McDowell, L., Gupta, K.M., & Aha, D.W. (2009). Using caution to explain and improve collective classification (Technical Report AIC-09-140). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Using Caution to Explain and Improve Collective Classification

Luke K. McDowell[1], Kalyan Moy Gupta[2], and David W. Aha[3]

[1]Dept. of Computer Science; U.S. Naval Academy; Annapolis, MD 21402
[2]Knexus Research Corp.; Springfield, VA 22153
[3]Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5514); Washington, DC 20375
lmcdowel@usna.edu, kalyan.gupta@knexusresearch.com, david.aha@nrl.navy.mil

Abstract. Many algorithms for collective classification (CC) have been shown to increase accuracy when instances are interrelated. Such algorithms must be carefully applied, however, since CC's use of estimated labels can in some cases *decrease* accuracy. Thus, a deeper understanding of algorithmic performances on data sets of different characteristics is needed. Although prior work has begun to study and compare such algorithms, many important questions remain unanswered. To address these limitations, we extend the recently introduced notion of *caution* in CC algorithms to predict which CC algorithms and training techniques will outperform others and identify the data characteristics for which such performance differences will be substantial. Using the theme of caution and our experimental results we demonstrate the close relationship between two very different algorithms (Gibbs sampling and Gradual Commit), show when they outperform less cautious algorithms, and explain multiple conflicting results from prior CC research.

Keywords: collective inference, approximate probabilistic inference, statistical relational learning

1 Introduction

Traditional methods for supervised learning assume that the instances 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 (e.g., "faculty" vs. "student" home page) than unlinked pages. Such *auto-correlation* (correlation of class labels among interrelated instances) has been observed in a wide variety of data [11], including situations where the relationships are implicit (e.g., email messages between two people are likely to share topics).

Collective classification (CC) is a methodology that simultaneously classifies related instances. To do so, CC uses a *base classifier* and iterative *collective inference*, enabling it often to attain higher accuracies than traditional methods when instances are interrelated [9,16,4,11,15]. Several algorithms have been used for collective inference, including relaxation labeling [2], iterative convergence techniques [9,6], loopy belief propagation (LBP) [16], and Gibbs sampling [4,11].

All collective inference algorithms exploit relational features based on uncertain (and thus noisy) estimation of class labels, and thus may in some cases actually decrease accuracy [11,14,15]. Consequently, there is a need to compare the behavior of CC algorithms on data sets with varying characteristics. Although recent work has begun to study such comparisons [11,14,15,7], close examination of these prior studies reveals several important and unanswered questions. First, Gibbs sampling is often regarded as one of the most accurate inference algorithms, and has been shown to work well for CC [4,11]. If so, why did Sen et al. [15] find no significant difference between Gibbs and the much less sophisticated Iterative Classification Algorithm (ICA)? Second, we recently showed that Gradual Commit (GC), a simple variant of ICA, outperformed both Gibbs and ICA on three real-world tasks [8]. Why would GC outperform Gibbs, and for what data characteristics are GC's gains significant?

To answer these questions, we extend the notion of *caution* – favoring more certain (i.e. less noisy) label estimations to diminish their negative effects – that we previously introduced to explain GC [8]. In this paper, we show that the advantages of caution also apply to Gibbs sampling, though they are achieved differently. In addition, we explain how to utilize caution for training classifiers in CC by applying a standard cross-validation technique. This technique for parameter tuning, which we call *CVPL* (*Cross-Validation Parameter Learning*) has not been used for CC, yet can provide significant performance advantages. We then show that, in contrast to prior results, GC and Gibbs perform very similarly — *if* the base classifier uses appropriate features and cautiously learns parameters suitable for CC. Also, both algorithms can significantly outperform the more *aggressive* (i.e., less cautious) ICA, although GC is computationally much less expensive than Gibbs.

