Journal of Machine Learning Research 10 (2009) 2777-2836 Submitted 2/09; Revised 8/09; Published 12/09

Cautious Collective Classification

Luke K. McDowell LMCDOWEL@USNA.EDU
Department of Computer Science
U.S. Naval Academy
Annapolis, MD 21402, USA

Kalyan Moy Gupta KALYAN.GUPTA@KNEXUSRESEARCH.COM
Knexus Research Corporation
Springfield, VA 22153, USA

David W. Aha DAVID.AHA@NRL.NAVY.MIL
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory (Code 5514)
Washington, DC 20375, USA

Editor: Michael Collins

Abstract

Many collective classification (CC) algorithms have been shown to increase accuracy when instances are interrelated. However, CC algorithms must be carefully applied because their use of estimated labels can in some cases decrease accuracy. In this article, we show that managing this label uncertainty through *cautious* algorithmic behavior is essential to achieving maximal, robust performance. First, we describe *cautious inference* and explain how four well-known families of CC algorithms can be parameterized to use varying degrees of such caution. Second, we introduce *cautious learning* and show how it can be used to improve the performance of almost any CC algorithm, with or without cautious inference. We then evaluate cautious inference and learning for the four collective inference families, with three local classifiers and a range of both synthetic and real-world data. We find that cautious learning and cautious inference typically outperform less cautious approaches. In addition, we identify the data characteristics that predict more substantial performance differences. Our results reveal that *the degree of caution used usually has a larger impact on performance than the choice of the underlying inference algorithm.* Together, these results identify the most appropriate CC algorithms to use for particular task characteristics and explain multiple conflicting findings from prior CC research.

Keywords: collective inference, statistical relational learning, approximate probabilistic inference, networked data, cautious inference

1. Introduction

Traditional methods for supervised learning assume that the instances to be classified are independent of each other. However, in many classification tasks, instances can be related. For example, hyperlinked web pages are more likely to have the same class label than unlinked pages. Such autocorrelation (correlation of class labels among interrelated instances) exists in a wide variety of data (Neville and Jensen, 2007; Macskassy and Provost, 2007), including situations where the relationships are implicit (e.g., email messages between two people are likely to share topics).

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Cautious Collective Classification	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,77 Massachusetts Avenue,Cambridge,MA,02139**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**60**	

Collective classification (CC) is a method for jointly classifying related instances. To do so, CC methods employ a *collective inference* algorithm that exploits dependencies between instances (e.g., autocorrelation), enabling CC to often attain higher accuracies than traditional methods when instances are interrelated (Neville and Jensen, 2000; Taskar et al., 2002; Jensen et al., 2004; Sen et al., 2008). Several algorithms have been used for collective inference, including relaxation labeling (Chakrabarti et al., 1998), the iterative classification algorithm (*ICA*) (Lu and Getoor, 2003a), loopy belief propagation (*LBP*) (Taskar et al., 2002), Gibbs sampling (*Gibbs*) (Jensen et al., 2004), and variants of the weighted-vote relational neighbor algorithm (*wvRN*) (Macskassy and Provost, 2007).

During testing, all collective inference algorithms exploit relational features based on uncertain estimation of class labels. This test-time label uncertainty can diminish accuracy due to two related effects. First, an incorrectly predicted label during testing may negatively influence the predictions of its linked neighbors, possibly leading to cascading inference errors (cf., Neville and Jensen, 2008). Second, the training process may learn a poor model for test-time inference, because of the disparity between the training scenario (where labels are known and certain) and the test scenario (where labels are estimated and hence possibly incorrect). As a result, while CC has many potential advantages, in some cases CC's label uncertainty may actually cause accuracy to decrease compared to non-relational approaches (Neville and Jensen, 2007; Sen and Getoor, 2006; Sen et al., 2008).

In this article, we argue that managing this test-time label uncertainty through "cautious" algorithmic behavior is essential to achieving maximal, robust performance. We describe two complementary cautious strategies. Each addresses the fundamental problem of label uncertainty, but separately targets the two manifestations of the problem described above. First, *cautious inference* is an inference process that attends to the uncertainty of its intermediate label predictions. For example, existing algorithms such as *Gibbs* or *LBP* accomplish cautious inference by sampling from or directly reasoning with the estimated label distributions. These techniques are cautious because they prevent less certain label estimates from having substantial influence on subsequent estimations. Alternatively, we show how variants of a simpler algorithm, *ICA*, can perform cautious inference by appropriately favoring more certain information. Second, *cautious learning* refers to a training process that ameliorates the aforementioned train/test disparity. In particular, we introduce PLUL (Parameter Learning for Uncertain Labels), which uses standard cross-validation techniques, but in a way that is new for CC and that leads to significant performance advantages. In particular, PLUL is cautious because it prevents the algorithm from learning a model from the (correctly labeled) training set that overestimates how useful relational features will be when computed with uncertain labels from the test set.

We consider four frequently-studied families of CC algorithms: *ICA*, *Gibbs*, *LBP*, and *wvRN*. For each family, we describe algorithms that use varying degrees of cautious inference and explain how they all (except for the relational-only *wvRN*) can also exploit cautious learning via PLUL. We then evaluate the variants of these four families, with and without PLUL, over a wide range of synthetic and real-world data sets. To broaden the evidence for our results, we evaluate three local classifiers that are used by some of the CC algorithms, and also compare against a non-relational baseline.

While recent CC studies describe complementary results and make some related comparisons, they omit important variations that we consider here (see Section 3). Moreover, the scope and/or methodology of previous studies leaves several important questions unanswered. For instance, *Gibbs* is often regarded as one of the most accurate inference algorithms, and has been shown

to work well for CC (Jensen et al., 2004; Neville and Jensen, 2007). If so, why did Sen et al. (2008) find no significant difference between *Gibbs* and the much less sophisticated *ICA*? Second, we earlier reported that ICA_C (a cautious variant of *ICA*) outperforms both *Gibbs* and *ICA* on three real-world data sets (McDowell et al., 2007a). Why would ICA_C outperform *Gibbs*, and for what data characteristics are ICA_C's gains significant? We answer these questions and more in Section 8.

We hypothesize that *cautious CC algorithms will outperform more aggressive CC approaches when there exists a high probability of an "incorrect relational inference"*, which we define as a prediction error that is due to reasoning with relational features (i.e., an error that does not occur when relational features are removed). Two kinds of data characteristics increase the likelihood of such errors. First, when the data characteristics lead to lower overall classification accuracy (e.g., when the non-relational attributes are not highly predictive), then the computed relational feature values will be less reliable. Second, when a typical relational link is highly predictive (e.g., as occurs when the data exhibits high *relational autocorrelation*), then the potential effect of any incorrect prediction is magnified. As the magnitude of either of these data set characteristics increases, cautious algorithms should outperform more aggressive algorithms by an increasing amount.

Our contributions are as follows. First, we describe cautious inference and how four commonly-used families of existing CC inference algorithms can exhibit more or less caution. Second, we introduce cautious learning and explain how it can help compensate for the train/test disparity that occurs when a CC algorithm uses estimated class labels during testing. Third, we identify the data characteristics for which these cautious techniques should outperform more aggressive approaches, as introduced in the preceding paragraph and discussed in more detail in Section 6. Our experimental results confirm that cautious approaches typically do outperform less cautious variants, and that these effects grow larger when there is a greater probability of incorrect relational inference. Moreover, our results reveal that in most cases *the degree of caution used has a larger impact on performance than the choice of the underlying inference algorithm*. In particular, the cautious algorithms perform very similarly, regardless of whether ICA_C or *Gibbs* or *LBP* is used, although our results also confirm that, for some data characteristics, inference with *LBP* performs comparatively poorly. These results suggest that in many cases the higher computational complexity of *Gibbs* and *LBP* is unnecessary, and that the much faster ICA_C should be used instead. Finally, our results and analysis enable us to answer the previously mentioned questions regarding CC.

The next two sections summarize collective classification and related work. Section 4 then explains why CC needs to be cautious and describes cautious inference and learning in more detail. In Section 5, we describe how caution can be specifically used by the four families of CC inference algorithms. Section 6 then describes our methodology and hypotheses. Section 7 presents our results, which we discuss in Section 8. We conclude in Section 9.

2. Collective Classification: Description and Problem Definition

In this section, we first motivate and define collective classification (CC). We then describe different approaches to CC, different CC tasks, and our assumptions for this article.

2.1 Problem Statement and Example

In many domains, relations exist among instances (e.g., among hyperlinked web pages, social network members, co-cited publications). These relations may be helpful for classification tasks, such

as predicting the topic of a publication or the group membership of a person (Koller et al., 2007). More formally, we consider the following task (based on Macskassy and Provost, 2006):

Definition 1 *(Classification of Graph-based Data) Assume we are given a graph $G = (V, E, X, Y, C)$ where V is a set of nodes, E is set of (possibly directed) edges, each $\vec{x}_i \in X$ is an attribute vector for node $v_i \in V$, each $Y_i \in Y$ is a label variable for v_i, and C is the set of possible labels. Assume further that we are given a set of "known" values Y^K for nodes $V^K \subset V$, so that $Y^K = \{y_i | v_i \in V^K\}$. Then the task is to infer Y^U, the values of Y_i for the remaining nodes with "unknown" values ($V^U = V - V^K$), or a probability distribution over those values.*[1]

For example, consider the task of predicting whether a web page belongs to a professor or a student. Conventional supervised learning approaches ignore the link relations and classify each page using attributes derived from its content (e.g., words present in the page). We refer to this approach as *non-relational classification*. In contrast, a technique for *relational classification* would explicitly use the links to construct additional relational features for classification (e.g., for each page, including as features the words from hyperlinked pages). This additional information can potentially increase classification accuracy, though may sometimes decrease accuracy as well (Chakrabarti et al., 1998). Alternatively, even greater (and usually more reliable) increases can occur when the class *labels* of the linked pages are used instead to derive relevant relational features (Jensen et al., 2004). However, using features based on these labels is challenging, because some or all of the labels are initially unknown, and thus typically must be estimated and then iteratively refined in some way. This process of jointly inferring the labels of interrelated nodes is known as *collective classification* (CC).

Figure 1 summarizes an example execution of a simple CC algorithm, *ICA*, applied to the binary web page classification task. Each step in the sequence displays a graph of four nodes, where each node denotes a web page, and hyperlinks among them. Each node has a class label y_i; the set of possible class labels is $C = \{P, S\}$, denoting *professors* and *students*, respectively. Three nodes have *unknown* labels ($V^U = \{v_1, v_2, v_4\}$) and one node has a *known* label ($V^K = \{v_3\}$). In the initial state (step A), no label y_i has yet been estimated for the nodes in V^U, so each is set to *missing* (indicated by a question mark). Each node has three binary attributes (represented by $\vec{x}_i$). Nodes in V^U also have two relational features (one per class), represented by the vector $\vec{f}_i$. Each feature denotes the number of linked nodes (ignoring link direction) that have a particular class label.

In step B, some classifier (not shown) estimates class labels for nodes in V^U using only the (non-relational) attributes. These labels, along with the known label y_3, are used in step C to compute the relational feature value vectors. For instance, in step C, $\vec{f}_2 = (1\ 2)$ because v_2 links to nodes with one current *P* label and two current *S* labels. In step D, a classifier re-estimates the labels using both attributes and relational features, which changes the predicted label of v_2. In step E, relational feature values are re-computed using the new labels. Steps D and E then repeat until a termination criterion is satisfied (e.g., convergence, number of iterations).

This example exhibits how relational value uncertainty occurs with CC. For instance, the feature vector $\vec{f}_1$ is $(1\ 0)$ in step C but later becomes $(0\ 1)$. Thus, intermediate predictions use uncertain label estimates, motivating the need to cautiously use such estimates.

1. V^K may be empty. In addition, a separate training graph may be provided; see Section 2.3.

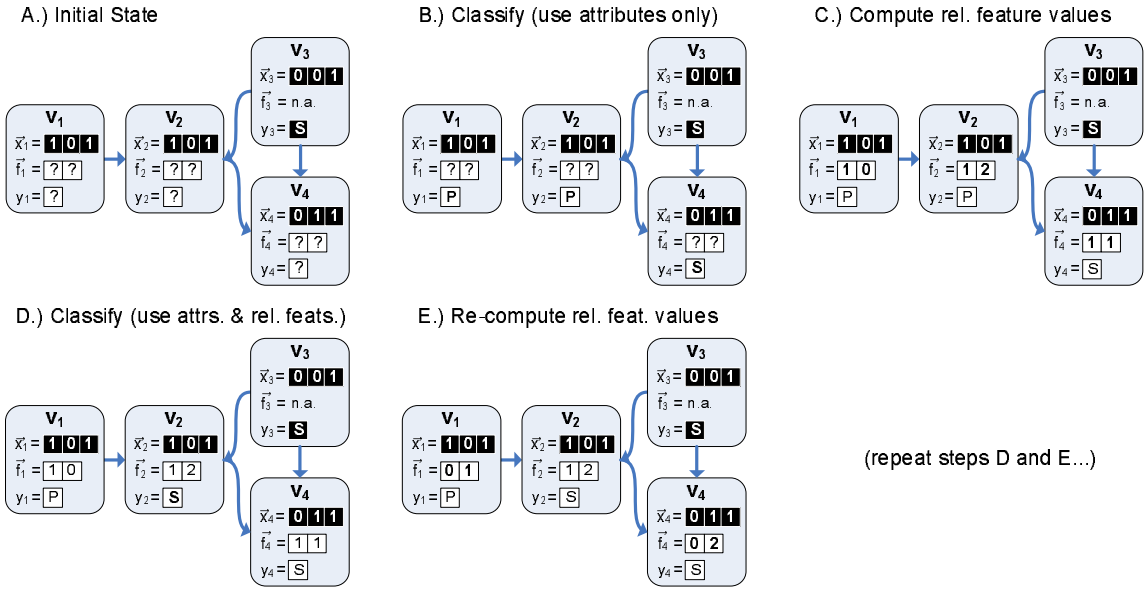

Figure 1: Example operation of *ICA*, a simple CC algorithm. Each step (A thru E) shows a graph of 4 linked nodes (i.e., web pages). "Known" values are are shown in white text on a black background; this includes all attribute values $\vec{x}_i$ and the class label y_3 for v_3. Estimated values are shown instead with a white background.

2.2 Algorithms for Collective Inference

For some collective inference tasks, exact methods such as junction trees (Huang and Darwiche, 1996) or variable elimination (Zhang and Poole, 1996) can be applied. However, these methods may be prohibitively expensive to use (e.g., summing over the remaining variable configurations is intractable for modest-sized graphs). Some research has focused on methods that further factorize the variables, and then apply an exact procedure such as belief propagation (Neville and Jensen, 2005), min-cut partition (Barzilay and Lapata, 2005), or methods for solving quadratic and linear programs (Triebel et al., 2007). In this article, we consider only approximate collective inference methods.

We consider three primary types of approximate collective inference algorithms, borrowing some terminology from Sen et al. (2008):

- **Local classifier-based methods.** For these methods, inference is an iterative process whereby a *local classifier* predicts labels for each node in V^U using both attributes and relational features (derived from the current label predictions), and then a *collective inference* algorithm recomputes the class labels, which will be used in the next iteration. Examples of this type of CC algorithm include *ICA* (used in the example above) and *Gibbs*. Local classifiers that have been used include Naive Bayes (Jensen et al., 2004), relational probability trees (Neville et al., 2003a), k-nearest neighbor (McDowell et al., 2007b), and logistic regression (Sen et al., 2008). Typically, a supervised learner induces the local classifier from the training set using both attributes and relational features.

- **Global formulation-based methods.** These methods train a classifier that seeks to optimize one global objective function, often based on a Markov random field (Dobrushin, 1968; Besag, 1974). As above, the classifier uses both attributes and relational features for inference. Examples of these algorithms include loopy belief propagation and relaxation labeling. These do not use a separate local classifier; instead, the entire algorithm is used for both training (e.g., to learn the clique potentials) and inference. See Taskar et al. (2002) and Sen et al. (2008) for more details.

- **Relational-only methods.** Recently, Macskassy and Provost (2007) demonstrated that, when some labels are known (i.e., $|V^K| > 0$), algorithms that use *only* relational information can in some cases perform very well. We consider several variants of the algorithm they described, $wvRN_{RL}$ (weighted-vote relational neighbor, with relaxation labeling). This algorithm computes a new label distribution for a node by averaging the current distributions of its neighbors. It does not require any training.

With local classifier methods, learning the classifier can often be done in a single pass over the data, does not require running collective inference, and in fact is independent of the collective inference procedure that will be used. In contrast, for global methods the local classifier and inference algorithm are effectively unified. As a result, learning for a global method requires committing to and actually executing a specific inference algorithm, and thus can be much slower than with a local classifier-based method.

All of these algorithms jointly classify interrelated nodes using some iterative process. Those that propagate from one iteration to the next a single label for each node are called *hard-labeling* methods. Methods that instead propagate a probability distribution over the possible class labels are called *soft-labeling* methods (cf., Galstyan and Cohen, 2007). All of the local classifier-based methods that we examine are hard-labeling methods.[2] Soft-labeling methods, such as variants of relaxation labeling, are also possible but require that the local classifier be able to reason directly with label distributions, which is more complex than the label aggregation for features typically done with approaches like *ICA* or *Gibbs*. Section 6.6 provides more detail on these features.

2.3 Task Definitions and Focus

Collective classification has been applied to two types of inference tasks, namely the **out-of-sample** task, where V^K is empty, and the **in-sample** task, where V^K is not empty. Both types of tasks may emerge in real-world situations (Neville and Jensen, 2005). Prior work on out-of-sample tasks (Neville and Jensen, 2000; Taskar et al., 2002; Sen and Getoor, 2006) assume that the algorithm is also provided with a training graph G_{Tr} that is disjoint from the test graph G. For instance, a model may be learned over the web-graph for one institution, and tested on the web-graph of another.

For in-sample tasks, where some labels in G are known, CC can be applied to the single graph G (Macskassy and Provost, 2007; McDowell et al., 2007a; Sen et al., 2008; Gallagher et al., 2008); *within-network* classification (Macskassy and Provost, 2006) involves training on the subset $G^K \subset G$ with known labels, and testing by running inference over the entire graph. This task simulates, for example, fraud detection in a single large telecommunication network where some entities/nodes are

2. We could also consider $wvRN_{RL}$, which is a soft-labeling method, to be a local classifier-based method, albeit a simple one that ignores attributes and does no learning. However, for explication we list relational-only methods as a separate category in the list above because our results will show they often have rather different performance trends.

known to be fraudulent. Another in-sample task (Neville and Jensen, 2007; Bilgic and Getoor, 2008; Neville and Jensen, 2008) assumes a separate training graph G_{Tr}, where a model is learned from G_{Tr} and inference is performed over the test graph G, which includes both labeled and unlabeled nodes. For both tasks, predictive accuracy is measured only for the unlabeled nodes.

In Section 6, we will address three types of tasks (i.e., out-of-sample, sparse in-sample, and dense in-sample). This is similar to the set of tasks addressed in some previous evaluations (e.g., Neville and Jensen, 2007, 2008; Bilgic and Getoor, 2008) and subsumes some others (e.g., Neville and Jensen, 2000; Taskar et al., 2002; Sen and Getoor, 2006). We will not directly address the within-network task, but the algorithmic trends observed from our in-sample evaluations should be similar.[3]

2.4 Assumptions and Limitations

In this broad investigation on the utility of caution in collective classification, we make several simplifying assumptions. First, we assume data is obtained passively rather than actively (Rattigan et al., 2007; Bilgic and Getoor, 2008). Second, we assume that nodes are homogeneous (e.g., all represent the same kind of object) rather than heterogeneous (Neville et al., 2003a; Neville and Jensen, 2007). Third, we assume that links are not missing, and need not be inferred (Bilgic and Getoor, 2008). Finally, we do not attempt to increase autocorrelation via techniques such as link addition (Gallagher et al., 2008), clustering (Neville and Jensen, 2005), or problem transformation (Tian et al., 2006; Triebel et al., 2007).

Our example in Figure 1 employs a simple relational feature (i.e., that counts the number of linked nodes with a specific class label). However, several other types of relations exist. For example, Gallagher and Eliassi-Rad (2008) describe a topology of feature types, including structural features that are independent of node labels (e.g., the number of linked neighbors of a given node). We focus on only three simple types of relational features (see Section 6.6), and leave broader investigations for future work. Likewise, for CC algorithms that learn, we assume that training is performed just once, which differs from some prior work where the learned model is updated in each iteration (Lu and Getoor, 2003b; Gurel and Kersting, 2005).

3. Related Work

Besag (1986) originally described the "Iterated Conditional Modes" (ICM) algorithm, which is a version of the *ICA* algorithm that we consider. Several researchers have reported that employing inter-instance relations in CC algorithms can significantly increase predictive accuracy (e.g., Chakrabarti et al., 1998; Neville and Jensen, 2000; Taskar et al., 2002; Lu and Getoor, 2003a). Furthermore, these algorithms have performed well on a variety of tasks, such as identifying securities fraud (Neville et al., 2005), ranking suspicious entities (Macskassy and Provost, 2005), and annotating semantic web services (Heß and Kushmerick, 2004).

In each iteration, a CC algorithm predicts a class label (or a class distribution) for each node and uses it to determine the next iteration's predictions. Although using label predictions from linked nodes (instead of using the larger number of attributes from linked nodes) encapsulates the influence of a linked node and simplifies learning (Jensen et al., 2004), it can be problematic. For example,

3. Indeed, we performed additional experiments where we reproduced the synthetic data of Sen et al. (2008), but then transformed the task from their within-network variant to a variant that uses a separate graph for training (as done in this article), and obtained results similar to those they reported.

iterating with incorrectly predicted labels can propagate and amplify errors (Neville and Jensen, 2007; Sen and Getoor, 2006; Sen et al., 2008), diminishing or even reducing accuracy compared to non-relational approaches. In this article, we examine the data characteristics (and algorithmic interactions) for which these issues are most serious and explain how cautious approaches can ameliorate them.

The performance of CC compared to non-relational learners depends greatly on the data characteristics. First, for CC to improve performance, the data must exhibit *relational autocorrelation* (Jensen et al., 2004; Neville and Jensen, 2005; Macskassy and Provost, 2007; Rattigan et al., 2007; Sen et al., 2008), which is correlation among the labels of related instances (Jensen and Neville, 2002). Complex correlations can be exploited by some CC algorithms, capturing for instance the notion "Professors primarily have out-links to Students." In contrast, the simplest kind of correlation is *homophily* (McPherson et al., 2001), in which links tend to connect nodes with the same label. To facilitate replication, Appendix A defines homophily more formally.

