results indicate the advantage of collective inference over non-collective, holding all other factors constant. As shown in figure 8, the relative advantage of collective inference is reduced as more of the data are labeled. That is, collective inference procedures become less and less necessary as the percentage of true labels increases.

While only a few studies [1,9,16] have actively varied the percentage of known labels, the results above closely parallel those of Macskassy and Provost [9], who show that the relative advantage of an iterative inference procedure over a non-iterative procedure reduces as the percentage of labeled data increases. They show that, in general, their collective inference procedure performs better when class skew is present or when few labels are known with certainty.

We also evaluated this effect using the yeast protein data, obtaining the results shown in table 4. For each of the ten-fold cross-validation partitions, we learned a CI model on the 90% training partition and applied the model to the entire dataset. During collective inference, we varied the proportion of the data that was labeled—the test partition was always unlabeled but the training partition was labeled at the following levels {1.0,0.55,0.11} to produce overall levels of {0.9,0.5,0.1}. Accuracy was measured on the unlabeled instances and averaged over the ten folds. Loss increases significantly when only 10% of the data are labeled, but there is no significant difference in performance between 50% and 90% labeled.

Table 4: Yeast protein data results with partial labeling.

Model	Percent Labeled	Zero-one loss	p-value
CI	0.90	0.306	--
CI	0.50	0.296	0.474
CI	0.10	0.360	0.010

5. CONCLUSIONS

Our experiments with real and synthetic data indicate that the reduced error attributed to collective inference results primarily from a clever factoring of the space of statistical dependencies in relational data. Models that represent this factoring, when combined with algorithms for collective inference, can greatly reduce bias in data with strong autocorrelation with the minimum possible increase in variance. When autocorrelation is absent, the models have practically equivalent error to their non-relational counterparts.

6. ACKNOWLEDGMENTS

Amy McGovern and Agustin Schapira provided technical assistance with experiments. This effort is supported under a AT&T Labs Graduate Research Fellowship and by AFRL under contract number 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 AFRL or the U.S. Government.

7. REFERENCES

[1] Chakrabarti, S., B. Dom & P. Indyk. Enhanced Hypertext Classification Using Hyper-Links, In *Proc. ACM SIGMOD Conference*, pp. 307-318, 1998.

[2] Domingos, P. A Unified Bias-Variance Decomposition for Zero-One and Squared Loss. In *Proc. of the 17th National Conference on Artificial Intelligence*, pp. 564-569, 2000.

[3] Domingos, P. & M. Richardson. Mining the Network Value of Customers. In *Proc. of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 57-66, 2001.

[4] Getoor, L., N. Friedman, D. Koller, & A. Pfeffer. Learning Probabilistic Relational Models. In *Relational Data Mining*, S. Dzeroski and N. Lavrac, Eds., Springer-Verlag, 2001.

[5] Getoor, L., E. Segal, B. Taskar, & D. Koller. Probabilistic Models of Text and Link Structure for Hypertext Classification. In *Proc. IJCAI01 Workshop on Text Learning: Beyond Supervision*, 2001.

[6] Getoor, L., J. Rhee, D. Koller, & P. Small. Understanding Tuberculosis Epidemiology using Probabilistic Relational Models. *Journal of Artificial Intelligence in Medicine*, vol. 30, pp. 233-256, 2004.

[7] Jensen, D. & J. Neville. Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning. In *Proc. of the 19th International Conference on Machine Learning*, pp. 259-266, 2002.

[8] Kersting, K. & L. De Raedt. Basic principles of learning Bayesian logic programs. Technical Report No. 174, Institute for Computer Science, University of Freiburg, Germany, June 2002.

[9] Macskassy, S. & F. Provost. A Simple Relational Classifier. In *Proc. KDD-2003 Workshop on Multi-Relational Data Mining (MRDM-2003)*, pp. 64-76, 2003.

[10] Neville, J. & D. Jensen. Iterative Classification in Relational Data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pp. 13-20, 2000.

[11] Neville, J. & D. Jensen. Supporting Relational Knowledge Discovery: Lessons in Architecture and Algorithm Design. In *Proc. ICML2002 Data Mining Lessons Learned Workshop*, pp. 57-64, 2002.

[12] Neville, J., & Jensen, D. Collective Classification with Relational Dependency Networks. In *Proc. KDD-2003 Workshop on Multi-Relational Data Mining (MRDM-2003)*, pp. 77-91, 2003.

[13] Neville, J., D. Jensen & B. Gallagher. Simple Estimators for Relational Bayesian Classifiers. In *Proc. of the 3rd IEEE International Conference on Data Mining*, pp. 609-612, 2003.

[14] Slattery, S., & T. Mitchell. Discovering Test Set Regularities in Relational Domains. In *Proc. 17th International Conference on Machine Learning*, pp.895-902, 2000.

[15] Taskar, B., P. Abbeel & D. Koller. Discriminative Probabilistic Models for Relational Data. In *Proc. 18th Conference on Uncertainty in Artificial Intelligence*, pp. 485-492, 2002.

[16] Taskar, B., E. Segal & D. Koller. Probabilistic Classification and Clustering in Relational Data. In *Proc. 17th International Joint Conference on Artificial Intelligence*, pp. 870-878, 2001.

[17] Yang, Y, S. Slattery & R. Ghani. A Study of Approaches to Hypertext Categorization. *Journal of Intelligent Information Systems*. 18(2-3): 219-241. 2002.