Our contributions are as follows. First, we broaden the original notion of caution to include three distinct types of cautious behaviors and show how specific algorithms such as Gibbs benefit from it, as we previously did for GC. Second, we explain how to make a CC algorithm, cautious or otherwise, more cautious via CVPL to account for the algorithm's use of estimated class labels during testing. Third, we identify the data characteristics for which these cautious techniques should out-perform more aggressive approaches such as ICA and/or naïve parameter learning. In particular, we hypothesize that the cautious algorithms will outperform more aggressive versions when the data characteristics are such that the intermediate relational feature values estimated by the algorithms are highly uncertain. Fourth, we evaluate our hypotheses over a wide range of synthetic data using two base classifiers, several CC algorithms, and multiple baseline algorithms. Our results identify which data set characteristics lead to significant performance differences and highlight the importance of CC-specific parameter tuning. Finally, based on our findings, we answer the previously mentioned questions regarding CC.

We next summarize related work and collective classification in general. We then explain why CC needs to be cautious and describe three types of cautious behavior, followed by specific CC algorithms that use such caution. Finally, we present our experimental evaluation and discuss our findings.

2 Background on Collective Classification

In some classification tasks, the unlabeled instances can be implicitly or explicitly related (e.g., hyperlinked web pages). Standard classifiers ignore such relations and would typically classify a web page by considering only the features derived from its words. Accuracy can be increased by adding features derived from related instances (e.g., the words from hyperlinked pages). Even greater increases can occur when the class label(s) of the related pages are used to derive relevant *relational* features [4]. However, some or all of their labels are initially unknown and must be *estimated* to bootstrap the classification process. For example, initial class label estimates can be obtained using non-relational features only. Next, these estimates could be used to compute relational feature values and reclassify the instances. This process iterates, and may increase accuracy.

CC algorithms operate in this manner and thus *simultaneously* classify interrelated instances. They have two primary components:

- *Base Classifier*: To classify an instance *i* (e.g., a webpage), the base classifier uses *non-relational* features (e.g., the words in page *i*) and *relational* features (e.g., the most common class label among other pages linked to *i*). Many classifiers have been used to do this, including those derived from Naïve Bayes [4], Markov networks [16], k-nearest neighbor [8], and logistic regression [6].
- *Collective inference*: An inference algorithm (e.g., Gibbs sampling, LBP, ICA) is used to update the class labels (or conditional probabilities), which are then used to re-compute the relational feature values. This process repeats until some criteria or convergence test is met.

Jensen et al. [4] examine how some data characteristics and feature choices affect CC, but do not compare CC variants. Sen et al. [15] compare a different set of CC algorithms than we do here, vary fewer data characteristics, and do not focus on the topic of caution.

3 Types of Caution in CC and Why Caution is Important

In each iteration, a CC algorithm usually predicts the most likely class label for each instance and uses it to determine the next iteration's predictions. Although using label predictions encapsulates the influence of a linked instance and simplifies learning [4], it can be problematic. For example, iterating with incorrectly predicted labels can propagate and amplify errors [11,14,15].

To address this problem, we recently proposed the use of *caution* in a CC algorithm [8]. We defined an algorithm to be cautious if it sought to "explicitly identify and preferentially exploit the more certain relational information," and explained that GC is cautious because it selectively ignores the estimated class labels for which the classifier is less certain. Neville and Jensen [9] had introduced a simpler version of GC but compared it only with non-relational classifiers. We showed that GC could outperform ICA and Gibbs, but did not identify the conditions under which such gains hold.

In this paper, we broaden our original notion of caution to identify three general types of cautious techniques for CC. All three address the key potential problem with CC (which is also its potential strength): its use of estimated labels during testing. Below we summarize all three types and identify a specific example of that type for further study.

- **Caution type #1: Favoring more certain information**. CC algorithms may choose to favor predicted information that has higher confidence. This is the approach taken by Gradual Commit (GC), which chooses to use only the most certain labels at the beginning of it's operation, then "gradually" incorporates less certain predictions in later iterations.
- **Caution type #2: Reasoning with uncertainty.** At each iteration, instead of always selecting the most likely class label for each instance (like ICA), a CC algorithm can utilize the estimated label distribution of each instance. For example, techniques like LBP and relaxation labeling directly reason with the estimated label distributions. Alternatively, at each iteration Gibbs sampling re-samples the label of each instance based on its estimated distribution. For further study of this class of techniques, we select Gibbs sampling, in part because it has been frequently studied and its generalization behavior tends to be more consistent than relaxation labeling or LBP [15].
- **Caution type #3: Training influenced by test procedure uncertainty.** Training a CC algorithm can be 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 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 CVPL to achieve this goal.