A second data characteristic that can influence CC performance is *attribute predictiveness*. For example, if the attributes are far less predictive than the selected relational features, then CC algorithms should perform comparatively well vs. traditional algorithms (Jensen et al., 2004). Third, *link density* plays a role (Jensen and Neville, 2002; Neville and Jensen, 2005; Sen et al., 2008); if there are few relations among the instances, then collective classification may offer little benefit. Alternatively, algorithms such as *LBP* are known to perform poorly when link density is very high (Sen and Getoor, 2006). Fourth, an important factor is the *labeled proportion* (the proportion of test nodes that have known labels). In particular, if some node labels are known ($|V^K| > 0$), these labels may help prevent CC estimation errors from cascading. In addition, if a substantial number of labels are known, simpler relational-only algorithms may be the most effective. Although additional data characteristics exist that can influence the performance of CC algorithms, such as *degree of disparity* (Jensen et al., 2003) and *assortativity* (Newman, 2003; Macskassy, 2007), we concentrate on these four in our later evaluations.

Compared to this article, prior studies provide complementary results and make some relevant comparisons, but do not examine important variations that we consider here. For instance, Jensen et al. (2004) only investigate a single collective inference algorithm, and Macskassy and Provost (2007) focus on relational-only (univariate) algorithms. Sen et al. (2008) assess several algorithms on real and synthetic data, but do not examine the impact of attribute predictiveness or labeled proportion. Likewise, Neville and Jensen (2007) evaluate synthetic and real data, but vary data characteristics (autocorrelation and labeled proportion) for only the synthetic data, do not consider *ICA*, and consider *LBP* only for the synthetic data. In addition, only one of these prior studies (Neville and Jensen, 2000) evaluates an algorithm related to ICA_C, which is a simple cautious variant of *ICA* that we show has promising performance. Moreover, these studies did not compare algorithms that vary only in their degree of cautious inference, or use cautious learning.

4. Types of Caution in CC and Why Caution is Important

Section 3 described how collective classification exploits label predictions to try to increase accuracy, but how iterating with incorrectly predicted labels can sometimes propagate and amplify errors. To address this problem, we recently proposed the use of cautious inference for CC (McDowell et al., 2007a). We defined an inference algorithm to be cautious if it sought to "explicitly identify and preferentially exploit the more certain relational information." In addition, we ex-

plained that a variant of *ICA* that we here call ICA_C is cautious because it selectively ignores class labels that were predicted with less confidence by the local classifier. Previously, Neville and Jensen (2000) introduced a simpler version[4] of ICA_C but compared it only with non-relational classifiers. We showed that ICA_C can outperform *ICA* and *Gibbs*, but did not identify the data conditions under which such gains hold.

In this article, we expand our original notion of caution in two ways. First, we broaden our idea of *cautious inference* to encompass several other existing CC inference techniques that seek the same goal (managing prediction uncertainty). Recognizing the behavioral similarities between these different algorithms helps us to better assess the strengths and weaknesses of each algorithm for a particular data set. Second, we introduce *cautious learning*, a technique that ameliorates prediction uncertainty even before inference is applied, which can substantially increase accuracy. Below we detail these two types of caution.

- A CC algorithm exhibits **cautious inference** if its inference process attends to the uncertainty of its intermediate label predictions. Usually, this uncertainty is approximated via the posterior probabilities associated with each predicted label. For instance, a CC algorithm may exercise cautious inference by favoring predicted information that has less uncertainty (higher confidence). This is the approach taken by ICA_C, which uses only the most certain labels at the beginning of its operation, then gradually incorporates less certain predictions in later iterations. Alternatively, instead of always selecting the most likely class label for each node (like *ICA* and ICA_C), *Gibbs* re-samples the label of each node based on its estimated distribution. This re-sampling leads to more stochastic variability (and less influence) for nodes with less certain predictions. Finally, soft-labeling algorithms like *LBP*, relaxation labeling, and $wvRN_{RL}$ directly reason with the estimated label distributions. For instance, $wvRN_{RL}$ averages the estimated distributions of a node's linked neighbors, which gives more influence to more certain predictions.

- A CC algorithm exhibits **cautious learning** if its training process is influenced by recognizing the disparity between the training set (where labels are known and certain) and the test scenario (where labels may be estimated and hence incorrect). In particular, a relational feature may appear to be highly predictive of the class when examining the training set (e.g., to learn conditional probabilities or feature weights), yet its use may actually decrease accuracy if its value is often incorrect during testing. In response, one approach is to ensure that appropriate training parameters are cross-validated using the actual testing conditions (e.g., with estimated test labels). We use PLUL to achieve this goal.

The next section describes how these general ideas can be applied. Later, our experimental results demonstrate when they lead to significant performance improvements.

5. Applying Cautious Inference and Learning to Collective Classification

The previous section described two types of caution for CC. Each attempts to alleviate potential estimation errors in labels during collective inference. Cautious inference and cautious learning can often be combined, and at least one is used or is applicable to every CC algorithm known to

4. Their algorithm is like ICA_C, except that it does not consider how to favor "known" labels from V^K.

us. In this section, we provide examples of how both types can be applied by describing specific CC algorithms that exploit cautious inference (Sections 5.1-5.4), and by describing how PLUL can complement these algorithms with cautious learning (Section 5.5). Section 5.6 then discusses the computational complexity of these algorithms.

We describe and evaluate four families of CC inference algorithms: *ICA*, *Gibbs*, *LBP*, and *wvRN*.[5] Among local classifier-based algorithms, we chose *ICA* and *Gibbs* because both have been frequently studied and often perform well. As a representative global formulation-based algorithm, we chose *LBP* instead of relaxation labeling because previous studies (Sen and Getoor, 2007; Sen et al., 2008) found similar performance, with in some cases a slight edge for *LBP*. Finally, we select *wvRN* because it is a good relational-only baseline for CC evaluations (Macskassy and Provost, 2007).

Table 1 summarizes the four CC families that we consider. Within each family, each variant use more cautious inference than the variant listed below it. Cautious variants of standard algorithms are given a "C" subscript (e.g., ICA_C), while non-cautious variants of standard algorithms are given a "NC" subscript (e.g., $Gibbs_{NC}$). For the latter case, our intent is not to demonstrate large performance "gains" for a standard algorithm vs. a new non-cautious variant, but to isolate the impact of a particular cautious algorithmic behavior on performance. While the result may not be a theoretically coherent algorithm (e.g., $Gibbs_{NC}$, unlike *Gibbs*, is not a MCMC algorithm), in every case the resultant algorithm *does* perform well under data set situations where caution is not critical (see Section 7). Thus, comparing the performance of the cautious vs. non-cautious variants allows us to investigate the data characteristics for which cautious behavior is more important.[6]

5.1 ICA Family of Algorithms

Figure 2 displays pseudocode for *ICA*, ICA_C, and ICA_{Kn}, depending on the setting of the parameter *AlgType*. We describe each in turn.

5.1.1 *ICA*

In Figure 2, step 1 is a "bootstrap" step that predicts the class label y_i of each node in V^U using only attributes ($conf_i$ records the confidence of this prediction, but *ICA* ignores this information). The algorithm then iterates (step 2). During each iteration, *ICA* selects all available labels (step 3), computes the relational features' values based on these labels (step 4), and then re-predicts the class label of each node using both attributes and relational features (step 5). After iterating, hopefully to convergence, step 6 returns the final set of estimated class labels.

Types of Caution Used: Steps 3-4 of *ICA* use all available labels for feature computation (including estimated, possibly incorrect labels) and step 5 picks the single most likely label for each node based on the new predictions. In these steps, uncertainty in the predictions is ignored. Thus, *ICA* does not

5. Technically, *wvRN* by itself is a local classifier, not an inference algorithm, but for brevity we refer to the family of algorithms based on this classifier (such as $wvRN_{RL}$) as *wvRN*.
6. Section 7 shows that the non-cautious variants *ICA*, $Gibbs_{NC}$, and LBP_{NC} perform similarly to each other. Thus, our empirical results would change little if we compared all of the cautious algorithms against the more standard *ICA*. However, the results for *Gibbs* and *LBP* would then concern performance differences between distinct algorithms, due to conjectured but unconfirmed differences in algorithmic properties. By instead comparing *Gibbs* vs. $Gibbs_{NC}$ and *LBP* vs. LBP_{NC}, we more precisely demonstrate that the cautious algorithms benefit from specifically identified cautious behaviors.

Name	Cautious Inf.?	Key Features	Type	Evaluated by?
Local classifier-based methods that iteratively classify nodes, yielding a final graph state				
ICA_C	Favors more conf. labels	Relational features depend only on "more confident" estimated labels; later iterations loosen confidence threshold.	Hard	Neville and Jensen (2000); McDowell et al. (2007a)
ICA_{Kn}	Favors known labels	First iteration: rel. features depend only on known labels. Later iterations: use all labels.	Hard	McDowell et al. (2007a)
ICA	Not cautious	Always use all labels, known and estimated.	Hard	Lu and Getoor (2003a); Sen and Getoor (2006); McDowell et al. (2007a,b)
Local classifier-based algorithms that compute conditional probabilities for each node				
Gibbs	Samples from estimated distribution	At each step, classifies using *all* neighbor labels, then samples new labels from the resultant distributions. Records new labels to produce final marginal statistics.	Hard	Jensen et al. (2004); Neville and Jensen (2007); Sen et al. (2008)
$Gibbs_{NC}$	Not cautious	Like *Gibbs*, but always pick most likely label instead of sampling.	Hard	None, but very similar to ICA.
Global formulation algorithms based on loopy belief propagation (LBP)				
LBP	Reasons with estimated distribution	Passes continuous-valued messages between linked neighbors until convergence.	Soft	Taskar et al. (2002); Neville and Jensen (2007); Sen et al. (2008)
LBP_{NC}	Not cautious	Like *LBP*, but each node always chooses single most likely label to use for next round of messages.	Hard	—
Relational-only algorithms				
$wvRN_{RL}$	Reasons with estimated distribution	Computes new distribution by averaging neighbors' label distributions; combines old and new distributions via relaxation labeling.	Soft	Macskassy and Provost (2007); Gallagher et al. (2008)
$wvRN_{ICA+C}$	Favors nodes closer to known labels	Initializes nodes in V^U to *missing*. Computes most likely label by averaging neighbors' labels, ignoring *missing* labels.	Hard	Macskassy and Provost (2007); similar to Galstyan and Cohen (2007)
$wvRN_{ICA+NC}$	Not cautious	Like $wvRN_{ICA+C}$, but no *missing* labels are used. Instead, initialize nodes in V^U by sampling from the prior label distribution.	Hard	—

Table 1: The ten collective inference algorithms considered in this article, divided into four families. Hard/soft refers to hard-labeling and soft-labeling (see Section 2.2).

perform cautious inference. However, it may exploit cautious learning to learn the classifier models that are used for inference (M_A and M_{AR}).

5.1.2 ICA_C

In steps 3-4 of Figure 2, *ICA* assumes that the estimated node labels are all equally likely to be correct. When *AlgType* instead selects ICA_C, the inference becomes more cautious by only considering more certain estimated labels. Specifically, step 3 "commits" into Y' only the best m of

ICA_classify $(V, E, X, Y^K, M_{AR}, M_A, n, AlgType)$=
// V=nodes, E=edges, X=attribute vectors, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// M_{AR}=local classifier (uses attributes and relations), M_A=classifier that uses only attributes
// n=# of iterations, $AlgType$=ICA_C, ICA_{Kn}, or ICA

1 **for each** node $v_i \in V^U$ **do** // Bootstrap: estimate label y_i for each node
  $(y_i, conf_i) \leftarrow M_A(\vec{x}_i)$ // using attributes only
2 **for** h = 0 **to** n **do**
3 // Select node labels to use for computing relational feature values, store in Y'
  if ($AlgType = ICA_C$) // For ICA_C: use known and m most confident
    $m \leftarrow |V^U| \cdot (h/n)$ // estimated labels, gradually increase m
    $Y' \leftarrow Y^K \cup \{y_i | v_i \in V^U \wedge rank(conf_i) \leq m\}$
  else if (h = 0) and ($AlgType = ICA_{Kn}$)
    $Y' \leftarrow Y^K$ // For ICA_{Kn}(first iteration): use *only* known labels
  else // For ICA_{Kn} (after first iteration) and ICA: use *all*
    $Y' \leftarrow Y^K \cup \{y_i | v_i \in V^U\}$ // labels (known and estimated)
4 **for each** node $v_i \in V^U$ **do**
    $\vec{f}_i \leftarrow calcRelatFeats(V, E, Y')$ // Compute feature values, using labels selected above
5 **for each** node $v_i \in V^U$ **do** // Re-predict most likely label, using attributes
    $(y_i, conf_i) \leftarrow M_{AR}(\vec{x}_i, \vec{f}_i)$ // and relational features
6 **return** $\{y_i | v_i \in V^U\}$ // Return predicted class label for each node

Figure 2: Algorithm for *ICA* family of algorithms. We use $n = 10$ iterations.

the current estimated labels; other labels are considered *missing* and thus ignored in the next step. Step 4 computes the relational features using only the committed labels, and step 5 classifies using this information. Step 3 gradually increases the fraction of estimated labels that are committed per iteration (e.g., if n=10, from 0%, 10%, 20%,..., up to 100%). Node label assignments committed in an iteration h are not necessarily committed again in iteration $h+1$ (and may in fact change).

ICA_C requires some kind of confidence measure ($conf_i$ in Figure 2) to determine the "best" m of the current label assignments (those with the highest confidence "rank"). We adopt the approach of Neville and Jensen (2000) and use the posterior probability of the most likely class for each node i as $conf_i$. In exploratory experiments, we found that alternative measures (e.g., probability difference of the top two classes) produced similar results.

Types of Caution Used: ICA_C favors more confident information by ignoring nodes whose labels are estimated with lower confidence. Step 3 executes this preference, which affects the algorithm in several ways. First, omitting the estimated labels for some nodes causes the relational feature value computation in step 4 to ignore those less certain labels. Since this computation favors more reliable label assignments, subsequent assignments should also be more reliable. Second, if any node links only to nodes with *missing* labels, then the computed value of the relational features for that node will also be *missing*; Section 6.5 describes how the classifier in Step 5 handles this case. Third, recall that a realistic CC scenario's test set may have links to nodes with known labels; these nodes, represented by V^K, provide the "most certain" labels and thus may aid classification. ICA_C exploits *only* these labels for iteration $h = 0$. In this case, step 3 ignores all estimated labels (every estimate for V^U), but step 4 can still compute some relational feature values based on known labels

Gibbs_classify $(V, E, X, Y^K, M_{AR}, M_A, n, n_B, C, AlgType)$=
// V=nodes, E=edges, X=attribute vectors, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// M_{AR}=local classifier (uses attributes and relations), M_A=classifier that uses only attributes
// n=# of iterations, n_B= "burn-in" iters., C=set of class labels, $AlgType$=*Gibbs* or $Gibbs_{NC}$

1 **for each** node $v_i \in V^U$ **do** // Bootstrap: estimate label probs. $\vec{b}_i$
  $\vec{b}_i \leftarrow M_A(\vec{x}_i)$ // for each node, using attributes only
2 **for each** node $v_i \in V^U$ **do** // Initialize statistics
  for each $c \in C$
    $stats[i][c] \leftarrow 0$
3 **for** $h = 1$ **to** n **do** // Repeat for n iterations
  for each node $v_i \in V^U$ **do**
4 **switch** ($AlgType$):
    case ($Gibbs$): $y_i \leftarrow sampleDist(\vec{b}_i)$ // Sample next label from distribution
    case ($Gibbs_{NC}$): $y_i \leftarrow argmax_{c \in C}\ b_i(c)$ // Or, pick most likely label from dist.
5 **if** $(h > n_B)$ $stats[i, y_i] \leftarrow stats[i, y_i] + 1$ // Record stats. on chosen label
6 $Y' \leftarrow Y^K \cup \{y_i | v_i \in V^U\}$ // Compute feature values, using known
  for each node $v_i \in V^U$ **do** // labels and labels chosen in step 4
    $\vec{f}_i \leftarrow computeRelatFeatures(V, E, Y')$
7 **for each** node $v_i \in V^U$ **do** // Re-estimate label probs., using
    $\vec{b}_i \leftarrow M_{AR}(\vec{x}_i, \vec{f}_i)$ // attributes and relational features
8 **return** *stats* // Return marginal stats. for each node

Figure 3: Algorithm for Gibbs sampling. Thousands of iterations are typically needed.

from V^K. Thus, the known labels influence the first classification in step 5, before any estimated labels are used, and in subsequent iterations. Finally, ICA_C can also benefit from PLUL.

5.1.3 ICA_{Kn}

The above discussion highlighted two different effects from ICA_C: favoring more confident estimated labels vs. favoring known labels from V^K. An interesting variant is to favor the known labels in the first iteration (just like ICA_C), but then use all labels for subsequent iterations (just like *ICA*). We call this algorithm ICA_{Kn} ("ICA+Known").

Types of Caution Used: ICA_{Kn} favors only known nodes. It is thus more cautious than *ICA*, but less cautious than ICA_C. It can also benefit from cautious learning via PLUL.

5.2 Gibbs Family of Algorithms

Figure 3 displays pseudocode for Gibbs sampling (*Gibbs*) and the non-cautious variant $Gibbs_{NC}$. We describe each in turn.

5.2.1 *Gibbs*

In Figure 3, step 1 (bootstrapping) is identical to step 1 of the *ICA* algorithms, except that for each node v_i the classifier must output a distribution $\vec{x}_i$ containing the likelihood of each class, not just

the most likely class. Step 2 initializes the statistics that will be used to compute the marginal class probabilities for each node. In step 4, within the loop, the algorithm probabilistically samples the current class label distribution of each node and assigns a single label y_i based on this distribution. This label is also recorded in the statistics during Step 5 (after the first n_B iterations are ignored for "burn-in"). Step 6 then selects all labels (known labels and those just sampled) and uses them to compute the relational features' values. Step 7 re-estimates the posterior class label probabilities given these relational features. The process then repeats. When the process terminates, the statistics recorded in step 5 approximate the marginal distribution of class labels, and are returned by step 8.
Types of Caution Used: Like ICA_C, *Gibbs* is cautious in its use of estimated labels, but in a different way. In particular, ICA_C exercises caution in step 3 by ignoring (at least for some iterations) labels that have lower confidence. In contrast, *Gibbs* exercises caution by sampling, in step 4, values from each node's predicted label distribution—causing nodes with lower prediction confidence to reflect that uncertainty via higher fluctuation in their assigned labels, yielding less predictive influence on their neighbors. Gibbs can also benefit from cautious learning via PLUL.

We expect *Gibbs* to perform better than ICA_C, since it makes use of more information, but this requires careful confirmation. In addition, *Gibbs* is considerably more time intensive than ICA_C or *ICA* (see Section 5.6).

5.2.2 $Gibbs_{NC}$

$Gibbs_{NC}$ is identical to *Gibbs* except that instead of sampling in step 4, it always selects the most likely label. This change makes $Gibbs_{NC}$ deterministic (unlike *Gibbs*), and makes $Gibbs_{NC}$ behave almost identically to *ICA*. In particular, observe that after any number of iterations h ($1 \leq h \leq n$), *ICA* and $Gibbs_{NC}$ will have precisely the same set of current label assignments for every node. However, *ICA*'s result is the final set of label assignments, whereas $Gibbs_{NC}$'s result is the marginal statistics computed from these time-varying assignments. For a given data set, if *ICA* converges to an an unchanging set of label assignments, then for sufficiently large n $Gibbs_{NC}$'s final result (in terms of accuracy) will be identical to *ICA*'s. If, however, some nodes' labels continue to oscillate with *ICA*, then *ICA* and $Gibbs_{NC}$ will have different results for some of those nodes.
Types of Caution Used: Just like *ICA*, $Gibbs_{NC}$ uses all available labels for relational feature computation, and always picks the single most likely label based on the new predictions. Thus, $Gibbs_{NC}$ does not perform cautious inference, though it can benefit from cautious learning to learn the classifiers M_A and M_{AR}.

5.3 LBP Family of Algorithms

This section describes loopy belief propagation (*LBP*) and the non-cautious variant LBP_{NC}.

5.3.1 *LBP*

LBP has been a frequently studied technique for performing approximate inference, and has been used both in early work on CC (Taskar et al., 2002) and in more recent evaluations (Sen and Getoor, 2006; Neville and Jensen, 2007; Sen et al., 2008). Most works that study *LBP* for CC treat the entire graph, including attributes, as a pairwise Markov random field (e.g., Sen and Getoor, 2006; Sen et al., 2008) and then justify *LBP* as an example of a variational method (cf., Yedidia et al., 2000). The basic inference algorithm is derived from belief propagation (Pearl, 1988), but applied to graphs with cycles (McEliece et al., 1998; Murphy et al., 1999).

LBP performs inference via passing messages from node to node. In particular, $m_{i\to j}(c)$ represents node v_i's assessment of how likely it is that node v_j has a true label of class c. In addition, $\phi_i(c)$ represents the "non-relational evidence" (e.g., based only on attributes) for v_i having class c, and $\psi_{ij}(c',c)$ represents the "compatibility function" which describes how likely two nodes of class c and c' are linked together (in terms of Markov networks, this represents the potential functions defined by the pairwise cliques of linked class nodes). Given these two sets of functions, Yedidia et al. (2000) show the belief that node i has class c can be calculated as follows:

$$b_i(c) = \alpha\phi_i(c)\prod_{k\in N_i} m_{k\to i}(c) \tag{1}$$

where α is a normalizing factor to ensure that $\sum_{c\in C} b_i(c) = 1$ and N_i is the neighborhood function defined as:

$$N_i = \{v_j | \exists (v_i, v_j) \in E\} .$$

The messages themselves are computed recursively as:

$$m'_{i\to j}(c) = \alpha \sum_{c'\in C} \left(\phi_i(c')\psi_{ij}(c',c) \prod_{k\in N_i\setminus j} m_{k\to i}(c') \right) . \tag{2}$$

Observe that the message from i to j incorporates the beliefs of all the neighbors of i (N_i) *except* j itself. $m'_{i\to j}(c)$ is the "new" value of $m_{i\to j}(c)$ to be used in the next iteration.