Section 4 describes how these ideas can be applied. Later, our experimental results demonstrate when they lead to significant performance improvements.

4 Applying Caution to Collective Classification

The previous section described three general types of cautious techniques for CC. Each addresses the fundamental problem of potential estimation errors in labels during collective inference. Some of the techniques can be combined, and at least one is used or is applicable to every CC algorithm known to us.

In this section, we provide examples of how each of the three types of caution can be applied by describing specific CC algorithms that exploit them. Table 1 summarizes the four CC algorithms that we will consider (along with one non-collective baseline) and the types of caution they can exploit. Below, we first describe the non-cautious ICA algorithm, then explain how GC adds Type 1 caution to it. Second, we summarize Gibbs sampling and explain how it exhibits Type 2 caution. Third, we describe the wvRN algorithm, which also uses Type 2 caution. Finally, we

Table 1. The five classification algorithms considered in this paper. `CO` is a baseline (non-collective) algorithm that only uses non-relational features. The other four algorithms are CC algorithms. For the Caution Types (see Section 3), a black check indicates that type is used, while a grey check indicates that the algorithm *could* profitably use that type; our experiments consider both variants.

	Features Used		Caution Type Used		
	Non-relat.	Relat.	1	2	3
`CO` (content only)	✓				
`wvRN` (weighted vote Relat. Neighbor)		✓		✓	
`ICA` (Iterative Classification Alg.)	✓	✓			✓
`GC` (Gradual Commit)	✓	✓	✓		✓
`Gibbs` (Gibbs Sampling)	✓	✓		✓	✓

describe CVPL, our parameter learning technique for CC that can add caution to any CC algorithm (excluding the few that do not learn from a training set).

ICA: Figure 1 displays pseudocode that can represent either ICA or GC; DoGC and DoCVPL are boolean parameters that control its operation. In particular, when DoGC is *false*, then Figure 1 represents ICA, which operates as follows. In step 1, it computes the relational features' values for the fully labeled training set. In Step 2, a base classifier is learned using the training data. If DoCVPL is *true*, then this step utilizes Type 3 caution via CVPL, which is described more fully later. Step 3 predicts the test instances' labels using only non-relational features. In steps 6-7, ICA estimates the relational features' values based on its predictions and reclassifies the test set using all features. These steps are then repeated for *n* iterations. Note that step 6 uses all available labels for feature computation and step 7 picks the most likely label for each instance based on the new predictions, so this process utilizes neither Type 1 nor Type 2 caution. Step 8 returns the final set of estimated class labels.

GC: In step 6, ICA assumes that the assigned instance labels are all equally likely to be correct. When DoGC is true, the algorithm becomes GC, a more cautious algorithm because it only considers label assignments for which it has more confidence (Type 1 caution). Specifically, step 5 commits only the best *K* of the current label assignments (we use posterior probability as a confidence measure) and sets all other labels to *unknown*. Step 6 computes the relational features using *only* the committed labels, and step 7 classifies using this information. Step 5 gradually increases the number of test set labels that are committed per iteration (e.g., 0%, 10%, 20%,…, up to 100% when *n*=10). Instances committed in an iteration *j* are not necessarily committed again in iteration *j+1*.