For CC, we need a model that generalizes from the training nodes to the test nodes. The above equations do not provide this, since they have node-specific potential functions (i.e., ψ_{ij} is specific to nodes i and j). Fortunately, we can represent each potential function as a log-linear combination of generalizable features, as commonly done for such Markov networks (e.g., Della Pietra et al., 1997; McCallum et al., 2000a). More specifically for CC, Taskar et al. (2002) used a log-linear combination of functions that indicate the presence or absence of particular attributes or other features. Several papers (e.g., Sen and Getoor, 2006; Sen et al., 2008) have described a general model on how to accomplish this, but do not completely explain how to perform the computation. For a slight loss in generality (e.g., assuming that our nodes are represented by a simple attribute vector), we now describe how to perform *LBP* for CC on an undirected graph. In particular, let N_A be the number of attributes, $\mathcal{D}_h$ be the domain of attribute h, and $w_{c,h,k}$ be a learned weight indicating how strongly a value of k for attribute h indicates that a given node has class c. In addition, let $f_i(h,k)$=1 iff the h^{th} attribute of node i is k (i.e., $x_{ih} = k$). Then

$$\phi_i(c) = exp\left(\sum_{h\in\{1..N_A\}} \sum_{k\in\mathcal{D}_h} exp(w_{c,h,k})\, f_i(h,k)) \right)$$

which is a special case of logistic regression. We likewise define similar learned weights of the form $w_{c,c'}$ that indicate how likely a node with label c is linked to a node with label c', yielding the compatibility function

$$\psi_{ij}(c,c') = exp(w_{c,c'}) .$$

LBP_classify $(V, E, X, Y^K, w, C, N, AlgType)$=
// V=nodes, E=edges, X=attribute vectors, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// w=learned params., C=set of class labels, N=neighborhood funct., $AlgType$=*LBP* or LBP_{NC}

1 **for each** $(v_i, v_j) \in E$ such that $v_j \in V^U$ **do** // Initialize all messages
  for each $c \in C$ **do**
    if $(v_i \in V^K)$ // If class is known (y_i), set message to its
      $m_{i \to j}(c) \leftarrow \alpha \cdot exp(w_{y_i,c})$ // final, class-specific value
    else // Otherwise, message starts with same value
      $m_{i \to j}(c) \leftarrow \alpha$ // for every class, but will vary later

2 **while** (messages are still changing)

3 **for each** $(v_i, v_j) \in E$ such that $v_j \in V^U$ **do** // Perform message passing
    for each $c \in C$ **do**
      $m'_{i \to j}(c) \leftarrow \alpha \sum_{c' \in C} \left(\phi_i(c') exp(w_{c',c}) \prod_{k \in N_i \setminus j} m_{k \to i}(c') \right)$

4 **for each** $(v_i, v_j) \in E$ such that $v_j \in V^U$ **do**
    if ($AlgType$ = *LBP*) // For *LBP*, copy new messages for use in
      $m_{i \to j}(c) \leftarrow m'_{i \to j}(c)$ // next iteration
    else
      $c' \leftarrow argmax_{c \in C}(m'_{i \to j}(c))$ // For LBP_{NC}, select most likely label for node
      for each $c \in C$ **do** // Treat selected label the same as a "known"
        $m_{i \to j}(c) \leftarrow exp(w_{c',c})$ // label for use in the next iteration

5 **for each** node $v_i \in V^U$ **do** // Compute final beliefs
  for each $c \in C$ **do**
    $b_i(c) \leftarrow \alpha \phi_i(c) \prod_{k \in N_i} m_{k \to i}(c)$

6 **return** $\{\vec{b}_i\}$ // Return final beliefs

Figure 4: Algorithm for loopy belief propagation (LBP). α is a normalization factor.

As desired, the compatibility function is now independent of specific node identifiers, that is, it depends only upon the class labels c and c', not i and j. We use conjugate gradient descent to learn the weights (cf., Taskar et al., 2002; Neville and Jensen, 2007; Sen et al., 2008).

Finally, we must consider how to handle messages from nodes with a "known" class label. Suppose node v_i has known class y_i. This is equivalent to having a node where the non-relational evidence $\phi_i(c) = 1$ if c is y_i and zero otherwise. Since y_i is known, node v_i is not influenced by its neighbors. In that case, using Equation 2 (with an empty neighborhood for the product) yields:

$$m_{i \to j}(c) = \alpha \sum_{c' \in C} \phi_i(c') \psi_{ij}(c', c) = \alpha \cdot \psi_{ij}(y_i, c) = \alpha \cdot exp(w_{y_i,c}) \; . \tag{3}$$

Given these formulas, we can now present the complete algorithm in Figure 4. In Step 1, the messages are initialized, using Equation 3 if v_i is a known node; otherwise, each value is set to α (creating a uniform distribution). Steps 2-4 performs message passing until convergence, based on Equation 2. Finally, step 5 computes the final beliefs using Equation 1 and step 6 returns the results.
Types of Caution Used: Like *Gibbs*, *LBP* exercises caution by reasoning based on the estimated label uncertainty, but in a different manner. Instead of sampling from the estimated distribution, *LBP* in step 3 directly updates its beliefs using all of its current beliefs, so that the new beliefs reflect the underlying uncertainty of the old beliefs. In particular, this uncertainty is expressed

WVRN_RL_classify $(V, Y^K, n, C, \vec{b}_{prior}, N, \Gamma)$=
// V=nodes, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$), n=# of iterations
// C=set of class labels, $\vec{b}_{prior}$=class priors, N=neighborhood funct., Γ=decay factor

1 **for each** node $v_i \in V^K$ **do** // Create belief vector for each known label
  $\vec{b}_i \leftarrow makeBeliefsFromKnownClass(|C|, y_i)$ // (all zeros except at index for class y_i)
  for each node $v_i \in V^U$ **do** // Create initial beliefs for unknown labels
  $\vec{b}_i \leftarrow \vec{b}_{prior}$ // (using class priors as initial setting)
2 **for** $h = 0$ **to** n **do** // Iteratively re-compute beliefs
3 **for each** node $v_i \in V^U$ **do** // Compute new distribution for each node
  $\vec{b'}_i \leftarrow \frac{1}{|N_i|} \sum_{v_j \in N_i} \vec{b}_j$ // by averaging neighbors' distributions
4 **for each** node $v_i \in V^U$ **do** // Perform simulated annealing
  $\vec{b}_i \leftarrow \Gamma^h \vec{b'}_i + (1 - \Gamma^h)\vec{b}_i$
5 **return** $\{\vec{b}_i | v_i \in V^U\}$ // Return belief distribution for each node

Figure 5: Algorithm for $wvRN_{RL}$. Based on Macskassy and Provost (2007), we use $n = 100$ iterations with a decay factor of $\Gamma = 0.99$.

by the continuous-valued numbers that represent each message $m_{i \to j}$. *LBP* can also benefit from cautious learning with PLUL; in this case, PLUL influences the $w_{c,h,k}$ and $w_{c,c'}$ weights that are learned (see Section 6.4).

5.3.2 LBP_{NC}

LBP_{NC} is identical to *LBP* except that after the new messages are computed in step 3, in step 4 LBP_{NC} picks the single most likely label c' to represent the message from v_i to v_j. LBP_{NC} then treats c' as equivalent to a "known" label y_i for v_i and re-computes the appropriate message $m_{i \to j}(c)$ using Equation 3.

Types of Caution Used: Like *ICA* and $Gibbs_{NC}$, LBP_{NC} is non-cautious because it uses all available labels for relational feature computation and always picks the single most likely label based on the new predictions. In essence, the "pick most likely" step transforms the soft-labeling *LBP* algorithm into the hard-labeling LBP_{NC} algorithm, removing cautious inference just as the "pick most likely" step did for $Gibbs_{NC}$. However, LBP_{NC}, like *LBP*, can still benefit from cautious learning with PLUL.

5.4 wvRN Family of Algorithms

Figure 5 displays pseudocode for $wvRN_{RL}$, a soft-labeling algorithm. For simplicity, we present the related, hard-labeling variants $wvRN_{ICA+C}$ and $wvRN_{ICA+NC}$ separately in Figure 6. Each of these is a relational-only algorithm; Section 7.9 will discuss variants that incorporate attribute information.

5.4.1 $wvRN_{RL}$

$wvRN_{RL}$(Weighted-Vote Relational Neighbor, with relaxation labeling) is a relational-only CC algorithm that Macskassy and Provost (2007) argued should be considered as a baseline for all CC

WVRN_ICA_classify $(V, Y^K, n, C, \vec{b}_{prior}, N, AlgType)$=
// V=nodes, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$), n=# of iters., C=class labels
// $\vec{b}_{prior}$=class priors, N=neighborhood function, $AlgType$=$wvRN_{ICA+C}$ or $wvRN_{ICA+NC}$

1 **for each** node $v_i \in V^U$ **do**
  switch ($AlgType$): // Set initial value for unknown labels...
    case ($wvRN_{ICA+C}$): $y_i \leftarrow$ '?' // ...start labels as *missing*
    case ($wvRN_{ICA+NC}$): $y_i \leftarrow sampleDist(\vec{b}_{prior})$ // ...or sample label from class priors
2 **for** h = 0 **to** n **do** // Iteratively re-label the nodes
3 **for each** node $v_i \in V^U$ **do**
    $N'_i \leftarrow \{v_j \in N_i | y_j \neq$ '?' $\}$ // Find all non-*missing* neighbors
    if $(|N'_i| > 0)$ // New label is the most common label
      $y'_i \leftarrow argmax_{c \in C} | \{v_j \in N'_i | y_j = c\} |$ // amongst those neighbors
    else $y'_i = y_i$ // If no such neighbors, keep same label
4 **for each** node $v_i \in V^U$ **do** // After all new labels are computed,
    $y_i \leftarrow y'_i$ // update to store the new labels
5 **return** $\{y_i | v_i \in V^U\}$ // Return est. class label for each node

Figure 6: Algorithm for $wvRN_{ICA+C}$ and $wvRN_{ICA+NC}$. This is a "hard labeling" version of $wvRN_{RL}$; each of the 5 steps corresponds to the same numbered step in Figure 5. We use $n = 100$ iterations.

evaluations. At each iteration, each node *i* updates its estimated class distribution by averaging the current distributions of each of its linked neighbors. $wvRN_{RL}$ ignores all attributes (non-relational features). Thus, $wvRN_{RL}$ is useful only if the test set links to some nodes with known labels to "seed" the inference process. Macskassy and Provost showed that this simple algorithm can work well if the nodes exhibit strong homophily and enough labels are known.

Step 1 of $wvRN_{RL}$ (Figure 5) initializes a belief vector for every node, using the known labels for nodes in V^K, and a class prior distribution for nodes in V^U. For each node, step 3 averages the current distributions of its neighbors, while step 4 performs simulated annealing to ensure convergence. Step 5 returns the final beliefs. For simplicity, we omit edge weights from the algorithm's description, since our experiments do not use them.

Types of Caution Used: Since $wvRN_{RL}$ computes directly with the estimated label distributions, it exercises cautious inference in the same manner as *LBP*. However, unlike the other CC algorithms, it does not learn from a training set, and thus cautious learning with PLUL does not apply.

5.4.2 $wvRN_{ICA+C}$ AND $wvRN_{ICA+NC}$

Figure 6 presents a hard-labeling alternative to $wvRN_{RL}$. Each of the five steps mirror the corresponding step in the description of $wvRN_{RL}$. In particular, for node v_i, step 3 computes the most common label among the neighbors of v_i (the hard-labeling equivalent of averaging the distributions), and step 4 commits the new labels without annealing.

However, with a hard-labeling algorithm, the initial labels for each node become very important. The simplest approach would be to initialize every node to have the most common label from the prior distribution. However, that approach could easily produce interlinked regions of labels that

that were incorrect but highly self-consistent; leading to errors even when many known labels were provided. Instead, Macskassy and Provost (2007) suggest initializing each node $v_i \in V^U$ to *missing* (indicated in Figure 6 by a question mark), a value that is ignored during calculations. They call the resulting algorithm wvRN-ICA; here we refer to it as $wvRN_{ICA+C}$. A *missing* label remains for node v_i after iteration h if during that iteration every neighbor of v_i was also *missing*.

Alternatively, a simpler algorithm is to always compute with all neighbor labels (do not initialize any to *missing*), but initialize each label in V^U by sampling from the prior distribution. We call this algorithm $wvRN_{ICA+NC}$. This process is the hard-labeling analogue of $wvRN_{RL}$'s approach: instead of initializing *each* node with the prior distribution, with $wvRN_{ICA+NC}$ sampling initializes the *entire set* so that it represents, in aggregate, the prior distribution.

Types of Caution Used: $wvRN_{ICA+NC}$ always uses the estimated label of every node, without regard for how certain that estimate is. Thus, it does not exhibit cautious inference. However, $wvRN_{ICA+C}$ does exhibit cautious inference, although this effect was not discussed by prior work with this algorithm. In particular, during the first iteration $wvRN_{ICA+C}$ uses only the certain labels from Y^K, since all nodes in Y^U are marked *missing*. These known labels are used to estimate labels for every node in V^U that is directly adjacent to some node in V^K. In subsequent iterations, $wvRN_{ICA+C}$ uses both labels from Y^K and labels from Y^U that have been estimated so far. However, the labels estimated so far are likely to be more reliable than later estimations, since the former labels are from nodes that were closer to at least one known label. Thus, in a manner similar to ICA_C's gradual commitment of labels based on confidence, $wvRN_{ICA+C}$ gradually incorporates more and more estimated labels into its computation, where more confident labels (those closer to known nodes) are incorporated sooner. This effect causes $wvRN_{ICA+C}$ to exploit estimated labels more cautiously.

5.5 Parameter Learning for Uncertain Labels (PLUL)

CC algorithms typically train a local classifier on a fully-labeled training set, then use that local classifier with some collective inference algorithm to classify the test set. Unfortunately, this results in asymmetric training and test phases: since all labels are known in the training phase, the learning process sees no uncertainty in relational feature values, unlike the reality of testing. Moreover, the classifier's training is unaffected by the type of collective inference algorithm used, and how (if at all) that collective algorithm attempts to compensate for the uncertainty of estimated labels during testing. Consequently, the learned classifier may tend to produce poor estimates of important parameters related to the relational features (e.g., feature weights, conditional probabilities). Even for CC algorithms that do not use a local classifier, but instead take a global approach that learns over the entire training graph (as with *LBP* and relaxation labeling), the same fundamental problem occurs: if autocorrelation is present, then parameters learned over the fully labeled training set tend to overstate the usefulness of relational features for testing, where estimated labels must be used.

To address these problems, we developed PLUL (Parameter Learning for Uncertain Labels). PLUL is based on standard cross-validation techniques for performing automated parameter tuning (e.g., Kohavi and John, 1997). The key novelty is not in the cross-validation mechanism, but in the selection of *which* parameters should be tuned and *why*. To use PLUL, we must first select or create an appropriate parameter that controls the amount of impact that relational features have on the resultant classifications. In principle, PLUL could search a multi-dimensional parameter space, but for tractability we select a single parameter that affects all relational features. For instance, when

PLUL_learn ($CCtype, P, lp, V_{Tr}, E_{Tr}, X_{Tr}, Y_{Tr}, V_H, E_H, X_H, Y_H$)=
// *CCtype*=CC alg. to use, P=set of parameter values to consider, lp=labeled proportion to use
// $V_{Tr}, E_{Tr}, X_{Tr}, Y_{Tr}$ = vertices, edges, attributes, and labels from training graph
// V_H, E_H, X_H, Y_H = vertices, edges, attributes, and labels from holdout graph

1 $Y'_H = keepSomeLabels(lp, Y_H)$ // Randomly select lp% of labels to keep; discard others

2 $bestParam \leftarrow \emptyset$ // Initialize variables to track best parameter so far
$bestAcc \leftarrow -1$

3 **for each** $p \in P$ **do** // Iterate over every parameter value

4 // Learn complete CC classifier from fully-labeled training data, influenced by p
$cc = learn\_CC\_classifier(CCtype, V_{Tr}, E_{Tr}, X_{Tr}, Y_{Tr}, p)$

5 // Run CC on holdout graph (with some known labels Y'_H) and evaluate accuracy
$acc \leftarrow execute\_CC\_inference(cc, V_H, E_H, X_H, Y'_H)$

6 // Remember this parameter if it's the best so far
if $(acc > bestAcc)$
$bestParam \leftarrow p$
$bestAcc \leftarrow acc$

7 **return** *bestParam* // Return best parameter found over the holdout graph

Figure 7: Algorithm for Parameter Learning for Uncertain Labels (PLUL). The holdout graph is derived from the original training data and is disjoint from the graph that is used later for testing.

using a k-nearest-neighbor rule as the local classifier, we employ PLUL to adjust the weight w_R of relational features in the node similarity function. PLUL performs automated tuning by repeatedly evaluating different values of the selected parameter, as used by the local classifier, together with the collective inference algorithm (or the entire learned model for *LBP*). For each parameter value, accuracy is evaluated on a holdout set (a subset of the training set). PLUL then selects the parameter value that yields the best accuracy to use for testing.

Figure 7 summarizes these key steps of PLUL and some additional details. First, note that proper use of PLUL requires a holdout set that reflects the test set conditions. Thus, step 1 of the algorithm removes some or all of the labels from the holdout set, leaving only the same percentage of labels (lp%) that are expected in the test set. Second, running CC inference with a new parameter value may require re-learning the local classifier (for *ICA* or *Gibbs*) or the entire learned model (for *LBP*). This is shown in step 4 of Figure 7. Alternatively, for Naive Bayes or k-nearest-neighbor local classifiers, the existing classifier can simply be updated to reflect the new parameter value.

We expect PLUL's utility to vary based upon the fraction of known labels (lp) that are available to the test set. If there are few such labels, there is more discrepancy between the training and test environments, and hence more need to apply PLUL. However, if there are many such labels, then PLUL may not be useful.

Because almost all CC algorithms learn parameters based in some way on relational features, PLUL is widely applicable. In particular, Table 2 shows how we select an appropriate relational parameter to apply PLUL for different CC algorithms. The top of the table describes how to apply PLUL to a local classifier that is designed to be used with a CC algorithm like *ICA* or *Gibbs*. The

Local Classifier (or CC algorithm)	**Parameter set by PLUL (per relational feature)**	**Values tested by PLUL (default in bold)**
Naive Bayes (NB)	Hyperparameter α for Dirichlet prior	**1**, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096
Logistic Regression (LR)	Variance σ^2 of Gaussian prior	5, **10**, 20, 40, 80, 160, 320, 640, 1280, 2560, 512
k-Nearest Neighbor (kNN)	Weight w_R	0.01, 0.03, 0.0625, 0.125, 0.25, 0.5, 0.75, **1.0**, 2.0
LBP	Variance σ^2 of Gaussian prior	5, **10**, 20, 100, 200, 1000, 10000, 100000, 1000000

Table 2: The classifiers (NB, LR, and kNN) and CC algorithm (*LBP*) used in our experiments for which PLUL can be applied to improve performance. The second column lists the key relational parameters that we identified for PLUL to learn, while the last column shows the values that PLUL considers in its cross-validation.

last row demonstrates how it can instead be applied to a global algorithm like *LBP*. For instance, for the NB classifier, most previous research has used either no prior or a simple Laplacian ("add one") prior for each conditional probability. By instead using a Dirichlet prior (Heckerman, 1999), we can adjust the "hyperparameter" α of the prior for each relational feature. Larger values of α translate to less extreme conditional probabilities, thus tempering the impact of relational features. For the kNN classifier, reducing the weight of relational features has a similar net effect. For the LR classifier and the *LBP* algorithm, both techniques involve iterative MAP estimation. Increasing the value of the variance of the Gaussian prior for relational features causes the corresponding parameter to "fit" less closely to the training data, again making the algorithm more cautious in its use of such relational features.

While the core mechanism of PLUL—cross-validation tuning—is common, techniques like PLUL to explicitly compensate for the bias incurred from training on a fully-labeled set while testing using estimated labels have not been previously used for CC. A possible exception is Lu and Getoor (2003a), who appear to have used a similar technique to tune a relational parameter, but, in contrast to this work, they did not discuss its need, the specific procedure, or the performance impact.

PLUL attempts to compensate for the bias incurred from training on the correctly-labeled training set. Alternatively, Kou and Cohen (2007) describe a "stacked model" that learns based on estimated, rather than true labels. While the original goal of this stacked approach was to produce a more time-efficient algorithm, Fast and Jensen (2008) recently demonstrated that this technique, by eliminating the bias between training and testing, does indeed reduce "inference bias." This reduced bias enables the stacked models to perform comparably to Gibbs sampling, even though the stacked model is a simpler, non-iterative algorithm that consequently has higher learning bias. Interestingly, Fast and Jensen (2008) note that the stacked model performs an "implicit weighting of local and relational features," as with PLUL. The stacked model accomplishes this by varying the learning and inference procedure, whereas PLUL modifies only the learning procedure, and thus works with any inference algorithm that relies on a learned model.

5.6 Computational Complexity and the Cost of Caution

For learning and inference, all of the CC algorithms (variants of *ICA*, *Gibbs*, *wvRN*, and *LBP*) use space that is linear in the number of nodes/instances (N_I). *ICA* and *Gibbs* have significant similarities, so we consider their time complexity first. For these two algorithms, the dominant computation costs for inference stem from the time to compute relational features and the time to classify each node with the local classifier. Typically, nodes are connected to a small number of other instances, so the first cost is $O(N_I)$ per iteration. For the second cost, the time per iteration is $O(N_I)$ for NB and LR, and $O(N_I^2)$ for kNN. However, the number of iterations varies significantly. Based on previous work (Neville and Jensen, 2000; McDowell et al., 2007a), we set $n = 10$ for variants of *ICA*; more iterations did not improve performance. In contrast, *Gibbs* typically requires *thousands* of iterations.

Adding or removing cautious inference to *ICA* and *Gibbs* does not significantly change their time complexity. In particular, $Gibbs_{NC}$ has the same complexity as *Gibbs*. ICA_C introduces an additional cost, compared to *ICA*, of $O(N_I log N_I)$ per iteration to sort the nodes by confidence. However, in practice classification time usually dominates. Therefore, the overall computational cost per iteration for all variants of *ICA* and *Gibbs* are roughly the same, but the larger number of iterations for variants of *Gibbs* makes them much more time-expensive than *ICA*, ICA_{Kn}, or ICA_C.

LBP does not explicitly compute relational features, but its main loop iterates over all neighbors of each node, thus again yielding a cost of $O(N_I)$ per iteration under the same assumptions as above. We found that *LBP* inference was comparable in cost to that of *ICA*, which agrees with Sen and Getoor (2007). However, training the *LBP* classifier is much more expensive than training the other algorithms. *ICA* and *Gibbs* only require training the local classifier, which involves zero to one passes over the data for kNN and NB, and a relatively simple optimization for LR. On the other hand, training *LBP* with conjugate gradient requires executing *LBP* inference many times. We found this training to be at least an order of magnitude slower than the other algorithms, as also reported by Sen and Getoor (2007). LBP_{NC} has the same theoretical and practical time results as *LBP*.

wvRN is the simplest CC algorithm, since it requires no feature computation and the key step of each iteration is a simple average over the neighbors of each node. As with previous algorithms, assuming a small number of neighbors for each node yields a total time per iteration of $O(N_I)$. Prior work (Macskassy and Provost, 2007) suggested using a somewhat larger number of iterations (100) than with *ICA*. Nonetheless, in practice *wvRN*'s simplicity makes it the fastest algorithm.