GC favors more confident information (Type 1 caution) by *ignoring* instances whose labels are estimated with lower confidence. Step 5 executes this preference, but it affects the algorithm in several ways. First, leaving some label assignments as *unknown* in step 5 causes the feature value computation in step 6 to ignore those labels. Since this computation depends only on the most reliable label assignments, subsequent assignments should also be more reliable. Also, a secondary effect is that the computed value of some features will be *unknown* (e.g., when an instance links

```
ICC(Tr,Te,NR,R,n,C,DoGC,DoCVPL) =
// Tr=Training data, Te=Test data,  NR=non-relational features,
// R=rel. features, n=#iters, C=classifier, DoGC=do Gradual Commit,
// DoCVPL=do Cross-Validation Parameter Learning
1   Tr.R.values←setRelationalFeatures(Tr,R)
2   M←induce_model(Tr,NR,R,C,DoCVPL)          // Train
3   Te.Labels←classify(Te,Tr,M,NR,∅,C)         // Bootstrap
4   for j = 0 to n                             // Iterate
5       if (DoGC) // Keep only top K labels (none when j=0)
            K = (j / n) * |Te|
            Te.Labels←commit_best_k (Te.Labels, K)
6       Te.R.values←setRelationalFeatures(Te∪Tr,R)
7       Te.Labels←classify(Te,Tr,M,NR,R,C)     // Classify
8   return Te.Labels // return most likely class per test instance
```

Figure 1. Pseudocode for ICA (when DoGC is false) and GC (when DoGC is true). We use *n* = 10.

only to instances labeled *unknown*). Second, a realistic CC scenario's test set may have links to training instances (e.g., new web pages may link to pages with known labels); these are the "most certain" labels that link to the test set and thus may aid classification. GC exploits only these labels when j=0. In this case, step 5 sets the value of all labels in the test set to *unknown*, but some relational feature values in step 6 can be still be computed based on known labels in the training set. Thus, the known labels influence the first classification in Step 7, before any estimated labels are used, and in subsequent iterations.

In prior work [8], we separately evaluated GC's performance benefit from these two effects (favoring more confident labels vs. favoring known labels), and found both helpful. For this paper, we likewise found benefits from both, but for simplicity only report results with them together, since both are Type 1 cautious behaviors.

Gibbs Sampling: Figure 2 summarizes how Gibbs sampling can be applied to collective inference. Steps 1-3 are identical to those in Figure 1, except that the classifier must output distributions with the likelihood of each class. In step 5, within the loop, the algorithm probabilistically samples the current class label distributions and assigns a label to each instance based on its distribution. In step 6, it records these labels, and in step 7 it computes the relational feature values given the current class labels. In step 8, it re-computes the posterior class label probabilities given these relational features. The process then repeats. When the process terminates, the statistics recorded in step 6 approximate the joint distribution of class labels, which is used in step 9 to identify each instance's most likely class label. These labels are returned in step 10.

Like GC, Gibbs is cautious in its use of estimated labels, but in a different way. In particular, GC exercises caution in step 5 by *ignoring* (at least for some iterations) labels that have lower confidence (Type 1 caution). In contrast, Gibbs exercises caution by *sampling*, in step 5, values from each instance's predicted label distribution (Type 2 caution) – causing instances with lower prediction confidence to reflect that uncertainty via higher fluctuation in their assigned labels. We expect Gibbs to perform better, since it makes use of more information, but this requires

```
GibbsCC(Tr,Te,NR,R,n,C,DoCVPL) =
// Tr=Training data, Te=Test data,  NR=non-relational features,
// R=rel. features, n=#iterations, C=classifier,
// DoCVPL = do Cross-Validation Parameter Learning
1   Tr.R.values←setRelFeatures(Tr,R)
2   M←induce_model(Tr,NR,R,C,DoCVPL)              // Train
3   Te.ClassProbs←classify(Te,Tr,M,NR,∅,C)        // Bootstrap
4   for j =1 to n                                 // Iterate
5       Te.Labels ←sampleDist(Te.ClassProbs)      // Sample
6       Te.Stats←updateStats(Te.Stats,Te.Labels)  // Take stats
7       Te.R.values←setRelFeatures(Te∪Tr,R)
8       Te.ClassProbs←classify(Te,Tr,M,NR,R,C)    // Classify
9   Te.Labels←pickMostLikelyClass(Te.Stats)
10  return Te.Labels     // return most likely class for each instance
```

Figure 2. Psuedocode for CC using Gibbs sampling. We use n = 1000.

careful confirmation. Furthermore, the sophistication of Gibbs comes at a cost – ICA and GC generally converge in about 10 iterations, whereas Gibbs typically requires ***thousands*** of iterations to yield good performance (for all three algorithms the cost per iteration is similar). Thus, `GC`'s simplicity and speed may make it a promising alternative to `Gibbs`.

Weighted-Vote Relational Neighbor Classifier (wvRN). wvRN is a relational-only CC algorithm that Macskassy and Provost [7] argued should be considered as a baseline for all CC evaluations. At each iteration, each instance *i* updates its estimated class distribution by averaging the current distributions of each of its linked neighbors. wvRN ignores all non-relational features. Thus, wvRN is useful only if the test set links to some instances with known labels to "seed" the inference process. Macskassy and Provost showed that this simple algorithm could work well if auto-correlation of instance labels was high and enough known labels were available. Since wvRN computes directly with the estimated label distributions, it exercises Type 2 caution. However, unlike the other CC algorithms, it does not learn from a training set, and thus parameter learning with CVPL (Type 3 caution) does not directly apply.

Cross-Validation Parameter Learning (CVPL). CC algorithms typically train a base classifier on a fully-labeled training set, then use that base classifier with some collective inference algorithm to classify the test set. Unfortunately, the classifier learned from the (fully labeled) training set may tend to produced poor estimates of important parameters related to the relational features (e.g., feature weights, conditional probabilities), since these features depend upon labels that may be estimated incorrectly during collective inference. We found that, in some cases, this can have a large negative impact on performance.

One way to address this problem is to perform automated parameter tuning based on cross-validation (e.g., [5]). However, unlike the typical situation where the training and test sets have the same known features, the CC situation differs, since relational features must be estimated for testing. CVPL performs automated tuning by repeatedly evaluating a learned base classifier, with the collective inference algorithm, on a holdout set (a subset of the training set) using different values of a parameter that controls the impact of relational features. It selects the value that yields the best performance and applies it during testing. In essence, we use CVPL to set a classifier parameter that compensates for the bias that would otherwise be incurred from training on a fully-labeled set while testing using estimated labels.

We expect CVPL's utility to vary based upon the number of "known labels" that are available to the test set. If the test set has many such known labels, then there is less discrepancy between the training and test environments, and hence less need to apply CVPL. Conversely, if few labels are known, there is a large discrepancy and we expect CVPL to have a large effect. Also, because almost all CC algorithms use a base classifier that learns parameters based on relational features, CVPL is widely applicable (e.g., it can be applied to ICA, GC, and Gibbs, as shown in Table 1).

CVPL has not been previously used for CC. A possible exception is Lu and Getoor [6], who appear to have used a form of CVPL to tune a relational parameter, but they did not discuss its need, the specific procedure, or the performance impact. Here, we explain its importance for CC, and empirically confirm that it can significantly increase accuracy.

5 Evaluation Methodology

Our goal is to investigate the performance and the utility of the three types of caution over a wide range of data characteristics (e.g., link density, auto-correlation).

Data. We use a synthetic data generator with two components: a Graph Generator and an Attribute Generator. First, the Graph Generator [15] 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 specifies the probability that the linked nodes have the same class label; higher values yield higher auto-correlation. The final number of links is approximately $N_I/(1\text{-}ld)$, and the final link degrees follow a power law distribution, which is common in real networks [1]. To make this a practical study, we chose default parameter values that mimic characteristics of two frequently studied CC datasets, Cora and Citeseer [8,11,15]. In particular, N_C=5 classes and Table 2 shows additional default values. We chose N_I=250 instances, a smaller value than with Cora/Citeseer, to reduce CC execution time, but larger values did not change our trends. We use synthetic data instead of the actual Cora/Citeseer data to allow us to directly vary all of the interesting data characteristics.

Second, the Attribute Generator[1] uses a method motivated by our observations of common CC datasets. We found that, unlike synthetic models used in prior studies, different attributes vary in their utility for class prediction. To simulate this we construct 10 (N_A) binary attributes A_j ($j \in [0,N_A\text{-}1]$) and generate each attribute A_j's values so that they are most predictive of a particular class C_k, where k=j mod N_C. For node i with class C_i, we set the probability that its j^{th} attribute X_{ij} has value 1 as follows:

[1] For simplicity, we henceforth refer to non-relational features as *attributes*.