Finally, all of the algorithms, except for *wvRN*, can be augmented with cautious learning via PLUL. Executing PLUL requires repeatedly running the CC algorithm with different values of the selected parameter. We used 9-13 different parameter values, and hence the cost of PLUL vs. not using PLUL is about one order of magnitude.

6. Evaluation Methodology

This section describes our hypotheses and the method that we use to evaluate them.

6.1 Hypotheses

Table 3 summarizes our five hypotheses. As described in Section 1, we expect cautious behaviors to be more important when there is a higher probability of incorrect relational inference. Thus, each

Data characteristic	**Type of caution considered**	**Hypothesis:** relative gain of caution will increase as value of characteristic...
Autocorrelation	Inference	...increases (H1)
Attribute predictiveness	Inference	...decreases (H2)
Link density	Inference	...decreases (H3)
Labeled proportion	Inference	...decreases (H4)
Labeled proportion	Learning	...decreases (H5)

Table 3: The five hypotheses that we investigate.

hypothesis varies one data characteristic that impacts the likelihood of such errors. In particular, hypotheses H1-H4 vary a data characteristic to measure the impact of cautious inference, which Section 7 will evaluate for different pairs of cautious and non-cautious inference algorithms. We define the "relative gain of cautious inference" as the difference between the accuracies of two such algorithms (e.g., *Gibbs* vs. $Gibbs_{NC}$). Hypothesis H5 also varies a data characteristic, but does so to measure the "relative gain of cautious learning" (i.e., comparing performance with vs. without PLUL).

- **H1: The relative gain of cautious inference increases with increasing autocorrelation**. Larger autocorrelation implies that relations are more predictive, and will be learned as such by the classifier. This magnifies the impact that an error in a predicted label can have on linked nodes. Therefore, we expect cautious inference algorithms to improve classification by a greater margin in such cases.

- **H2: The relative gain of cautious inference increases with decreasing attribute predictiveness (*ap*)**. Decreased *ap* implies a greater potential of errors/uncertainty in the predicted labels. The effect of cautiously using uncertain labels should be greater in such cases.

- **H3: The relative gain of cautious inference increases with decreasing link density (*ld*).** When the number of links is high, a single mispredicted label has relatively little impact on its neighbors. As the number of links decreases, however, a single misprediction can cause larger relational feature uncertainty, increasing the need for caution.

- **H4: The relative gain of cautious inference increases with decreasing labeled proportion (*lp*).** When *lp* is high, only a few of each node's neighbors have estimated labels (most are known with certainty). Consequently, there is less uncertainty in relational feature values, and less need to use estimated labels cautiously.

- **H5: The relative gain of cautious learning with PLUL increases with decreasing labeled proportion(*lp*).** As with H4, when *lp* is high there is less uncertainty in the relational features. Thus there is less disparity between the fully correct training set (where classifier parameters were learned) and the test set. Consequently, we expect PLUL, which compensates for any such disparity, to matter less when *lp* is high.

6.2 Tasks

We will evaluate three general tasks (see Section 2.3):

Parameter	Abbrev.	Values tested (defaults in bold)
Nodes per graph	N_I	250
Number of class labels	N_C	5
Number of attributes	N_A	10
Degree of homophily	*dh*	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, **0.7**, 0.8, 0.9
Link density	*ld*	0.1, **0.2**, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Attribute predictiveness	*ap*	0.1, 0.2, 0.3, 0.4, 0.5, **0.6**, 0.7, 0.8, 0.9
Labeled proportion	*lp*	**0%**, **10%**, 20%, 40%, **50%**, 60%, 80%

Table 4: Synthetic data parameters. Defaults were chosen based on averages from Cora and Citeseer, two commonly studied data sets for CC.

1. **Out-of-sample task:** Here the test set does not contain or link to any known nodes, as with Neville and Jensen (2000), Taskar et al. (2002), and Sen and Getoor (2006).

2. **Sparse in-sample task:** Here some of the test nodes, but only a few, have known labels (we use 10%). We focus particularly on this task, because some researchers argue that it is the most realistic scenario, since often networks are large, and acquiring known labels is expensive (Bilgic and Getoor, 2008). This was the primary scenario considered by the recent work of McDowell et al. (2007a,b), Bilgic and Getoor (2008), and Gallagher et al. (2008).

3. **Dense in-sample task:** Here a substantial number of test nodes may have known labels (we use 50%). This task was the one recently evaluated by Sen et al. (2008).

6.3 Data

We evaluate the hypotheses over both synthetic and real-world data sets, which we describe below. We use the synthetic data to highlight how different data characteristics affect the relative gain of cautious behaviors, then the real-world data sets to validate these findings.

6.3.1 SYNTHETIC DATA

We use a synthetic data generator (see Table 4) with two components: a Graph Generator and an Attribute Generator. The Graph Generator has four inputs: N_I (the number of nodes/instances), N_C (the number of classes), *ld* (the link density), and *dh* (the degree of homophily). For each link, *dh* controls the probability that the linked nodes have the same class label; higher values yield higher autocorrelation (see Appendix A for details). The final number of links is approximately $N_I/(1-ld)$, and the final link degrees follow a power law distribution, which is common in real networks (Bollobás et al., 2003). The Graph Generator is identical to that used by Sen et al. (2008); see that article for more detail.

To make this a practical study, we chose default parameter values that mimic characteristics of two frequently studied CC data sets, Cora and Citeseer (McDowell et al., 2007a; Neville and Jensen, 2007; Sen et al., 2008). In particular, N_C=5 classes and Table 4 shows additional default values. We chose N_I=250 nodes, a smaller value than with Cora/Citeseer, to reduce CC execution time, but larger values did not change the performance trends.

The Attribute Generator generates 10 (N_A) binary attributes. Our design for it is motivated by our observations of common CC data sets. We found that, unlike synthetic models used in prior studies, different attributes vary in their utility for class prediction. To simulate this, we associate each attribute h with a particular class c_m, where $m = h \ mod \ N_C$, and vary the strength of each attribute's predictiveness based on the value of h. In particular, for node v_i with class y_i, the probability that v_i's h^{th} attribute x_{ih} has value 1 depends upon the class y_i as follows:

$$P(x_{ih} = 1 | y_i = c_k) = \begin{cases} 0.15 + (ap - 0.15) \cdot \frac{h}{N_A - 1} & \text{if } k = h \ mod \ N_C \\ 0.1 & \text{if } k = (h-1) \ mod \ N_C \\ 0.05 & \text{if } k = (h+1) \ mod \ N_C \\ 0.02 & \text{otherwise.} \end{cases}$$

The first line indicates that, when y_i $(= c_k)$ is the class associated with attribute h (i.e., $k = h \ mod \ N_C$), then $P(x_{ih} = 1 | y_i = c_k)$ ranges from 0.15 for $h = 0$ to ap (a constant representing the strength of attribute predictiveness) for $h = 9$. As a result, each of the five classes has two attributes associated with that class, but some classes have associated attributes that are more useful for prediction. However, x_{ih} may also be 1 when y_i is some other class besides an "associated class"; the next three lines encode this class ambiguity. This ambiguity/noise is based on our observations of Cora and Citeseer and is similar to the binomial distribution used by Sen et al. (2008).

Finally, we use a parameter for test set generation called *lp* (labeled proportion), which is the proportion of test nodes with known labels. We use default values of *lp*=0%, *lp*=10%, and *lp*=50% for the three tasks defined in Section 6.2. Nodes to be labeled are selected uniformly at random from the test set until the desired value of *lp* is reached. In contrast, some real data sets are likely to exhibit non-uniform clustering of known nodes. We conjecture that such data sets will have a smaller "effective" *lp*, since each known node will have, on average, fewer direct connections to unknown nodes. For instance, a data set with *lp*=10% may behave more like a data set with *lp*=5% where the labels are more uniformly distributed. Such effects should be examined in future work.

6.3.2 "REAL-WORLD" DATA SETS

We consider the following five "real-world" data sets (see Table 5). "Real-world" is a somewhat subjective term; however, all of the data sets are based on naturally arising networks and have been used in some form for previous research on relational learning.

1. **Cora (McCallum et al., 2000b):** A collection of machine learning publications categorized into seven classes. The relational links are (directed) citations.

2. **Citeseer (Lu and Getoor, 2003a):** A collection of research publications drawn from CiteSeer. The relational links are (directed) citations.

3. **WebKB (Craven et al., 1998):** A collection of web pages from four computer science departments categorized into six classes (Faculty, Student, Staff, Course, ResearchProject, or Other). "Other" is problematic because it is too general, representing 74% of the pages. Like Taskar et al. (2002), we discarded all "Other" pages that did not have at least three outgoing links, yielding a total of 1541 instances of which 30% are Other. The relational links are the (directed) hyperlinks among these pages.

	Cora	CiteSeer	WebKB	HepTH	Terror
Characteristics of entire graph					
Instances/nodes	2708	3312	1541	2194	645
Attributes (non-relat. feats.) available	1433	3703	100	387	106
Attributes used (max)	100	100	100	100	100
Attributes used (default)	20	20	40	40	2
Link/relation directedness	directed	directed	directed	directed	undirected
Type of relational features used	in,out	in,out	in,out,co	in,out	linksto
Class labels	7	6	6	7	6
Total relational features used	14	12	18	14	6
Links per node	3.9	2.7	5.8(64.6)	8.9	9.8
Autocorrelation	0.88	0.83	0.30(0.53)	0.54	0.16
Characteristics of each test set (on average)					
Instances/nodes	400	400	335-469	300	150
Number of folds	5	5	4	5	3
Links per node	2.7	2.7	5.7(61.0)	4.3	12.3
Approx. link density	0.23	0.23	0.64(0.97)	0.53	0.79
Autocorrelation	0.85	0.84	0.38(0.53)	0.64	0.24
Label consistency	0.78	0.75	0.21(0.90)	0.61	0.56
Approximate homophily	0.74	0.70	0.05(0.88)	0.54	0.47

Table 5: Summary of the five real-world data sets used. *in* and *out* features compute separate values based on incoming or outgoing links, while *linksto* features make no such distinction. *co* features are based on virtual co-citation links; nodes A and B are linked via a *co* link if there exists some node C with outgoing links to both A and B. For WebKB, the first statistic listed is computed ignoring co-links, while the statistic in parentheses is computed using *only* co-links. Label consistency is the percentage of links connecting nodes with the same label; Appendix A defines this and approximate homophily. Section 6.9 describes the "default" number of attributes used.

4. **HepTH:** A collection of journal articles in the field of theoretical high-energy physics, derived from the Proximity Hep-Th database (http://kdl.cs.umass.edu/data/hepth). The original data set did not have any single class label, but some pages were classified into topic subtypes. Among pages with one such subtype, we selected all articles belonging to the six most common subtypes, yielding 1404 articles. To create a more connected graph, we also selected all articles with a date after 2001 that linked to at least two of the 1404 pre-selected articles. There were 790 such articles, which we treated as having a class label of "Other." The relational links are the (directed) citations among all 2194 articles.

5. **Terror (Zhao et al., 2006):** A collection of terrorist incidents, drawn from the Profile in Terror project (http://profilesinterror.mindswap.org). The incidents are non-uniformly distributed into six categories: Bombing (44%), WeaponAttack (38%), Kidnapping (14%), Arson (2%), NBCRAttack (1%), and OtherAttack (1%). The relational links indicate (undirected) geographical co-location.

These data sets are intended to demonstrate CC performance on a range of data characteristics. For instance, CC would be expected to be very helpful for Cora and CiteSeer, where autocorrelation is high, but not very helpful for Terror.

6.4 CC Algorithms

We evaluate the ten algorithms listed in Table 1, plus *Content Only* (*CO*), a non-relational baseline that uses only attributes. For each of the four main sections in Table 1, there is one non-cautious variant (*ICA*, $Gibbs_{NC}$, LBP_{NC}, and $wvRN_{ICA+NC}$) and one or two cautious variants (ICA_C, ICA_{Kn}, *Gibbs*, *LBP*, $wvRN_{RL}$, and $wvRN_{ICA+C}$). The *wvRN* algorithms also serve as a collective, relational-only baseline.

Based on previous work (Neville and Jensen, 2000; McDowell et al., 2007a), the *ICA*-based algorithms used $n = 10$ iterations; more iterations did not improve performance. For *Gibbs*, we used 1500 iterations, with a random restart every 300 iterations, and ignored the first 100 iterations after a restart for burn-in. Additional iterations did not improve performance. $Gibbs_{NC}$ converged in far fewer iterations because it does not sample and is deterministic; we used $n = 50$.

For *LBP*, we assumed that each parameter was a priori independent and had a zero-mean Gaussian prior with a default uniform prior variance of $\sigma^2 = 10$, which is similar to the values reported in previous work (e.g., Sen and Getoor 2006; Neville and Jensen 2007). We used MAP estimation to estimate these parameters based on conjugate gradient. σ^2 controls how tightly the parameters fit to the training data; Table 2 shows the alternative values of σ^2 considered by PLUL to constrain this fitting for the relational parameters.

6.5 Classifiers

To account for possible variations in overall CC performance trends due to the effect of the underlying classifier, we tested three local classifiers with each CC algorithm wherever applicable (this excludes *LBP* and *wvRN*). Section 5.5 already described, for each classifier, the key relational feature whose value is learned by PLUL; we now provide more detail on each classifier and its application of PLUL.

The first classifier is Naive Bayes (NB). PLUL was used to learn α for the Dirichlet prior of each relational feature. The second classifier is Logistic Regression (LR). We used MAP estimation with Gaussian priors to learn the parameters for LR; PLUL learned an appropriate variance σ^2 for the prior of each relational feature. The final classifier is k-Nearest Neighbor (kNN); we used k=11. When computing similarity, attributes were assigned a weight of 1. PLUL learned the weight w_R for each relational feature. Weighted similarity was used for voting.

For each classifier, Table 2 shows the specific values considered by PLUL. The "default" value shown (e.g., $\alpha = 1.0$ for NB) was used in two ways. First, the default was used as the parameter value for all attributes. Second, the default was used for a manual setting for the parameter value for all relational features when PLUL is not being used. When PLUL was used, the learned value was used instead for the relational features.

The ICA_C algorithm requires a classifier that can ignore *missing* relational feature values. kNN and NB can do this easily: kNN by dropping the feature from the similarity calculation and NB by skipping the feature in probability computation. For LR, however, dealing with missing values is a current research topic (e.g., Fung and Wrobel 1989), with typical techniques including mean value substitution or multiple imputation. However, for CC the situation is less complex than the more general case, because missing values occur only for the test set, only for relational features, and typically only when all neighbors of a node have missing labels. Thus, we can learn several LR classifiers: one that uses all relational features, and one for each combination of features that may be missing simultaneously (for our data, this is at most 4). Experimentally, we found this

to perform better than mean value substitution, though the difference was slight because missing values were rare. These results are consistent with those of Saar-Tsechansky and Provost (2007) on non-relational data. Section 7.8 discusses this effect in more detail.

6.6 Node Representation

Each node is represented by a set of (non-relational) attributes and relational features. Algorithms based on *LBP* and *wvRN* reason directly with each individual link, and their algorithms thus directly define the effective relational features used. Approaches based on *ICA* and *Gibbs*, however, use some kind of aggregation function to compute their relational feature values. We first describe the possible aggregation functions for these features, then separately describe the complete representation for the synthetic and real data.

6.6.1 RELATIONAL FEATURES CONSIDERED

We considered three different types of relational features:

- **Count**: This type represents the number of neighbors that belong to a particular class. For each node *i*, there is one such feature $f_i(c)$ per class label c. The value of $f_i(c) = Neighbors_i(c)$, which is the number of nodes linked to node *i* that have a known or current estimated label of *c*. For instance, in step C of Figure 1, $f_2(P) = 1$ and $f_2(S) = 2$.

- **Proportion**: This feature is like "count", except that the feature value represents the proportion of neighbors that have a particular label, rather than the raw number of such neighbors. For this feature, $f_i(c) = Neighbors_i(c)/Neighbors_i(*)$, where $Neighbors_i(*)$ is the number of nodes linked to node *i* that have any current label (known or estimated, but, for ICA_C, excluding those nodes whose label was set to *missing* because of low confidence). If $Neighbors_i(*)$ is zero, then $f_i(c)$ is set to *missing.* For example, if proportion features were being used, then the feature values for step C of Figure 1 would be $f_2(P) = 1/3$ and $f_2(S) = 2/3$.

- **Multiset**: Proportion and count features aggregate the labels of a node's neighborhood to produce a single numerical value for each possible label. During inference, this aggregate value is then compared against the mean value from the training set (with NB or LR), or compared against the aggregate values for nodes in the training set (with kNN). In contrast, a "multiset" feature uses a single multiset to represent the current labels of a node's neighbors. For instance, if multiset features were used, then for step C of Figure 1, $f_2 = \{P,S,S\}$. This has the same information content as with count features, but can be exploited differently by some local classifiers. In particular, during NB inference, each label in the multiset (excluding *missing* labels) is separately used to update the conditional probability that a node has true label *c*. This is the "independent value" approach introduced by Neville et al. (2003b) and used by Neville and Jensen (2007). However, this approach does not directly apply to LR or kNN.

6.6.2 SYNTHETIC DATA NODE REPRESENTATION

Each node is represented by ten binary attributes and some relational features. Because representation choices can affect how well a CC algorithm handles the uncertainty of estimated labels, for each local classifier-based algorithm we considered count and proportion relational features, as well

as multiset features when using NB. For each trial, we evaluated the two or three possible types of relational features with cross-validation (evaluating accuracy on the holdout set), then selected the feature type with the highest accuracy to use for testing. When PLUL was used, PLUL was also applied to each feature type; the best performance (on the holdout set) reported by PLUL for each feature type was then used for this feature selection. Section 7.8 describes which feature types were chosen most often for each local classifier. Since there are 5 class labels for the synthetic data and links are undirected, there were 5 relational features when using count or proportion features, and 1 relational feature (whose value is a multiset) when using multiset.

6.6.3 REAL-WORLD DATA NODE REPRESENTATION

For all five data sets we used binary attributes that indicated the presence or absence of a particular word. For WebKB, these words were from the body of each HTML page; we selected the 100 most frequent such words, which was all that was available in our version of the data set. For symmetry, and because adding more words had a small impact on performance, we likewise set up the remaining data sets to select 100 words as attributes. For Cora and CiteSeer, these words were taken from the body of the publications; as with previous work (McDowell et al., 2007a) we selected the 100 words with the highest information gain in the training set to use. For Terror, the words come from hand-written descriptions of each incident provided with the data set; we selected the first 100 of the 106 available attributes. For HepTH, we selected, based on information gain, the 100 highest-scoring words from the article title or the name of the corresponding journal.

For relational features, we again considered the proportion, multiset, and count features, and selected the best feature type using cross-validation as described above. All of the data sets except Terror had directed links. For these data sets, we computed separate relational feature values based on incoming and outgoing links. In addition, previous work has shown WebKB to have much stronger autocorrelation based on co-citation links than on direct links (see Table 5). However, using such links can sometimes be problematic. Thus, we evaluate two data sets: "WebKB" and "WebKB+co". For WebKB, algorithms use in and out links ("direct" links). For WebKB+co, algorithms use in, out, and co-links, except *wvRN* uses only co-links, as suggested by Macskassy and Provost (2007) (see Section 7.6).

6.7 Training/Test Splits Generation

For the synthetic data, we generate training, holdout, and test graphs that are disjoint. Likewise, for WebKB, the data was already divided into four splits (one for each department) that can be used for cross-validation.

For the other real data sets, we must manually construct training and test splits from the original graph. Sen et al. (2008) suggest a technique based on snowball sampling that involves picking a random starting node and iteratively growing a split around that node, where the class of the next node to be selected is sampled from the overall class distribution. However, we found that low graph connectivity often prevented the algorithm from producing a final subgraph whose class distribution resembled the whole graph's. Instead, we created the following technique, *similarity-driven snowball sampling*: given the whole graph G, pick a random starting node and add it to the split G_1. At each step, consider the *frontier* F of G_1 (all those nodes not in G_1 that link to some node in G_1). Among all labels c that exist in F, select the class label c' such that adding some node of label c' to G_1 would maximize the similarity (inverse Euclidean distance) of the class distributions

of G_1 and G. Given this c', randomly select some node in F of class c' and add it to G_1. Repeat this random selection and insertion until G_1 is of the desired size.

We run this algorithm in parallel for N_S different subgraphs, using N_S different seeds, and permit each node to be inserted into only one subgraph. This results in N_S disjoint splits that have similar class distributions and that can be used for N_S-fold cross validation. We set $N_S = 5$ for Cora, Citeseer, and HepTH, and $N_S = 3$ for the smaller Terror.

Table 5 shows some of the characteristics of the generated test sets vs. the original, complete graphs. In general, the autocorrelation and number of links per node are similar, indicating that the sampling procedure did not dramatically change the average characteristics of the graph. While the splitting procedure effectively removes links, the average degree of the test sets may still be greater than with the original graph if high-degree subsets of the original are selected.

6.8 Test Procedure

We first consider the synthetic data. For each control condition (i.e., data generated with a combination of *dh*, *ap*, *ld*, and *lp* values, see Table 4) we ran 25 random trials. For each trial, we generated training, holdout, and test data sets of 250 nodes each. All training is performed on the fully labeled training set. The holdout set, when not used for PLUL, was merged with the training set. We measured classification accuracy on the test set, excluding all nodes with "known" labels.

For the real-world data sets, each experiment involves using all of the relational features shown in Table 5 and a fixed number of attributes (N_A). We vary N_A from 2 to 100 (recall that for all data sets 100 attributes were selected for experimentation). For each setting of N_A, we perform N_S-fold cross-validation, where N_S is 3, 4, or 5, depending on the data set. Each one of these 3 to 5 trials is associated with one subgraph (the test set), and the remaining 2-4 subgraphs comprise the training set. We then apply PLUL by training on half of the training set and using the other half as the holdout set. After PLUL selects the best parameter setting, we re-train on the whole training set and evaluate accuracy on the test set. If PLUL is not used, training likewise uses the whole training set.

We report results with accuracy in order to ease comprehension of the results and to facilitate comparison with some of the most relevant related work (e.g., Sen et al., 2008; Macskassy and Provost, 2007). Results with area under the ROC curve (AUC) for the majority class demonstrated similar trends.

6.9 Statistical Analysis

We conducted two distinct types of analysis. First, to compare algorithms for a single control condition, we used a one-tailed paired t-test accepted at the 95% confidence level. For every such test each "test point" is the accuracy over a single trial's test graph. For example, for the synthetic data there are 25 trials for each control condition, and thus a single t-test compares 25 pairs of accuracies (e.g., ICA_C vs. *ICA*). In all cases the test graphs used by these t-tests are disjoint, for both the synthetic and the real data.

Second, we performed linear regression slope tests. In particular, for hypotheses H1-H4, we compared two algorithms (e.g., ICA_C vs. *ICA*) for each independent variable X (e.g., *ld*) as follows: For each trial, we computed the difference in the algorithms' classification accuracies (e.g., for the synthetic data, 225 such differences for 25 trials and 9 values of *ld*). We performed linear regression ($Y = a + bX$), where the accuracy difference is the dependent variable Y and X is the independent variable (e.g., *ld*). The estimated value of slope b, when non-zero, indicates an increasing (+)

or decreasing ($-$) trend. Regression produces a p value associated with the slope that indicates the significance level for hypothesis testing; we accept when $p < 0.05$. For hypothesis H5, the equations are the same but we compare a single CC algorithm with and without PLUL.

For the synthetic data, the analysis is straight-forward and we use the data generation parameters *dh*, *ap*, *ld*, and *lp* as the independent variable for regression. Analysis for the real data sets requires more explanation. For instance, each computed subgraph of a data set has similar autocorrelation, so regression for H1 (where autocorrelation is the X value) cannot be performed on a single data set. Instead, we combine the trials of all the real data sets into one analysis, where the independent variable is the measured autocorrelation of the corresponding data set (we include WebKB, but exclude WebKB+co because it's not clear how to compute its autocorrelation with direct links combined with co-citation links). In addition, our results show that when attribute predictiveness is high, there is less need for caution. Thus, to prevent any interactions between autocorrelation and caution from being obscured by high attribute predictiveness, we use fewer than 100 attributes for these experiments. In particular, for each data set we evaluated the baseline *CO* algorithm with varying numbers of attributes N_A, and selected the number that yields an average accuracy closest to 50%. Table 5 shows the resulting default number of attributes for each data set.

For H2 (attribute predictiveness), we can directly vary the number of attributes, so we can perform regression for each data set separately. However, attribute predictiveness is typically not a linear function of the number of attributes. Thus, for H2 we perform regression where the dependent variable is the accuracy of *CO* for each trial (as a surrogate for attribute predictiveness).

We do not directly evaluate H3 for the real data sets (see Section 7).

For H4 and H5 (varying labeled proportion), we directly vary *lp*, so we can compute separate results for each data set. Moreover, *lp* is suitable for direct use as the dependent variable. As with H1, we use the default number of attributes for each data set in order to avoid having high attribute predictiveness obscure the interaction of caution and *lp*. We omit nonsensical points (e.g., *wvRN* when *lp*=0%) from all of the analyses.

Finally, for each hypothesis we also perform a *pooled analysis*. For the synthetic data, this involves pooling the results of all the cautious CC algorithms, then performing the slope regression test. For the real-world data, we pool the results across both the CC algorithms and each of the real data sets. In addition, to account for differences in the data sets, we perform a multiple regression analysis that includes autocorrelation as one of the input variables (except for H1). In particular, we fit the data to the line $Y = a + b_1X_1 + b_2X_2$, where X_1 is the variable in question (e.g., *lp* for H4 or H5) and X_2 is the autocorrelation of the data set. The X_2 term factors out differences due only to autocorrelation, thus making the other trends more clear. The p-value corresponding to b_1 is then used for hypothesis testing.

6.10 Implementation Validation

To validate the implementation of our algorithms, we replicated three different synthetic data generators: those used by Sen and Getoor (2006), Neville and Jensen (2007), and Sen et al. (2008). We then replicated some of the experiments from these papers. While several of our CC algorithm variants were not evaluated in any of these earlier papers, we were able to compare results for *ICA*, *Gibbs*, and *LBP*, with the LR and NB classifiers as appropriate, and found very consistent results. Section 8.4 discusses one exception.

LBP is the most challenging algorithm to implement and to get to converge. To deal with such problems, Sen et al. (2008) seeded *LBP*'s learning process with weights learned from *ICA*. Alternatively, we found that seeding with values estimated from empirical counts over the data, combined with limiting the maximum step size of the search to prevent oscillation, worked well. With these enhancements, *LBP* achieved equivalent accuracy to that reported by Sen and Getoor (2006), and, when PLUL was applied, significantly improved it for the cases of high homophily and link density (where *LBP*'s accuracy had been very poor). In contrast, we found that *LBP* could replicate the performance of Sen et al. (2008), but that in this case PLUL had little effect. Section 8.4 explains the data characteristics of that study (effectively high *lp*) that led to this result.

7. Evaluation Results

This section presents our experimental results. Section 7.1 presents a summary of the results, Section 7.2 explains how we present the detailed results, and subsequent sections discuss these detailed results for each hypothesis. We focus on the sparse in-sample task, so *we accept a hypothesis if it is confirmed, for the* lp*=10% case, by the pooled analysis on both the synthetic data and the real-world data*. Hypotheses H4-H5 involve varying *lp*; here we accept the hypothesis if confirmed on both the synthetic and real data.

When a local classifier is needed, all results below use NB by default. We found that NB's performance was better or equivalent to that of LR and kNN in almost every case (see Section 8.4), for both the synthetic and real data sets, and that using LR or kNN led to very similar performance trends. Below we mention some of the results for LR and kNN; see the online appendix for more detail. In addition, PLUL is used everywhere unless otherwise specified; see analysis and motivation in Section 7.7.

7.1 Summary of Results

Tables 6-8 summarize our overall results for hypotheses H1-H5. Each table presents results for the synthetic data on the left and (where applicable) for the real data sets on the right. Each reported value represents the estimated slope of the line measuring the difference between a cautious and a non-cautious CC algorithm as the corresponding x-parameter (e.g., autocorrelation) is varied (see Section 6.9). Only values that were statistically different from zero are reported; otherwise a dash is shown. Bold values indicate a significant slope that supports the corresponding hypothesis. For instance, H2 predicted that caution becomes more important as attribute predictiveness *decreases* (a negative slope). Thus, Table 7 shows a minus sign for the expected slope and all significant, negative slopes are shown in bold. Where possible, we show separate results for the out-of-sample, sparse in-sample, and dense in-sample tasks (using *lp* = 0%, 10%, and 50%). However, to simplify the table the real-world data results for H2 are shown only with *lp*=10%; Section 7.4 describes other results.

The tables show strong support for hypotheses H1, H2, and H4. In particular, we accept H1, H2, and H4 because the pooled analyses find significant slopes in the expected direction; non-pooled results also demonstrate consistent support. Thus, the data support the claims that each cautious inference algorithm outperforms[7] its non-cautious variant by increasing amounts when autocorrelation

7. Technically, the slope results don't by themselves show that the cautious algorithms "outperform" the non-cautious algorithms—only that the relative performance of the cautious algorithms is improving in the hypothesized direction.

	Expected slope	Syn. data lp=0%	Syn. data lp=10%	Syn. data lp=50%	Real-world data lp=0%	Real-world data lp=10%	Real-world data lp=50%
H1: auto-correlation							
ICA_C vs. *ICA*	+	**+0.13**	**+0.13**	**+0.03**	—	**+0.27**	**+0.15**
ICA_{Kn} vs. *ICA*	+	n.a.	**+0.04**	**+0.02**	n.a.	**+0.15**	**+0.13**
Gibbs vs. $Gibbs_{NC}$	+	**+0.18**	**+0.15**	**+0.03**	**+0.27**	**+0.25**	**+0.15**
LBP vs. LBP_{NC}	+	**+0.10**	**+0.08**	—	—	—	—
$wvRN_{RL}$ vs. $wvRN_{ICA+NC}$	+	n.a.	**+0.43**	—	n.a.	**+0.41**	**+0.07**
$wvRN_{ICA+C}$ vs. $wvRN_{ICA+NC}$	+	n.a.	**+0.40**	—	n.a.	**+0.67**	**+0.10**
Pooled	+	**+0.13**	**+0.21**	**+0.01**	**+0.13**	**+0.30**	**+0.11**

Table 6: Summary of results for hypothesis H1. All values shown represent a slope that is significantly different from zero; values in bold support H1. For H1, at a given lp value all data sets (except WebKB+co) are used to compute a single slope value by treating the auto-correlation of the data set as the X value. All algorithms used PLUL where applicable. "n.a." indicates that the algorithm doesn't make sense at lp=0%.

	Expected slope	Syn. data lp=0%	Syn. data lp=10%	Syn. data lp=50%	Real-world data ($lp = 10\%$) Cora	CiteSeer	HepTH	WebKB+co	WebKB	Terror
H2: attribute predictiveness										
ICA_C vs. *ICA*	-	**-0.10**	**-0.25**	**-0.12**	**-0.60**	**-0.61**	**-0.29**	—	—	—
ICA_{Kn} vs. *ICA*	-	n.a.	**-0.06**	**-0.08**	**-0.14**	—	**-0.16**	—	—	—
Gibbs vs. $Gibbs_{NC}$	-	**-0.09**	**-0.27**	**-0.14**	**-0.44**	**-0.50**	—	—	—	—
LBP vs. LBP_{NC}	-	**-0.12**	**-0.28**	**-0.05**	**-0.46**	**-0.35**	—	n.c.	**-0.29**	—
Pooled	-	**-0.10**	**-0.22**	**-0.10**	**-0.23** (over all real data and CC algs.)					
H3: link density										
ICA_C vs. *ICA*	-	**-0.08**	**-0.09**	**-0.03**						
ICA_{Kn} vs. *ICA*	-	n.a.	+0.06	**-0.02**						
Gibbs vs. $Gibbs_{NC}$	-	**-0.09**	**-0.07**	**-0.04**	(not evaluated)					
LBP vs. LBP_{NC}	-	+0.12	**-0.23**	**-0.04**						
$wvRN_{RL}$ vs. $wvRN_{ICA+NC}$	-	n.a.	**-0.18**	**-0.05**						
$wvRN_{ICA+C}$ vs. $wvRN_{ICA+NC}$	-	n.a.	0.11	**-0.03**						
Pooled	-	—	**-0.07**	**-0.04**						

Table 7: Summary of results for hypotheses H2 and H3. As before, all values shown represent a slope that is significantly different from zero; values in bold support the corresponding hypothesis. All algorithms used PLUL where applicable. "n.c." indicates where *LBP* did not converge.

is higher (H1), attribute predictiveness is lower (H2), and/or the labeled proportion is lower (H4). In addition, the data show consistent interactions among these factors. In particular, the strength of

However, the raw accuracies do show consistent performance gains for the cautious algorithms, so in this context the slope results do show the cautious algorithms outperforming the others by increasing amounts.

	Expected slope	Syn. data	Real-world data					
			Cora	CiteSeer	HepTH	WebKB+co	WebKB	Terror
H4: labeled proportion (comparing cautious vs. non-cautious algorithm)								
ICA_C vs. *ICA*	-	**-0.09**	**-0.11**	**-0.14**	**-0.05**	—	—-	—
ICA_{Kn} vs. *ICA*	-	**-0.02**	**-0.06**	—	**-0.05**	**-0.29**	—	—
Gibbs vs. $Gibbs_{NC}$	-	**-0.11**	**-0.14**	**-0.13**	0.05	0.28	—	—
LBP vs. LBP_{NC}	-	**-0.05**	—	—	—	n.c.	—	—
$wvRN_{RL}$ vs. $wvRN_{ICA+NC}$	-	**-0.28**	**-0.37**	**-0.39**	**-0.18**	—	—-	—
$wvRN_{ICA+C}$ vs. $wvRN_{ICA+NC}$	-	**-0.27**	**-0.36**	**-0.32**	**-0.15**	**-0.31**	—	+0.28
Pooled	-	**-0.12**	**-0.07** (over all real data and CC algs.)					
H5: labeled proportion (comparing with PLUL vs. without PLUL)								
ICA_C	-	**-0.02**	—	—	—	—	—	—
ICA_{Kn}	-	**-0.01**	—	—	**-0.02**	—	—	—
ICA	-	—	—	—	—	**-0.18**	—	—
Gibbs	-	**-0.02**	—	—	**-0.04**	—	**-0.07**	—
LBP	-	**-0.03**	—	—	—	n.c.	—	—
Pooled	-	**-0.02**	**-0.01** (over all real data and CC algs.)					

Table 8: Summary of results for hypotheses H4 and H5, which both vary the labeled proportion (lp). As before, all values shown represent a slope that is significantly different from zero; values in bold support the corresponding hypothesis. For H4, all algorithms used PLUL where applicable.

the dependence (the magnitude of the slope) generally decreases as the labeled proportion increases from 10% to 50% (Section 7.4 discusses the differences between lp=0% and 10% in more detail).

Table 7 shows weaker support for H3 (cautious inference gain increases as link density decreases). H3 is supported by most of the synthetic data cases and by the pooled analysis for lp=10% and lp=50%, but the magnitude of the slopes indicates a weaker effect. Moreover, Section 7.5 examines these results more closely and proposes that a more appropriate hypothesis would state that the cautious inference gain is greatest when link density is moderate. This conclusion is also tentatively supported by a per-node degree analysis of the real data.

Table 8 also shows weaker support for H5. The synthetic data results supported H5 for every algorithm except *ICA*. In addition, for 18 of the 29 possible cases shown for the real data sets, the computed slope was negative, as predicted by H5. However, the magnitude of these slopes indicate a weaker effect than with H1, H2, or H4. This decreased magnitude, in conjunction with the smaller number of trials for the real data, leads to only 4 of those 18 slopes reaching statistical significance. Nonetheless, by combining trials across algorithms and data sets, the pooled analysis does find significant (but small) negative slopes for both the synthetic and real data, so we accept H5. This indicates, as expected, that cautious learning with PLUL is most important when lp is small; Section 7.7 also demonstrates that in this case PLUL can provide substantial performance gains.

In addition to these results for each hypothesis, regarding relative performance trends as data characteristics vary, our results also show statistically significant differences between the cautious and non-cautious algorithms for at least some of the data conditions. These differences are consistent with the accepted hypotheses. For instance, using the default synthetic data characteristics, each cautious algorithm showed a significant performance gain over its non-cautious variant, and the amount of this gain increased as autocorrelation increased, attribute predictiveness decreased, or labeled proportion decreased.

7.2 Explanation of Results Presentation

In the following sections, we present several figures that compare CC algorithmic performance. In these figures some controllable parameter is the x-axis and the y-axis is the resultant accuracy for a given algorithmic variant, averaged over all trials. For instance, Figure 8 plots accuracy vs. the degree of homophily (*dh*). Each figure compares cautious and non-cautious variants of a particular CC algorithm: *ICA*, *Gibbs*, *LBP*, or *wvRN*. In addition, for the CC algorithms that use a local classifier (*ICA* and *Gibbs*), we often include results for the non-relational algorithm *CO* for comparison.

In each section below, we use these results to describe two kinds of analysis. First, we accept or reject a hypothesis, based on the pooled regression slope test. This analysis confirms or fails to confirm that the importance of the cautious techniques *does change* in the expected direction as some data parameter varies, but does not evaluate *how important* the cautious techniques are in improving performance. To answer the latter question, we report on a second analysis that evaluates, using paired t-tests, whether the cautious techniques perform significantly better than the non-cautious alternatives (see Section 6.9).

Each figure has embedded statistical information corresponding to some of these t-tests. In particular, each non-cautious CC variant is plotted with a $\times$ marker, while cautious CC variants are plotted with a triangle (where multiple cautious variants exist, two triangle orientations are used: $\bigtriangledown$ and $\bigtriangleup$). For a particular x-value, if the plotted triangle is filled in (solid color), then that cautious variant had accuracy that was significantly different from the accuracy *of the corresponding non-cautious variant*. Hollow triangles instead indicate no significant difference. This notation does *not* directly indicate other significance comparisons (e.g., between the two cautious variants ICA_C and ICA_{Kn}); where necessary we describe such results in the text. For example, in Figure 8, the graph in the third column of the first row (*LBP* at *lp*=0%) shows that *LBP* significantly outperforms LBP_{NC} when *dh*=0.6 (note the filled triangle). However, for *dh*=0.5, *LBP*'s small gain is not statistically significant (hollow triangle).

When *lp*=0%, ICA_{Kn} is equivalent to *ICA*, so results for ICA_{Kn} are not shown. Also, *LBP* with WebKB+co did not converge due to the very high number of links, so results for that case are not considered (cf., Taskar et al., 2002).

7.3 Result 1: The Relative Gain of Cautious Inference Increases with Increasing Autocorrelation

Table 6 reports that for H1, for the sparse in-sample task (*lp*=10%), the pooled regression analyses found all significant positive values for the slope *B*. Thus, we accept H1. In addition, all the non-pooled analyses found significant positive values. The only exception was *LBP* on the real data sets, which had a positive, non-significant slope ($b = +0.03$).

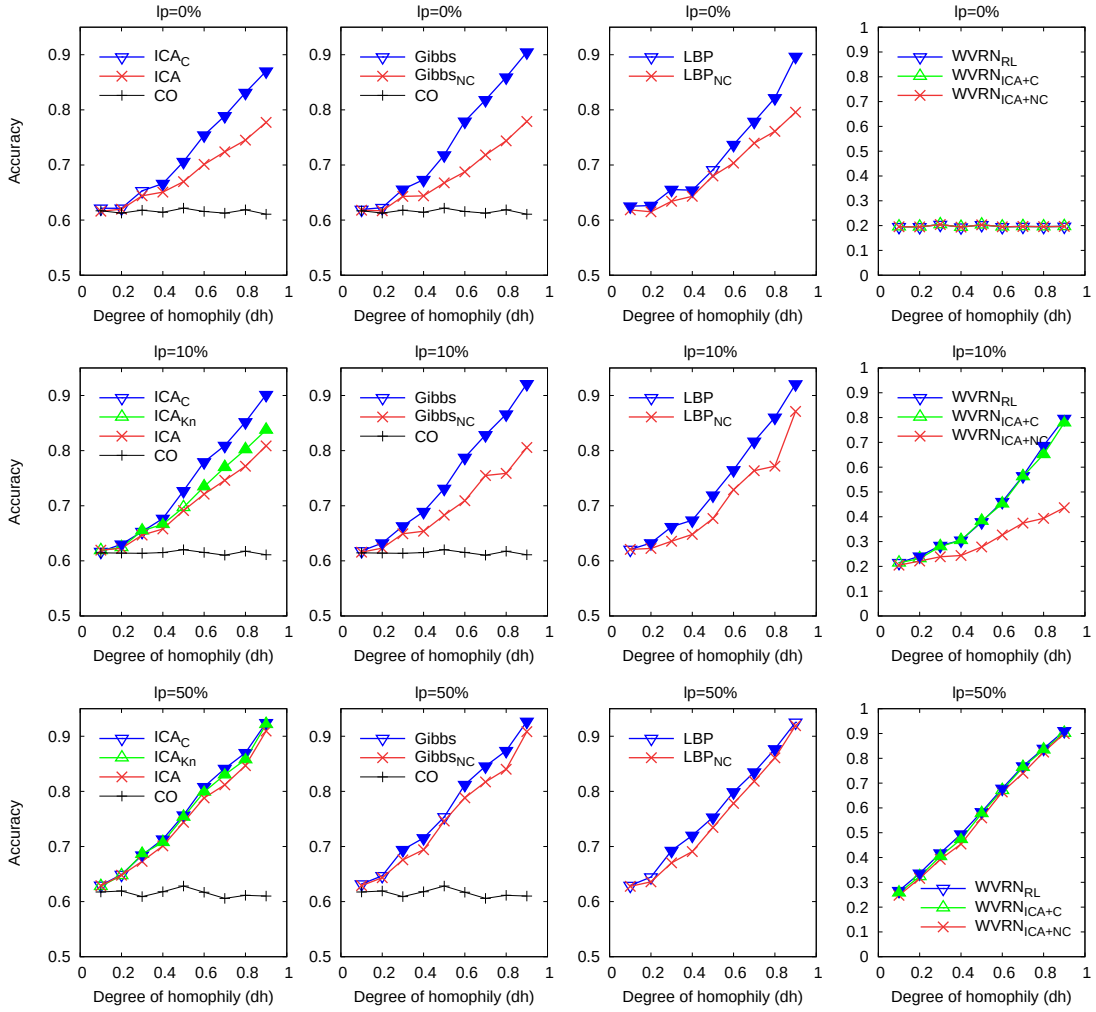

Figure 8: Results for the synthetic data as the degree of homophily (*dh*) varies. Section 7.2 explains how filled triangles indicate statistical significance. Some of the gains are small but consistent, leading to significance, as in the bottom right graph.

For *lp*=0% and *lp*=50%, the pooled analyses and most individual analyses show the same positive slopes (on the real data for ICA_C vs. *ICA* at *lp*=0%, the slope was $b = +0.11$, but the p-value was just over the significance threshold), as we also found with LR and kNN. The reduced significance and magnitude of the slopes when *lp*=50% is also consistent with our expectations, since the overall importance of caution should decrease as *lp* increases (see hypotheses H4 and H5). Section 7.4 explains more for the *lp*=0% case.

Figure 8 shows detailed performance trends for the synthetic data. Here each column presents results for different variants of a single CC algorithm (*ICA*, *Gibbs*, *LBP*, and *wvRN*), and each row shows results for a different value of *lp*. The x-axis varies homophily (which directly increases autocorrelation) and the y-axis reports average accuracy.

This figure confirms that when homophily is very low, CC offers little gain, and thus the cautious variants perform equivalently to the non-cautious variants (and, except for *wvRN*, to the non-

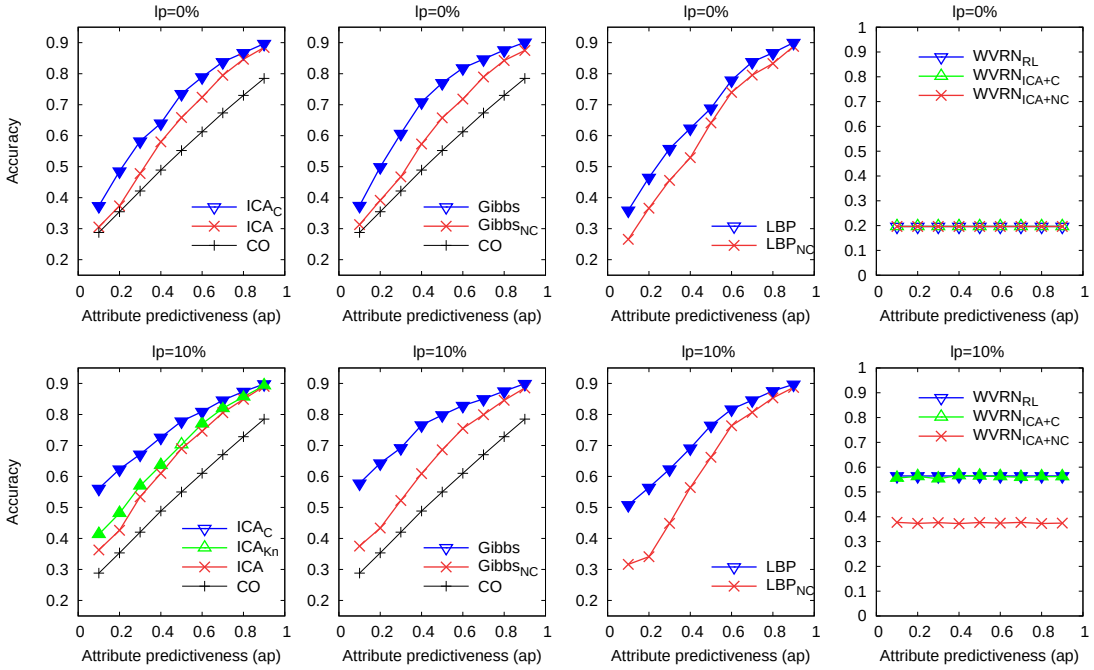

Figure 9: Results for the synthetic data as attribute predictiveness (*ap*) varies.

relational baseline *CO*). As the strength of relational influence (as well as the potential for incorrect relational inference) increases with higher homophily, the relative gain of the cautious methods increases substantially (e.g., at *lp*=10%, gains for NB-based algorithms rise from 4-5% at *dh*=0.5 to 9-12% at *dh*=0.9). The gains from caution are statistically significant in most cases when $dh \geq 0.3$. Results with LR and kNN show very similar trends (see online appendix).

Figure 8 also confirms that as *lp* increases, the cautious and non-cautious variants perform more similarly. However, even for *lp*=50%, the cautious variants maintain a significant, though smaller, advantage. In the other results discussed below, the same trend of very similar performances at *lp*=50% was evident. Likewise, the graphs for *lp*=0% are similar to those for *lp*=10%. Thus, we defer most results for *lp*=0% or 50% to the online appendix.

7.4 Result 2: The Relative Gain of Cautious Inference Increases as Attribute Predictiveness (*ap*) Decreases

Table 7 reports that, for *lp*=10%, the regression analyses found all significant negative slopes (as expected) for the synthetic data. Likewise, in almost all cases we found significant negative slopes for the real data sets that have substantial autocorrelation (Cora, Citeseer, HepTH, and WebKB+co), except for WebKB+co (which had very erratic performance with all the algorithms). We accept H2, because the pooled analysis found negative slopes at *lp*=10% for both synthetic and real data; this result also holds at *lp*=0% and 50%.

Figure 9 shows detailed performance trends for the synthetic data as the x-axis varies *ap*. For instance, for *lp*=10%, when the attribute predictiveness (*ap*) is 0.6 (the default), ICA_C and *Gibbs* outperform their non-cautious variants by 6-7%. However, as *ap* decreases to 0.2, label uncertainty

increases (as evidenced by the drop for *CO*), causing the relative gain of caution to increase to 20%. *LBP* shows very similar results.

Results for *lp*=0% are mostly similar, but with an interesting twist. In this case, the relative gain of cautious CC increases as *ap* decreases, as with *lp*=10%. However, this gain peaks at *ap*=0.2 or 0.3, then declines as *ap* continues to decrease. When attribute predictiveness is very low, and there are no known labels to help seed the inference (i.e., *lp*=0%), then even the cautious algorithms have difficulty exploiting relational information, and achieve accuracy only moderately above the baseline *CO*. However, even in this case the cautious algorithms maintain some small, statistically significant advantage over the their non-cautious variants (which at *ap*=0.1 do little better than *CO*). Also, observe that ICA_C, *Gibbs*, and *LBP* all improve substantially for the *lp*=10% case (compared to *lp*=0%), even though only ICA_C explicitly favors the provided known labels in its inference process. In this case, using caution appears to be the important performance factor, regardless of what specific behavior provides that caution.

Figures 10 and 11 provide similar results for the real data sets with *lp*=10%, where the x-axis is now the number of attributes used, which correlates with overall attribute predictiveness. In general, the trends shown are similar to those already observed for the synthetic data. In particular, the graphs for Cora, Citeseer, HepTH, and WebKB all follow the same pattern: cautious algorithms outperform non-cautious algorithms more when the number of attributes is low (and cautious ICA_C outperforms the somewhat cautious ICA_{Kn}). Consistent with H1, the magnitude of these gains varies with autocorrelation: larger for Cora and Citeseer, smaller for HepTH and WebKB, and non-existent for Terror (where autocorrelation is very weak).

There are two exceptions to the similarities of these results with the synthetic data. First, for some data sets *Gibbs* and/or *LBP* perform noticeably worse than ICA_C; we discuss this separately in Section 8.1. Second, WebKB+co shows fairly erratic performance for all algorithms except ICA_{Kn}. In general, the co-citation links used by WebKB+co appear to be very informative (peak accuracy is much higher than with WebKB), but also potentially misleading. This may be a function of the WebKB graph structure: Table 5 shows that co-citation links have a very high label consistency of 0.90 (implying that classifiers will learn a strong relational dependence), but this may be biased by the presence of some very high degree nodes. During learning the co-citation links may appear very informative on average, but this strong dependency may lead to mispredictions for low-degree nodes, leading to the observed erratic behavior.

We now briefly return to the slope analysis of Table 7. For the synthetic data, the negative slopes for H2 are significant in all cases, but generally largest for *lp*=10%. This behavior is consistent with our previously discussed analyses of the synthetic data: when *lp*=0%, the performance of cautious algorithms for very low *ap* is diminished, thus producing a smaller slope magnitude than when *lp*=10%. On the other hand, the more general observation that caution is less useful when *lp* is high explains why the magnitude of the slopes is less for *lp*=50% than for *lp*=10%. We found similar trends for the real-world data sets: while Table 7 shows significant negative slopes for H2 for most cases (excluding the erratic WebKB+co and the low autocorrelation Terror) when *lp*=10%, results (not shown) with *lp*=0% or 50% indicate slopes of reduced magnitude and/or slopes that do not reach statistical significance. However, in both cases the pooled analysis still indicates significant negative slopes for H2 (-0.05 for *lp*=0% and -0.13 for *lp*=50%).

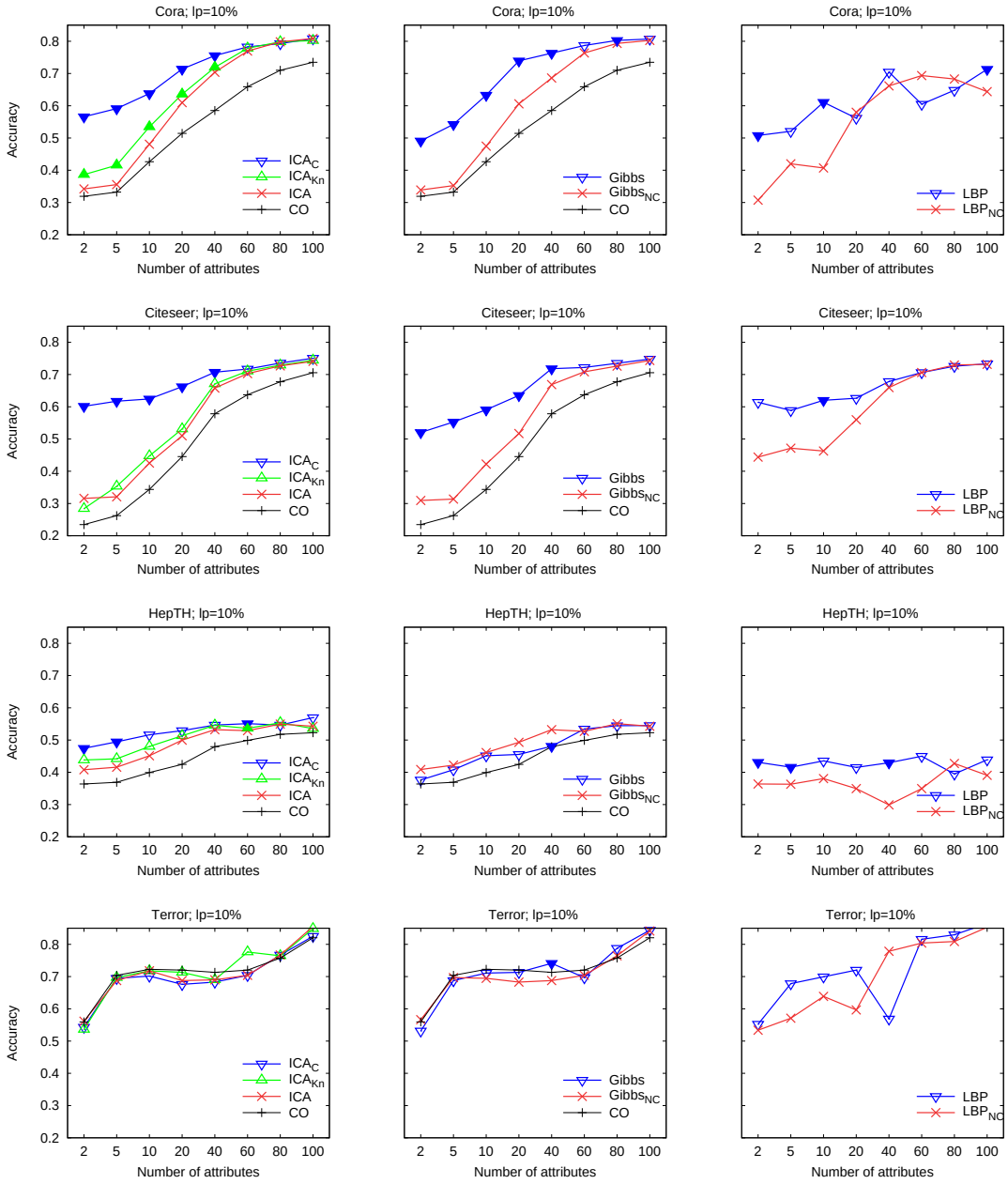

Figure 10: Results for four of the real data sets as the number of attributes is varied. The x-axis is not to scale; this is to improve readability and to yield a more linear curve for the baseline *CO* algorithm, thus facilitating comparison with Figure 9. Because there are only 3-5 trials for the real data, high variance sometimes causes substantial gains to not be statistically significant.

7.5 Result 3: The More Cautious Algorithms Outperform Non-Cautious Algorithms when Link Density (*ld*) is Moderate, But Have Mixed Results When *ld* is High

For the synthetic data, the results in Table 7 support H3 for all algorithms when *lp*=50%, for most algorithms when *lp*=10%, and for only two algorithms when *lp*=0%. The pooled analysis finds,

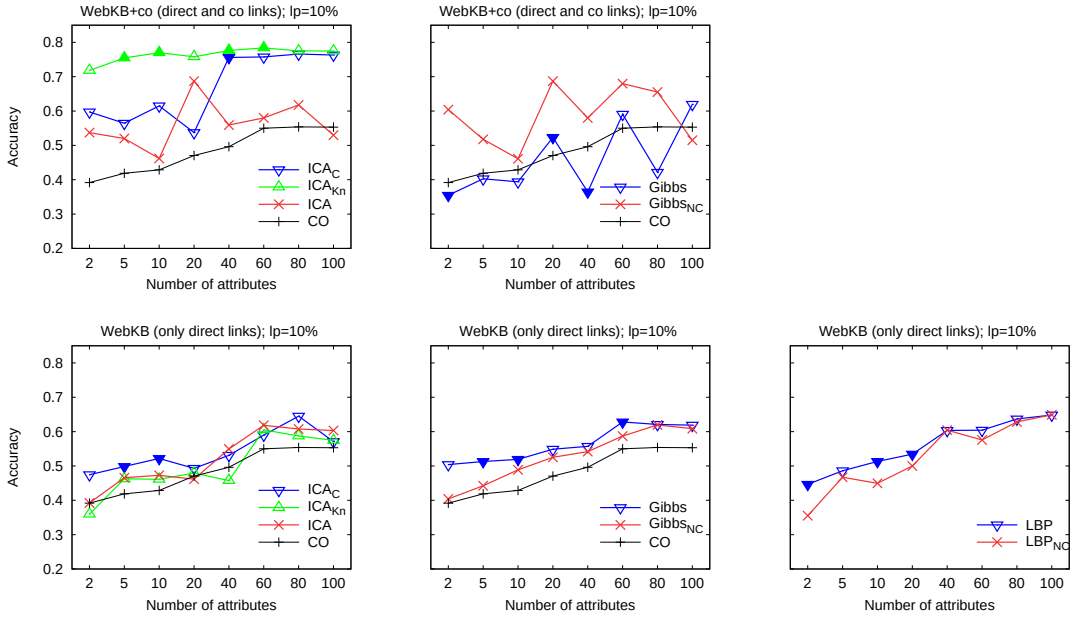

Figure 11: Results for the WebKB data sets as the number of attributes is varied. With WebKB+co, *LBP* did not converge, so results are not shown.

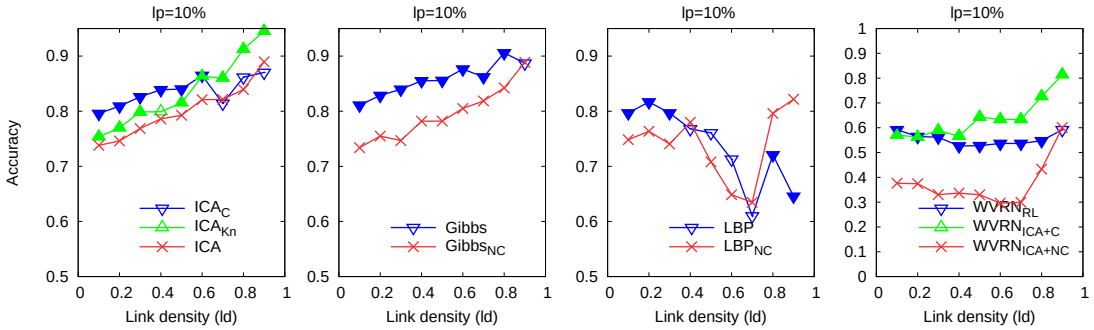

Figure 12: Results for the synthetic data as link density (*ld*) varies.

as expected, significant negative slopes for *lp*=10% and *lp*=50%. However, without corresponding pooled results for the real data, we cannot accept H3. Moreover, the results we present below will suggest a revision to H3.

Figure 12 shows the results as *ld* is varied, for *lp*=10%. When *ld* is low to moderate (up to *ld*=0.6), the cautious algorithms consistently and significantly outperform their non-cautious variants. We had hypothesized that this advantage would decrease as link density increased, because when the link graph is dense, the relational features are relatively unaffected by a few incorrect labels, and thus using such labels cautiously matter less; Figure 12 generally reflects this trend. In some cases the non-cautious algorithm even outperforms the cautious algorithm at very high *ld*. For instance, at *ld*=0.9 *ICA* outperforms the more cautious ICA_C (though not significantly). At such high link density, simply using all available information with *ICA* may work better than ICA_C's

cautious but partial use of estimated labels—provided that accuracy is high enough that errors are few. In separate experiments we confirmed that if the attribute predictiveness (and thus accuracy) was lower, ICA_C maintained it's advantage over *ICA* even when *ld* was very high.

While these results generally indicate, as expected, that the gain from caution decreases as *ld* becomes high, closer examination indicates that this gain from caution peaks not at very low *ld*, but at moderate *ld*. In particular, the gain from caution peaks when *ld* is 0.2 or 0.3 for ICA_C, ICA_{Kn}, or *Gibbs*, and when *ld* is 0.6 for $wvRN_{RL}$ and $wvRN_{ICA+C}$. In hindsight, this effect makes sense: as the number of links decrease, there is less relational influence, and thus less probability of incorrect relational influence, so caution matters less. Another effect is that with fewer links, there are fewer opportunities for a cautious algorithm to favor one node's predictions over another's.

To further analyze these trends, we turn to the real data. We did not attempt to directly vary the link density of the real data sets, because it's not clear how to realistically add links to an existing data set, as would be necessary to create a reasonable range of link densities for experimentation. However, Table 9 examines our previous results for the real data sets, showing the amount of cautious gain broken down by the link degree of each node. This approach does not directly correlate to varying the overall link density, so our conclusions are tentative, but it does provide some insight. We focus primarily on ICA_C; trends with other algorithms were similar.

The results support our previous conjectures. In particular, the cautious gain generally decreases for the highest link degrees, even going negative in some cases. Moreover, in most cases the cautious gain also decreases for the lowest link degrees, resulting in a peak for the cautious gain (shown in bold if present) at moderate link degrees. These effects generally hold true for the synthetic data and for the real data sets that have substantial autocorrelation.

We now return to Figure 12 to consider a few possible exceptions. First, with *LBP*, accuracy decreases with increasing *ld*, is erratic, and is sometimes better with LBP_{NC} than with *LBP*. This is not surprising: the short graph cycles caused by high *ld* produces great problems for *LBP* (e.g., Sen and Getoor, 2006; Sen et al., 2008). Even these *LBP* accuracies are much better than those achieved without PLUL (see Section 7.7).

Second, two of the cautious algorithms (ICA_{Kn} and $wvRN_{ICA+C}$) performed unexpectedly well, continuing to significantly outperform the non-cautious variants (and even alternative cautious variants) at very high link density. Interestingly, these effects also occur with WebKB+co (see Figures 11 and 14), which has by far the highest link density of the real data sets.[8] In addition, the superior performance of ICA_{Kn} at high *ld* remains even when the local classifier is changed to LR or kNN (see Figure 19 in the online appendix). We suspect that ICA_{Kn}'s advantage arises because it both achieves a better starting point than *ICA* (by favoring known labels in its first iteration) and exploits more information than ICA_C (by using all estimated labels in subsequent iterations—and when *ld* is high using a few erroneous labels doesn't harm performance). For $wvRN_{ICA+C}$, its advantage over $wvRN_{RL}$ must arise from the key algorithmic difference: since $wvRN_{ICA+C}$ is a hard-labeling algorithm, it gives all labeled nodes equal weight in the neighborhood average that determines the next label for a node. When link density is high, relying on this simple average may be better than $wvRN_{RL}$'s soft-labeling estimation, which implicitly gives more weight to nodes with more extreme

8. At first, these strong performances seem to conflict with Macskassy and Provost (2007), who generally find $wvRN_{RL}$ outperforms $wvRN_{ICA+C}$. However, two-thirds of their data sets are variants of WebKB, but where all "Other" pages have been removed from the classification task. This change makes the classification problem easier, and thus may explain the discrepancy. In addition, on the only other data set used in that work and this article (Cora), our performance trends are very similar.

	Degree 1-2	Degree 3-5	Degree 6-10	Degree 11-20
Synthetic data, using NB+ICA_C				
lp= 0%	5.5%	8.2 %	**13.8%**	8.2%
lp=10%	5.2%	9.7 %	8.6%	10.6%
lp=50%	2.2%	4.6 %	**8.9%**	7.3%
Average	4.3%	7.5 %	**10.4%**	8.7%
Synthetic data, using NB+*Gibbs*				
lp= 0%	8.5%	13.6 %	**18.6%**	15.4%
lp=10%	6.3%	9.7%	**12.7%**	10.6%
lp=50%	2.5%	3.6%	**7.4%**	5.3%
Average	5.7%	9.0%	**12.9%**	10.5%
Real data with substantial autocorrelation, using NB+ICA_C				
Cora	7.9%	**10.9%**	10.5%	-4.8%
Citeseer	15.8%	**20.5%**	14.6%	-8.3%
WebKB+co	8.3%	9.8%	**12.8%**	10.0%
HepTH	1.2%	-4.3%	3.3%	4.0%
Average	8.3%	9.2%	**10.3%**	0.2%
Other real data sets, using NB+ICA_C				
WebKB	5.8%	1.5%	-4.8%	-6.0%
Terror	2.4%	-5.7%	0.0%	0.0%

Table 9: Per-node degree results showing the amount of gain from caution (ICA_C vs. *ICA* or *Gibbs* vs. $Gibbs_{NC}$). Each value indicates the average accuracy gain from caution for all nodes in the test set within the given link degree range (nodes with degree greater than 20 were rare, and ignored for simplicity). Within each row, a value is in bold if it represents a clear peak, with monotonically decreasing accuracies to both the left and right of that value. The synthetic data used the default settings. The real data sets used the default number of attributes and *lp*=10%.

estimated distributions. In both cases, however, extending *ld* to even more extreme values (e.g., *ld*=0.95) does confirm the overall trend of the amount of cautious gain decreasing at high *ld*.

As expected, we found that these performance differences disappeared when many known labels were provided. In particular, at high link density and *lp*=50%, there were only small differences between ICA_C, ICA_{Kn}, and *ICA*, or between $wvRN_{RL}$, $wvRN_{ICA+C}$, and $wvRN_{ICA+NC}$. In addition, when PLUL was used, even *LBP* and LBP_{NC} performed on par with ICA_C and *Gibbs* when *lp*=50%, despite the challenges of *LBP* with high *ld*.

Overall, our results suggest that a more appropriate rendering of H3 should indicate that *the relative gain from caution will peak at some moderate value of* ld, with the precise value depending on the CC algorithm and the other data conditions. We leave confirmation of this revised hypothesis to future work.

7.6 Result 4: The Relative Gain of Cautious Inference Increases as the Labeled Proportion (*lp*) Decreases

Table 8 reports that, as *lp* varies, the regression analyses found all significant negative slopes (as expected) for the synthetic data. Likewise, in almost all cases we found significant negative slopes

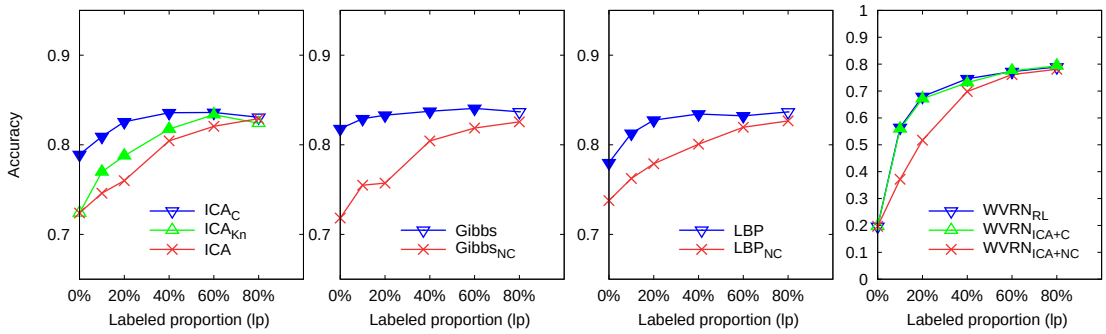

Figure 13: Results for the synthetic data as the labeled proportion (*lp*) varies.

for the real data sets with substantial autocorrelation (all except Terror and WebKB). We accept H4, because the pooled analyses find all negative, significant slopes.

For the real data, the exceptions to H4's stated trend were primarily WebKB+co, which had very erratic performance with all the algorithms, and WebKB, where none of the slopes attained statistical significance. In addition, *LBP* had highly variable behavior so that only for Citeseer did the slope approach statistical significance ($p = .053$, just over the threshold).

Figure 13, for the synthetic data, shows the performance of the cautious and non-cautious algorithms converging as *lp* increases. The cautious algorithms maintain a significant advantage until *lp*=80%. Observe that ICA_{Kn}'s curve lies between that of the more cautious ICA_C and the non-cautious *ICA*, while $wvRN_{RL}$ and $wvRN_{ICA+C}$ obtain the same results with their two different approaches to caution.

Figure 14 shows results for the real data sets as *lp* is varied. This figure show results only for *wvRN*, since results were previously presented for the other algorithms for varying numbers of attributes, and the *lp* graphs don't add additional insight for those algorithms.

The results in Figure 14 mirror those of the synthetic data, with a few exceptions. First, $wvRN_{ICA+C}$ does poorly on Terror, perhaps because of the low autocorrelation. Second, with WebKB+co, $wvRN_{ICA+C}$ outperforms $wvRN_{RL}$ when *lp* is low, though the gains are not quite significant; this effect was discussed in Section 7.5. Finally, the accuracy of *wvRN* for WebKB goes *down* with increasing *lp*. WebKB with just direct links has some autocorrelation but very low label consistency (see Table 5), because each node tends to link in certain patterns to nodes with a *different* label from itself (cf., Macskassy and Provost, 2007). Algorithms based on *wvRN* assume homophily, not such more complex forms of autocorrelation. Consequently, increasing *lp* only serves to reduce accuracy below the majority class baseline. Running *wvRN* with only co-citation links, as done for WebKB+co, works much better.

7.7 Result 5: The Relative Gain of Cautious Learning With PLUL Increases as the Labeled Proportion (*lp*) Decreases

The previous results compared cautious vs. non-cautious variants of a particular CC algorithm, in all cases using PLUL. We now justify the use of PLUL and examine its impact.

The bottom of Table 8 shows the regression slope results for H5, where the x-axis varies the labeled proportion (*lp*), and each table row compares a single CC algorithmic variant when using PLUL vs. not using PLUL. As expected, the slope analysis found all significant negative slopes for

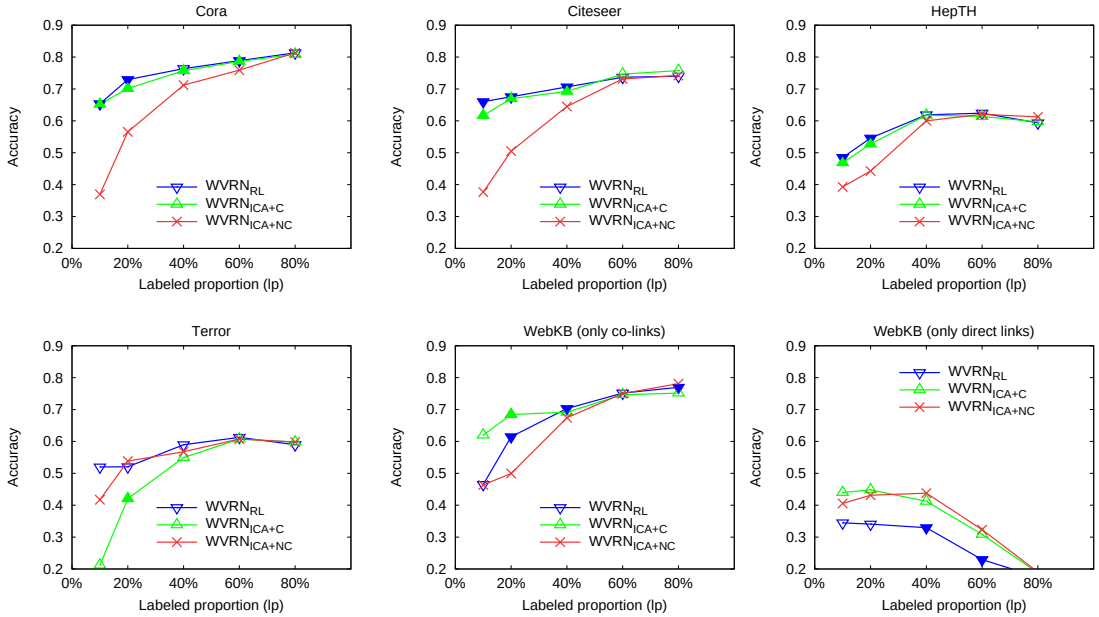

Figure 14: Results for variants of *wvRN* on the real data sets, as *lp* is varied. For the first WebKB results, *wvRN* uses *only* co-citation links (unlike previous results with other algorithms, where WebKB+co used direct links and co-links together; see Section 6.6.3). Recall that filled triangles indicate statistical significance, but only for comparing the cautious variant (here, $wvRN_{RL}$ or $wvRN_{ICA+C}$) vs. the non-cautious variant ($wvRN_{ICA+NC}$).

the synthetic data (with one exception where the p-value was close to the threshold), although the magnitude of the slopes suggests a weak trend. For the real data sets, while 18 of the 29 possible slopes were in the expected direction, only 4 of these slopes were statistically significant (recall that the real data sets have available only 3-5 trials, making significance harder to achieve). However, pooling the results across the data sets and algorithms yields a significant negative slope for both the synthetic and real data, so we accept H5.

Thus, while the effect (as *lp* varies) is smaller than with previous hypotheses, H5 indicates the PLUL provides the most gain when *lp* is small. To measure the magnitude of this gain, Table 10 shows the impact of PLUL when *lp*=0%. Each row shows the results for a different collective algorithm. Results are given for each algorithm both with and without PLUL, along with the overall gain from PLUL. Because PLUL interacts closely with the local classifier, we show results here for NB, LR, and kNN for the CC algorithms that use a local classifier. *CO* and *wvRN* are unaffected by PLUL, and thus are not shown.

In general, we found that PLUL improved performance, sometimes substantially, but the data regions where such substantial gains occur vary by classifier and/or CC algorithm. For instance, Column A of Table 10 shows results for the default synthetic data settings. Here, PLUL improves performance for almost all algorithms. In particular, the gains range from -0.3% to 10.8%, with an average of 4.0%, and are significant in 9 of the 14 cases. Column B shows results where the attribute predictiveness is 0.3 (instead of the default 0.6). In this case, the gains due to PLUL are almost

	A.) Default settings			B.) Low attr. predictiveness			C.) High link density		
	With PLUL?		PLUL	With PLUL?		PLUL	With PLUL?		PLUL
	Yes	No	Gain	Yes	No	Gain	Yes	No	Gain
Using the NB local classifier									
ICA_C	78.9	77.8	1.1	58.2	52.5	**5.7**	80.6	72.0	**8.6**
ICA	72.3	72.6	-0.3	47.7	46.8	0.9	77.0	75.4	1.6
Gibbs	81.8	81.5	0.3	60.6	55.9	**4.7**	80.8	79.2	1.6
$Gibbs_{NC}$	71.8	71.1	0.7	46.7	46.0	0.7	76.4	74.2	**2.2**
Using the LR local classifier									
ICA_C	78.6	74.1	**4.5**	56.8	43.2	**13.6**	82.9	73.9	**9.0**
ICA	70.8	68.5	**2.3**	48.5	44.4	**4.1**	70.8	72.5	-1.7
Gibbs	76.5	72.9	**3.6**	52.3	50.8	1.5	77.6	77.8	-0.2
$Gibbs_{NC}$	70.3	65.3	**5.0**	48.4	43.2	**5.2**	71.4	71.3	0.1
Using the kNN local classifier									
ICA_C	74.1	69.0	**5.1**	51.4	39.2	**12.2**	78.5	65.4	**13.1**
ICA	71.7	64.2	**7.5**	48.4	41.0	**7.4**	75.2	74.7	0.5
Gibbs	73.9	70.0	**3.9**	54.4	48.1	**6.3**	80.3	79.7	0.6
$Gibbs_{NC}$	71.7	61.3	**10.4**	47.7	38.9	**8.8**	75.0	74.0	1.0
Using *LBP*									
LBP	77.8	76.4	1.4	55.7	27.9	**27.8**	69.7	21.5	**48.2**
LBP_{NC}	73.9	63.1	**10.8**	45.5	24.4	**21.1**	54.3	31.2	**23.1**

Table 10: Impact of PLUL on accuracy with the synthetic data, for CC algorithms where PLUL applies, at *lp*=0%. Gains in bold are statisticaly significant.

all larger, ranging from 0.7% to 27.8% (average of 8.6%), and are significant in 11 of 14 cases. These results are consistent with H2: when attributes are less predictive of the class label, cautious techniques, including PLUL, become more important. Finally, column C shows results where the link density is now 0.7 (instead of the default 0.2); here the gains due to PLUL are more varied. For ICA_C, PLUL remains important and matters even more than with the default data settings. We conjecture that this is because with so many links, relational influence can spread very quickly in the graph, and thus the PLUL process is very important to ensuring that ICA_C's confidence measure selects the most reliable predictions during the first few iterations. Indeed, when *lp* is instead set to 10% (thus providing more certain estimates for the early iterations), PLUL became much less important for ICA_C. *LBP* has known issues with high link density, but PLUL helps substantially to ameliorate them. For the other algorithms, the increased link density leads to PLUL having a minor impact, consistent with H3.

Table 11 shows similar results for the real data sets, where results for all six data sets have been pooled together. Since we cannot directly vary link density, we instead show results with two conditions. On the left is the "fewer attributes" case; here each data set uses its default number of attributes, as explained in Section 6.9. On the right is the case where each data set uses 100 attributes.

Compared to results with the synthetic data, Table 11 shows less evidence for the effectiveness of PLUL with the real data sets. While all algorithms show a gain from using PLUL, only about half of the gains are statistically significant. To explain, consider that PLUL works best when the

	Fewer attributes(default)			More attributes(100)		
	With PLUL?		PLUL	With PLUL?		PLUL
	Yes	No	Gain	Yes	No	Gain
ICA_C	56.8	56.1	0.7	68.6	68.1	0.5
ICA	54.5	52.3	2.2	65.7	64.9	0.8
Gibbs	53.5	50.1	**3.4**	67.0	66.1	0.9
$Gibbs_{NC}$	55.5	53.0	2.5	66.5	65.6	**0.9**
LBP	49.9	44.3	**5.6**	65.2	58.4	**6.8**
LBP_{NC}	46.0	42.1	3.9	63.5	56.4	**7.1**

Table 11: Accuracy results showing the impact of using PLUL with the real data. Each value shows results pooled over the six real data sets, at *lp*=0%, using NB where applicable. Gains in bold are statistically significant.

holdout set used for learning is most similar to the test set. With the synthetic data, such similarity is likely, because the two graphs are generated from the same distribution. However, with the real data, splitting an arbitrary graph into multiple subgraphs, even while seeking to maintain similar class distributions, may nonetheless produce subgraphs with important differences (e.g., in autocorrelation), leading to sub-optimal parameter choices by PLUL. Future work is needed to explore these issues.

Nonetheless, the evidence suggests that in most cases for the real and synthetic data PLUL improves performance. Moreover, for every algorithm there was some type of data for which not using PLUL led to very poor performance. Thus, applying PLUL in all of our other experiments seemed advisable for maximizing performance and for ensuring the most equitable comparisons.

7.8 Choice of Relational Feature Types

Section 6.6 described how each trial selected a type of relational feature to use. For completeness, Table 12 summarizes how often each type of feature was chosen. In general, the best feature type (as chosen by cross-validation) varied based on the local classifier used and the data conditions. However, Table 12 shows that for NB, multiset features were dominant, especially for the more cautious algorithms (chosen 76-96% of the time for ICA_C and *Gibbs*). With kNN, proportion features were dominant, while with LR count features were chosen most often but proportion features were also fairly common, especially with high *ld*. These results suggest that an analyst should most likely use multiset with NB, use proportion with kNN, and consider the data conditions to select a feature type for LR.

The superiority of multiset features, when they were applicable, is interesting because they are "cautious" features that simply ignore nodes with no known or predicted label (see Section 6.6.1). Likewise, Section 6.5 reported that LR with ICA_C performed best when missing feature values were ignored (by using a separate classifier trained without the missing features). These results are consistent with Saar-Tsechansky and Provost (2007), who found (for non-relational data) this "reduced-feature model" approach to be superior to commonly used approaches based on imputation. For a non-relational setting, their results thus demonstrate the superiority of a more "cautious" approach to handling missing values during testing. For relational domains, we could imagine taking this idea of ignoring missing/estimated values even further, e.g., using a classifier that ignored

	A.) ICA_C			B.) *ICA*			C.) *Gibbs*		
	Mult.	Count	Prop.	Mult.	Count	Prop.	Mult.	Count	Prop.
Synthetic data, using the NB local classifier									
Default	96%	0%	4%	72%	0%	28%	100%	0%	0%
Low *ap*	88%	0%	12%	20%	4%	76%	92%	0%	8%
High *ld*	48%	0%	52%	80%	0%	20%	96%	0%	4%
Average	77%	0%	23%	57%	1%	41%	96%	0%	4%
Synthetic data, using the LR local classifier									
Default	n.a.	92%	8%	n.a.	80%	20%	n.a.	80%	20%
Low *ap*	n.a.	52%	48%	n.a.	60%	40%	n.a.	68%	32%
High *ld*	n.a.	80%	20%	n.a.	52%	48%	n.a.	48%	52%
Average	n.a.	75%	25%	n.a.	64%	36%	n.a.	65%	35%
Synthetic data, using the kNN local classifier									
Default	n.a.	0%	100%	n.a.	0%	100%	n.a.	0%	100%
Low *ap*	n.a.	0%	100%	n.a.	12%	88%	n.a.	0%	100%
High *ld*	n.a.	0%	100%	n.a.	0%	100%	n.a.	0%	100%
Average	n.a.	0%	100%	n.a.	4%	96%	n.a.	0%	100%
Real data, using the NB local classifier									
Cora	97.5%	2.5%	0.0%	70.0%	17.5%	12.5%	100.0%	0.0%	0.0%
Citeseer	92.5%	2.5%	5.0%	57.5%	32.5%	10.0%	100.0%	0.0%	0.0%
WebKB+co	84.4%	0.0%	15.6%	65.6%	34.4%	0.0%	71.9%	12.5%	15.6%
WebKB	53.1%	40.6%	6.3%	31.3%	56.3%	12.5%	75.0%	21.9%	3.1%
HepTH	85.0%	12.5%	2.5%	62.5%	25.0%	12.5%	70.0%	27.5%	2.5%
Terror	50.0%	8.3%	41.7%	50.0%	25.0%	25.0%	41.7%	16.7%	41.7%
Average	77.1%	11.1%	11.8%	56.1%	31.8%	12.1%	76.4%	13.1%	10.5%

Table 12: The relational feature type (multiset, count, or proportion) chosen by cross-validation. For the synthetic data, results are shown with the default settings, with low attribute predictiveness (*ap*=0.3), and with high link density (*ld*=0.7). For the real data, results are shown averaged across all the data points shown in Figures 10 and 11.

the estimated label of a linked node but instead directly used its non-relational features. However, Jensen et al. (2004) demonstrated that such an approach is generally inferior to the approaches we consider in this article (label-based features with collective inference), because of the much larger number of model parameters that must be learned for the former case.

7.9 Variants of *wvRN*

Most prior research involving *wvRN* has used $wvRN_{RL}$, the variant suggested as a relational-only baseline by Macskassy and Provost (2007). However, algorithms based on *wvRN* need not necessarily be relational-only. For instance, Macskassy (2007) described a technique for adding additional links to the graph between nodes that appeared similar based on their attributes. Alternatively, we could imagine, for $wvRN_{RL}$, initializing each node's predicted label probabilities based upon the output of an attribute-only local classifier (instead of using class priors as done in Figure 5). Unfortunately, this idea does not work well for a "soft" algorithm such as $wvRN_{RL}$, because after iterating many times the current state is almost completely determined by the known labels, independent

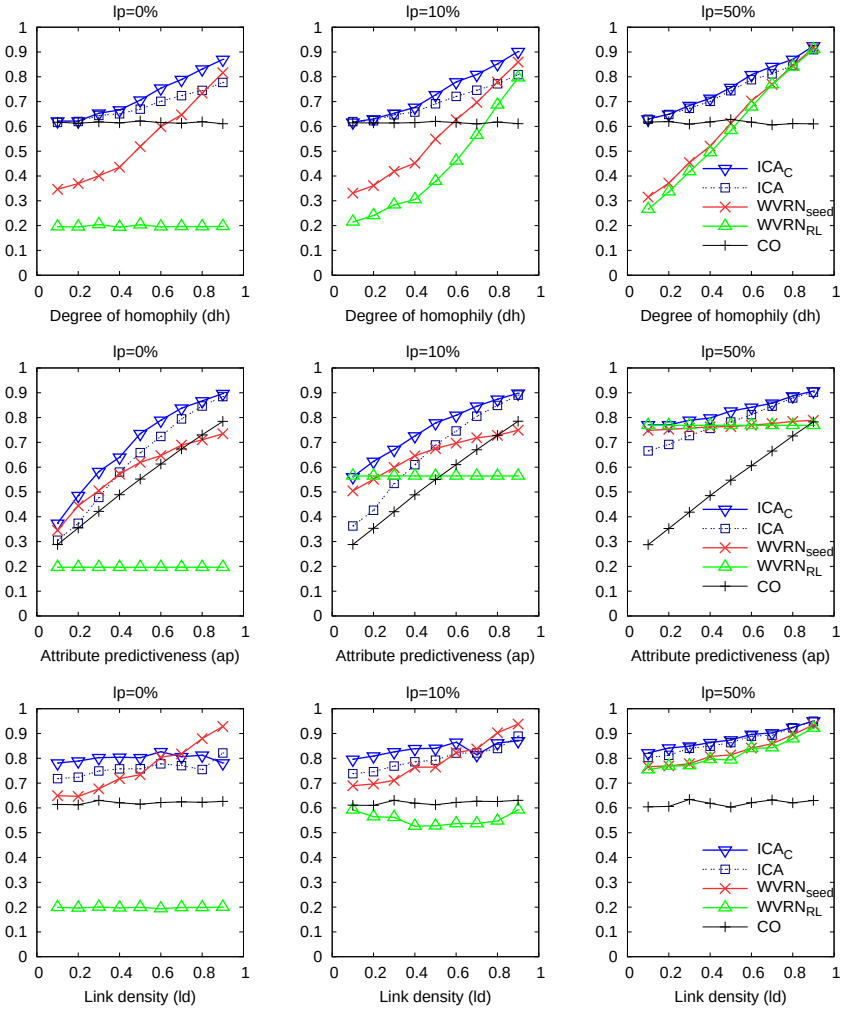

Figure 15: Results for the synthetic data where $wvRN_{seed}$ is added for comparison. Because of the multiple possible comparisons, filled triangles are not used here to indicate statistical significance.

of the starting state (Macskassy and Provost, 2005). While in principle this problem could be addressed via learning an appropriate decay parameter Γ and stopping point, this forfeits much of the simplicity of *wvRN*.

In contrast to $wvRN_{RL}$, with a hard-labeling algorithm such as $wvRN_{ICA+C}$, the initial conditions do matter. In particular, we evaluated $wvRN_{seed}$, an algorithm that behaves just like $wvRN_{ICA+C}$, except that each node's predicted label is initialized to the most likely label predicted by an attribute-only NB classifier. Non-relational information thus "seeds" the inference process but is then not explicitly used again. To the best of our knowledge, this algorithm has not been previously considered for CC.

Figure 15 shows a variety of results for the synthetic data; results with the real data showed similar trends. Overall, $wvRN_{seed}$ outperforms $wvRN_{RL}$ (especially when *lp* is low), which is to be expected since $wvRN_{seed}$ uses more information. $wvRN_{seed}$ generally underperforms ICA_C, which

is also to be expected since ICA_C both uses predicted labels cautiously (while $wvRN_{seed}$ treats all predictions equally) and continues to use both attribute and relational information after the first iteration. The differences with ICA_C are largest when *dh* is low (where *wvRN*'s homophily assumption is violated) or when attribute predictiveness is high (since $wvRN_{seed}$ uses the attributes only at initialization). However, $wvRN_{seed}$ outperforms all of the other shown algorithms when link density is high. This case is analogous to the results with ICA_{Kn} from Section 7.5: if accuracy and link density are high (and homophily is present), then caution with relational information may not be necessary, and this case shows that continuing to use non-relational information after initialization may also not be necessary. Overall, the results indicate that $wvRN_{seed}$ is not likely to be a strong contender as a general purpose CC algorithm, but they do demonstrate an effective way to add non-relational information to *wvRN*-based algorithms.

7.10 Impact of the Default Values for Synthetic Data Generation

The synthetic data evaluated above was generated with the default parameters described in Table 4. Conceivably, our choice of default values could have an important effect on the results. While our evaluation of multiple real data sets has already helped to validate the synthetic data results, we also carried out an extensive exploration with other default values. For instance, when varying *ld*, we experimented with all combinations of *ap*= {0.4, 0.6, 0.8}, *dh*= {0.5, 0.7, 0.9}, and *lp*={0%, 10%, 50%}. For tractability, we only evaluated variants of *ICA*, since the above results show that ICA_C produced the best or nearly the best results for all synthetic and real data sets, and that other cautious algorithms usually behaved like ICA_C.

The trends were highly consistent with the results we report and agree with our accepted hypotheses. For instance, if the default *ap* is very high, the results for varying *dh* showed a much smaller slope for the relative impact of cautious ICA_C vs *ICA*. The only default value that noticeably changed any result was already reported in Section 7.5: when *ap* was small (e.g., 0.4), the unusual advantage of *ICA* over ICA_C observed at very high *ld* disappeared. Thus, we believe the trends in our results are robust over a wide range of data characteristics.

8. Discussion

In this section we compare results with different families of algorithms, examine the overall effectiveness of caution, and use our results to explain the findings of some previous research.

8.1 Comparisons Across Algorithmic Families

Section 7 focused on comparing cautious vs. non-cautious variants within the same algorithmic family. We now briefly compare across these families. We focus on the algorithms that have been most frequently used in previous work: *ICA*, *Gibbs*, *LBP*, and $wvRN_{RL}$. We also include the less studied ICA_C, since our results show that it has very strong performance. We report specific results for *lp*=10%; comparisons were similar for *lp*=0%, while all of the algorithms perform very similarly when *lp*=50%.

$wvRN_{RL}$'s performance depends on homophily, link density, and *lp*. In our study, $wvRN_{RL}$ was thus competitive with the other CC algorithms when homophily and/or *lp* was high, or when the attributes were not very predictive. On the other hand, $wvRN_{RL}$ requires that some labels are known

in the test set, so it is not applicable when *lp*=0% (the out-of-sample task). $wvRN_{seed}$ would be an alternative.

For the synthetic data, the cautious algorithms ICA_C, *Gibbs*, and *LBP* had remarkably similar performance. Among the three, *Gibbs* had a small but sometimes significant performance advantage. For instance, across the results for varying *dh* at *lp*=10% shown in Figure 8, *Gibbs* outperformed ICA_C by an average of 1.0% (significantly for $dh \geq 0.6$) and *LBP* by an average of 0.7% (significantly for $0.4 \leq dh \leq 0.7$). Neither ICA_C nor *LBP* had consistent, significant gains over the other, except that both *Gibbs* and ICA_C had substantial, significant gains over *LBP* when attribute strength was very low (gains of 5-8%) or when link density was high (gains of 14-25%). However, all three algorithms did have substantial, significant gains vs. *ICA*, except for when *dh* was very low or when *ld* was very high. For instance, across the various *dh* levels, *Gibbs* outperformed *ICA* by 0.9-11.2% (all significantly) except for a loss of 0.1% at *dh*=0.1. Thus, based on the synthetic data results, ICA_C, *Gibbs*, and *LBP* usually achieve similar accuracies, despite their use of very different approaches to caution.

On the real data sets, ICA_C, *Gibbs*, and *LBP* likewise performed similarly. However, there are two kinds of differences that should be noted. First, there were a few data sets on which *LBP* and/or *Gibbs* performed noticeably worse than ICA_C. In particular, *Gibbs* has poor performance on HepTH and WebKB+co. In both cases, this is likely due to issues of high link density (WebKB has very many co-citation links; HepTH has fewer links but some nodes have very high degree). High link density can lead to extreme probabilities, where *Gibbs* is known to perform poorly. While this was not a particular problem with the synthetic data (perhaps because the training and test graphs were more similar), NB is well known for producing polarized probabilities in some cases. PLUL does help, for instance, improving performance on HepTH and WebKB+co by an average of 4% and 15%, respectively, in Figures 10 and 11. Nonetheless, performance with *Gibbs* lags that of ICA_C or *ICA*, which are not so influenced by extreme probabilities. We experimented with more and/or longer *Gibbs* chains but this did not improve performance. However, this is one case where the LR classifier performed better than NB: it appears to produce less polarized probabilities than NB, leading to improved performance with *Gibbs* (see Figures 24 and 27 in the online appendix). Similarly, *LBP*, which struggles with high link density, also has problems with HepTH (and likely would have low performance with WebKB+co, had it ever converged) and with Cora. Its difficulty with Cora is surprising and possibly indicates that the conjugate gradient training did not perform adequately, despite our attempts (cf., Sen et al., 2008). However, *LBP* did perform well on Citeseer, which has similar characteristics.

Second, in contrast to the small advantage for *Gibbs* on the synthetic data, for the real data ICA_C holds a small advantage. For instance, in Figure 10, ICA_C outperforms *Gibbs* on average by 1% for Cora and 2.4% for Citeseer, though not significantly. For HepTH and WebKB+co, where *Gibbs* had trouble, the gains averaged 5.4% and 21.0%, respectively, and were significant for HepTH when the number of attributes was small. ICA_C was also robust: it was the only algorithm to outperform *ICA* on average for every real data set considered. Moreover, using results pooled over all six data sets, ICA_C had moderate gains vs. *ICA*, *Gibbs*, *LBP*, and $wvRN_{RL}$, both at the default number of attributes (where the gains were significant) and using 100 attributes for each data set. Comparing to just *Gibbs* and *LBP*, ICA_C had a pooled gain of 4.9% and 7.8%, respectively, with the default number of attributes, and 1.8% and 4.5%, respectively, with 100 attributes.

8.2 Cautious Behavior as a Predictor of Performance

The previous section identified some of the situations in which the algorithms performed similarly or differently. However, if we exclude the extreme data conditions such as very low attribute predictiveness or high link density, a more remarkable finding emerges: *the amount of cautious inference used by an algorithm strongly predicts its relative performance*. This finding is especially interesting because the precise type of cautious inference seems to matter little. On both the synthetic and the real data sets, in most cases ICA_C, *Gibbs*, and *LBP* perform alike, while the non-cautious *ICA*, $Gibbs_{NC}$, and LBP_{NC} also perform similarly to each other (and at lower accuracy levels than the cautious algorithms). However, when many test labels are known (high *lp*), the need for caution decreases, and the differences between these two groups greatly diminish.

This effect can also be seen in other CC variants. For instance, $wvRN_{RL}$ and $wvRN_{ICA+C}$ perform similarly, despite their very different approaches to caution, and they both outperform the non-cautious $wvRN_{ICA+NC}$. Likewise, in almost every case the somewhat-cautious ICA_{Kn} attained an accuracy between that of the more cautions ICA_C and the non-cautious *ICA*.

Thus, the amount of cautious inference seems to be the biggest factor differentiating those algorithms that use attributes, much more so than whether some kind of *ICA* or *Gibbs* or *LBP* is used. Likewise, when attributes are not used, as with the variants of *wvRN*, caution also appears to be the largest factor in predicting relative performance.

8.3 Limitations of Cautious Inference

While our results show that the cautious use of relational information can significantly boost performance, adding more caution to an algorithm is not always beneficial. In particular, the most extreme form of relational caution is to not use any relational information (i.e., *CO*), but that is seldom optimal. Instead, an algorithm must seek to cautiously avoid errors from noisy predictions while still leveraging informative relations.

To illustrate these effects, Figure 16 shows accuracy results for three synthetic data conditions: low attribute predictiveness (*ap*=0.3), the default settings, and high link density (*ld*=0.9). Here the x-axis indicates the algorithm used, with the amount of relational caution used increasing to the right. We focus on variants of *ICA*, but add three new algorithms for further analysis. ICA_{70} is just like ICA_C, except that it stops after it has "committed" and used the most certain 70% of the predicted labels (i.e., after the iteration when $h = 7$ in Figure 2). ICA_{30} and ICA_0 likewise stop after accepting and using 30% and 0% of the predicted labels, respectively. Note that ICA_0 is identical to ICA_{Kn} during the very first iteration (when both use only the "known" labels for relational features), but that ICA_0 stops after that iteration, while ICA_{Kn} continues for 10 more iterations, using all available predictions during those iterations.

For the default and low attribute predictiveness data conditions, the trends are very similar: amongst *ICA*, ICA_{Kn}, and ICA_C, the most cautious ICA_C performs best. Adding more caution to ICA_C, however, consistently decreases performance, as ICA_{70}, ICA_{30}, and ICA_0 use less and less relational information, until the lowest performance is found with the non-relational *CO*. These results make sense: for this data, relational links *are* informative, so completely ignoring any (or all) of them is non-optimal. Indeed, using all of them without any caution (*ICA*) is much better than cautiously ignoring all relations (*CO*), but the cautious algorithm that eventually uses all relations (ICA_C) performs best. Note that this property of (eventually) using all available relational informa-

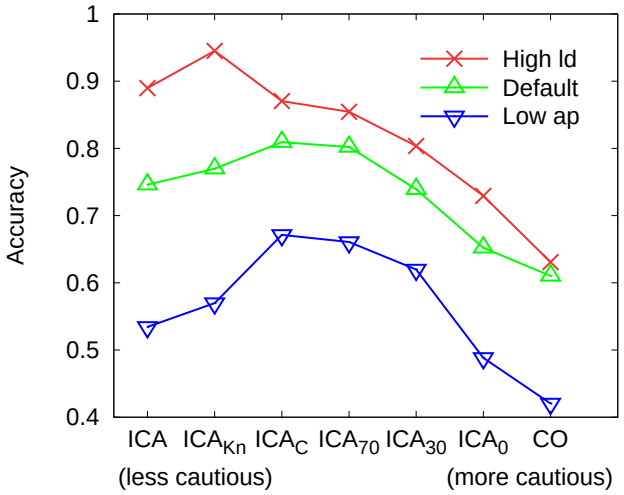

Figure 16: Accuracy as a function of the amount of relational caution used. ICA_{70}, ICA_{30}, and ICA_0 are (even more cautious) variants of ICA_C that stop iterating before some of the less certain relational information has been used.

tion is true of all of the more cautious algorithms that we considered in this article (ICA_C, *Gibbs*, *LBP*, $wvRN_{RL}$, and $wvRN_{ICA+C}$).

The high link density case provides an interesting contrast. Here the general shape of the curve is similar, but the peak performance is observed with ICA_{Kn}, not with the more cautious ICA_C. This effect was already discussed in Section 7.5: if the baseline accuracy is high and there are many links, simply using all available information after the first iteration is best. Similarly, for situations where caution is not very important (e.g., when *lp* is high), the curve would show similar results for *ICA*, ICA_{Kn}, and ICA_C. Thus, in most cases being cautious with relational information is best, but the algorithm should eventually use all available information (relational and non-relational), and in some cases using more caution may be less important or even harmful.

8.4 Explanation of Prior Results

Our investigation enables us to explain the questions from Section 1, among others:

1. **Why did Sen et al. (2008) find no consistent difference between** *Gibbs* **and** *ICA***?** In contrast, *Gibbs* had worked well in other work, and in this article we found that *Gibbs* (and ICA_C) often significantly increases accuracy vs. *ICA*. However, our results and careful study of Sen et al.'s methodology explains the discrepancy: to generate the test set, they used a snowball sampling method that we found produces an effective labeled proportion (*lp*) of at least 0.5—a region where the use of caution has little impact. Also, their study did not vary attribute predictiveness, which we show is a significant factor in the relative performance of more cautious CC algorithms.

2. **Why did McDowell et al. (2007a) find that** ICA_C **significantly outperforms** *Gibbs***, even though attribute predictiveness was high, while here we find that** *Gibbs* **performs on**

par or better than ICA_C in such cases? To investigate, we re-ran our experiments from our earlier paper, but with two variations informed by our now-refined understanding of CC. First, we used PLUL with both the NB and kNN classifiers. Second, we changed the NB classifier to use multiset relational features (instead of proportion), which use more information and which Section 7.8 shows is the feature of choice when using NB (it didn't apply for kNN). With these enhancements, *Gibbs*'s relative performance improved, so that ICA_C and *Gibbs* both significantly outperformed *ICA*, but the results for *Gibbs* and ICA_C did not significantly differ. Thus, more careful learning and representation choices resolves the discrepancy. This also suggests that not using PLUL could potentially have an important effect on performance comparisons. As an additional example, Sen and Getoor (2006) experimented with a wide range of link densities but did not use a technique like PLUL; our results suggest that using PLUL could have significantly improved their results with *LBP* for high *ld*.

3. **Why did Galstyan and Cohen (2007) find that a soft-labeling version of *wvRN* fails to consistently outperform a hard "label propagation" (LP) version?** Most authors have expected that, for relational-only classification, the soft-labeling algorithm that directly reasons with probabilities (thus exercising cautious inference) should outperform a hard-labeling version that only reasons with the single most likely label for each linked node. However, closer examination of their LP algorithm reveals that it includes elements of caution. In particular, after each iteration, LP labels a non-known node only if the estimated score for that node is among the *highest* of any such nodes. Thus, in a way similar to $wvRN_{ICA+C}$, nodes that are closest to known nodes are labeled first, and the algorithm effectively favors label information that was either known or is closer to other known nodes. This cautious behavior enables LP to be competitive with (and sometimes outperform) the soft-labeling algorithm.

4. **Why did Sen et al. (2008) find that *ICA* and *Gibbs* perform better with LR than with NB, while we find the reverse?** We replicated the synthetic data of their paper, and reproduced their results. A key point, however, is that Sen et al. used count relational features for both NB and LR, while we used cross-validation on a holdout set to select the best relational feature type (see Section 6.6). This procedure predominantly selected multiset features for NB (see Section 7.8), which we found in separate experiments to consistently improve NB performance compared to using count features. Consequently, in our results CC algorithms that use NB almost always outperformed those that use LR. While not a focus of our work, such differences can be seen in Table 10. The superior performance of multiset features also confirms the finding of Neville et al. (2003b).

5. **When will cautious algorithms outperform their aggressive variants?** We found that using more cautious CC frequently and sometimes dramatically increased accuracy. In general, cautious CC performs comparatively well whenever relational inference errors are more likely. These errors occur more frequently when there is more uncertainty in the estimated relational feature values (e.g., when the attribute predictiveness is low) or when the effect of any such uncertainty is magnified (e.g., when autocorrelation is high). In some cases, such as when the test set links to many known labels (high *lp*), using a more cautious CC algorithm may be unnecessary. However, in many cases (and with most previous work) *lp* is small or zero, and thus caution may be important.

9. Conclusion

Collective classification's greatest strength—making inferences based on the inferred labels of related nodes—can also be a significant weakness, since this use of uncertain labels may reduce accuracy when the estimates are incorrect. In this article, we demonstrated that managing this estimation uncertainty through "cautious" algorithmic behavior is essential to achieving maximal, robust performance. We showed how varying degrees of cautious inference could be manifested in four different collective inference families, and explained how to use cautious learning with PLUL to further improve performance. Our experimental results with both synthetic and real-world data sets showed that cautious algorithms did outperform their non-cautious variants. By exploring a wide range of data, we identified some data characteristics for which this performance advantage grew larger. In particular, cautious behavior is especially important when there is a higher probability of incorrect relational inference—which occurs when autocorrelation is higher, when link density is moderate, and/or when attribute predictiveness or the labeled proportion is lower. In addition, our study enabled us to answer several important questions from previous work.

Across a wide range of data, we found that an algorithm's degree of caution was a significant predictor of relative performance—in most cases a more important one than the specific collective inference algorithm used. This reinforces the fundamental importance of cautious behavior for CC. However, the cautious CC algorithms were not always comparable. *Gibbs* and (especially) *LBP* sometimes struggled (e.g., when the data had high link density). In contrast, ICA_C was a very reliable performer and almost always had maximal or near-maximal performance, especially for the real-world data. This finding is interesting because this article is the first to consider ICA_C in depth. Moreover, ICA_C is a simple modification to *ICA*, making it much more time-efficient than *Gibbs* or *LBP*. This suggests that ICA_C is a strong contender for general CC tasks, and should be used as a baseline for future CC performance comparisons.

Regarding cautious learning, we found that PLUL generally increased accuracy, sometimes substantially. Parameter tuning is known to be important for learning non-relational classifiers. We show that it can be especially critical for CC due to CC's reliance on uncertain labels during testing. For example, further results showed that for the synthetic data when link density was high, *Gibbs*+NB with a naive α (prior hyperparameter) of 1.0 attained 99% of the accuracy attainable with any α—*if* most test labels were known (e.g., *lp*=80%). However, when *lp*=0% this strategy's accuracy was just 61% of optimal. Using PLUL to set α instead increased accuracy. In addition, our results in Section 7.7 showed PLUL helping both cautious and non-cautious inference algorithms. Thus, using PLUL for cautious learning improves performance, and adding cautious inference helps even more.

Future work is needed to compare the algorithms considered here with alternative methods, such as Markov Logic Networks (Richardson and Domingos, 2006) and the "ghost edge" approach of Gallagher et al. (2008), and to compare PLUL to the alternative "stacked models" discussed in Section 5.5. In addition, further studies to consider the effect of training set size, noise in the known labels, and link uncertainty would be useful. Finally, techniques are needed to further improve the performance of cautious inference on data with high link density or other extreme conditions.

Acknowledgments

Thanks to Doug Downey, Lise Getoor, David Jensen, and Sofus Macskassy for helpful comments on this work, to Prithviraj Sen for the Cora and Citeseer data sets, to Jennifer Neville for helpful discussions and for code that implements LBP, and to Prithviraj Sen and Mustafa Bilgic for clarifications on their work. Thanks also to the anonymous reviewers for many helpful comments that helped to improve this article. Luke McDowell's funding for this research was partly supported by the U.S. Naval Academy Cooperative Program for Scientific Interchange, which is a component of NRL's General Laboratory Scientific Interchange Program. Portions of this analysis were conducted using Proximity, an open-source software environment developed by the Knowledge Discovery Laboratory at the University of Massachusetts Amherst (http://kdl.cs.umass.edu/proximity/). The HepTH data was derived from the Proximity HEP-Th database, which is based on data from the arXiv archive and the Stanford Linear Accelerator Center SPIRES-HEP database provided for the 2003 KDD Cup competition, with additional preparation performed by the Knowledge Discovery Lab.

Appendix A. Measuring the Strength of Relational Dependence

Data sets used for CC are often measured for their autocorrelation. Alternatively, *label consistency* is the percentage of links connecting nodes with the same label. A closely related measure is the *degree of homophily* (*dh*) used by Sen et al. (2008). To see the difference, suppose that a data set has five labels that occur with equal frequency. Sen et al. argue that, if *dh* is zero, the target of a link from a node labeled *A* should be to another node labeled *A* 20% of the time (random chance), not 0% of the time (Sen, 2008). Thus, for a uniform class distribution, the actual probability of a link connecting two nodes i and j of the same label is defined as:

$$\text{label consistency} = P(y_i = y_j | (i,j) \in E) = dh + \frac{1-dh}{|C|}. \quad (4)$$

To facilitate comparison, we adopt this definition to generate synthetic data with varying levels of *dh*. However, for real data sets, we can only directly compute label consistency. Thus, to facilitate comparison we also compute *approximate homophily* from the measured label consistency by assuming a uniform distribution of labels and solving for *dh* using Equation 4.

Appendix B. Information on Additional Results

In Section 7, we omitted some results for alternate local classifiers (LR and kNN) and/or alternate settings of *lp*, since they did not noticeably change our reported trends. These results are available in an online appendix that accompanies this article on the JMLR website.

References

Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 331–338, 2005.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3):259–302, 1986.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society*, 36(2):192–236, 1974.

Mustafa Bilgic and Lise Getoor. Effective label acquisition for collective classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–51, 2008.

Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver Riordan. Directed scale-free graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 132–139, 2003.

Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th Conference of the American Association for Artificial Intelligence (AAAI)*, pages 509–516, 1998.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Roland L. Dobrushin. The description of a random field by means of conditional probabilities and conditions of its regularity. *Theory of Probability and its Applications*, 13(2):197–224, 1968.

Andrew Fast and David Jensen. Why stacked models perform effective collective classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2008.

Karen Yuen Fung and Barbara A. Wrobel. The treatment of missing values in logistic regression. *Biometrical Journal*, 31(1):35–47, 1989.

Brian Gallagher and Tina Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In *Proceedings of the 2nd Workshop on Social Network Mining and Analysis at the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 256–264, 2008.

Aram Galstyan and Paul R. Cohen. Empirical comparison of "hard" and "soft" label propagation for relational classification. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP)*, pages 98–111, 2007.

Tayfun Gurel and Kristian Kersting. On the trade-off betweeen iterative classification and collective classification: first experimental results. In *Working Notes of the 3rd International ECML/PKDD Workshop on Mining Graphs, Trees, and Sequences*, 2005.

David Heckerman. A tutorial on learning with bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

Andreas Heß and Nicholas Kushmerick. Iterative ensemble classification for relational data: A case study of semantic web services. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 156–167, 2004.

Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 259–266, 2002.

David Jensen, Jennifer Neville, and Michael Hay. Avoiding bias when aggregating relational data with degree disparity. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 274–281, 2003.

David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artifical Intelligence*, 97 (1-2):273–324, 1997.

Daphne Koller, Nir Friedman, Lise Getoor, and Benjamin Taskar. Graphical models in a nutshell. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

Zhenzhen Kou and William W. Cohen. Stacked graphical models for efficient inference in Markov Random Fields. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, pages 533–538, 2007.

Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 496–503, 2003a.

Qing Lu and Lise Getoor. Link-based classification using labeled and unlabeled data. In *Proceedings of the Workshop on the Continuum from Labeled to Unlabeled data at the 20th International Conference on Machine Learning (ICML)*, 2003b.

Sofus A. Macskassy. Improving learning in networked data by combining explicit and mined links. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 590–595, 2007.

Sofus A. Macskassy and Foster Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *Proceedings of the International Conference on Intelligence Analysis*, 2005.

Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.

Sofus A. Macskassy and Foster Provost. A brief survey of machine learning methods for classification in networked data and an application to suspicion scoring. In *Proceedings of the Workshop on Statistical Network Analysis at the 23rd International Conference on Machine Learning (ICML)*, 2006.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598, 2000a.

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000b.

Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious inference in collective classification. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 596–601, 2007a.

Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Case-based collective classification. In *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 399–404, 2007b.

Robert J. McEliece, David J. C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16 (2):140–152, 1998.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

Jennifer Neville and David Jensen. A bias/variance decomposition for models using collective inference. *Machine Learning Journal*, 73(1):87–106, 2008.

Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proceedings of the Workshop on Learning Statistical Models from Relational Data at the 17th National Conference on Artificial Intelligence (AAAI)*, pages 13–20, 2000.

Jennifer Neville and David Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 170–177, 2005.

Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

Jennifer Neville, David Jensen, Lisa Friedland, and Michael Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, 2003a.

Jennifer Neville, David Jensen, and Brian Gallagher. Simple estimators for relational bayesian classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, pages 609–612, 2003b.

Jennifer Neville, Özgür Simsek, David Jensen, John Komoroske, Kelly Palmer, and Henry G. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 449–458, 2005.

Mark E. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Matthew J. Rattigan, Marc Maier, David Jensen, Bin Wu, Xin Pei, JianBin Tan, and Yi Wang:. Exploiting network structure for active inference in collective classification. In *Proceedings of the Workshop on Mining Graphs and Complex Structures at the 7th IEEE International Conference on Data Mining (ICDM)*, pages 429–434, 2007.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2): 107–136, 2006.

Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8(Jul):1623–1657, 2007.

Prithviraj Sen. Personal communication, 2008.

Prithviraj Sen and Lise Getoor. Empirical comparison of approximate inference algorithms for networked data. In *Proceedings of the Workshop on Open Problems in Statistical Relational Learning at the 23rd International Conference on Machine Learning (ICML)*, 2006.

Prithviraj Sen and Lise Getoor. Link-based classification. Technical Report CS-TR-4858, University of Maryland, College Park, MD, February 2007.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine, Special Issue on AI and Networks*, 29(3): 93–106, 2008.

Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probalistic models for relational data. In *Proceedings of the 18th Conference on Uncertainity in Artificial Intelligence (UAI)*, pages 485–492, 2002.

YongHong Tian, Tiejun Huang, and Wen Gao. Latent linkage semantic kernels for collective classification of link data. *Journal of Intelligent Information Systems*, 26(3):269–301, 2006.

Rudolph Triebel, Richard Schmidt, Oscar Martinez Mozos, and Wolfram Burgard. Instance-based AMN classification for improved object recognition in 2D and 3D laser range data. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2225–2230, 2007.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. *Advances in Neural Information Processing Systems (NIPS)*, 13:689–695, 2000.

Nevin Lianwen Zhang and David Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

Bin Zhao, Prithviraj Sen, and Lise Getoor. Event classification and relationship labeling in affiliation networks. In *Proceedings of the Workshop on Statistical Network Analysis (SNA) at the 23rd International Conference on Machine Learning (ICML)*, 2006.