Table 2. The four data generation parameters that we vary to investigate our hypotheses. Default values are based on measurements from the Cora and Citeseer datasets [8,15].

Data parameter	Abbrev	Default value	Hypothesis: Relative gain of caution will increase as parameter value…
Degree of homophily	*dh*	0.8	…increases. (H1)
Attribute predictiveness	*ap*	0.6	… decreases. (H2)
Link density	*ld*	0.2	…decreases. (H3)
Labeled proportion	*lp*	0.0, 0.2	…decreases. (H4)

$$\begin{aligned} P(X_{ij}=1|C_i=C_k) &= 0.15+(ap-0.15)*j/(N_A-1) && \text{if } k=j \bmod N_C \\ &= 0.1 && \text{if } k=(j-1)\bmod N_C \\ &= 0.05 && \text{if } k=(j+1)\bmod N_C \\ &= 0.02 && \text{Otherwise} \end{aligned}$$

The first line indicates that, when C_k is the class associated with A_j (i.e., $k=j$ mod N_C), $P(X_{ij}=1|C_i=C_k)$ ranges from 0.15 for j=0 to *ap* (attribute predictiveness) for j=9. However, X_{ij} may also be 1 when C_k is some other class; the next three lines encode this class ambiguity. This is based on our observations of Cora and Citeseer and is similar to the binomial distribution of Sen et al. [15].

We focus on two evaluation conditions: the *out-of-sample* task, where test set nodes do not link to nodes with known labels, and the *in-sample* task, where they do link to known labels (e.g., in the training set). Both types of tasks may emerge in real-world situations [10]. We simulate the latter by using a parameter for test set generation called *lp* (*labeled proportion* – the proportion of test instances with known labels), where accuracy is evaluated only over the unknown labels. We use a default *lp* value of 0.2, the average of the values found for Cora and Citeseer. For the out-of-sample task, we set *lp*=0.0 (a second default value).

Hypotheses. We evaluate the following hypotheses, which are summarized in Table 2. In general, we expect cautious behaviors to be more important as the uncertainty in the relational features increases. Thus, each hypothesis varies one data generation parameter that impacts such uncertainty, and compares the resultant performance of GC and Gibbs (Types 1 or 2 caution) with ICA (non-cautious). Section 6 also briefly summarizes results for CVPL (Type 3 caution).

H1: *The relative gain of GC and Gibbs vs. ICA increases with the degree of homophily (dh).* Larger homophily means higher label auto-correlation among related instances, in which case the relations are more predictive. This magnifies the impact that an error in a predicted label can have on neighboring instances. Therefore, we expect cautious algorithms to improve classification by a greater margin in such cases.

H2: *The relative gain of GC and Gibbs vs. ICA 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 GC and Gibbs vs. ICA 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 GC and Gibbs vs. ICA increases with decreasing labeled proportion* (*lp*). When *lp* is high, only a small fraction of each instance's neighbors have estimated labels (most are known with certainty), and thus it is less important to use estimated labels cautiously. However, as *lp* decreases the uncertainty in estimated labels increases and thus we expect the impact of caution to increase.

CC Algorithms. We evaluate the 5 algorithms listed in Table 1. `CO` is a non-CC baseline, while `wvRN` is a collective but relational-only baseline. `ICA`, `GC`, and `Gibbs` are collective algorithms that use both relational features and attributes, with `GC` and `Gibbs` being more cautious than `ICA`. For `Gibbs`, we use 1000 iterations with a burn-in of 200.

Classifiers. To account for possible variations in overall CC performance trends due to the effect of the underlying classifier, we tested two base classifiers with each CC algorithm except `wvRN` (which does not use a base classifier). The first classifier is Naïve Bayes (NB). With the relational features, NB employed a Dirichlet prior with a single "hyperparameter" α [3]. The manual setting used α=1.0, which reduces to the common Laplace smoothing. CVPL searched for α in [1,5000] (larger values yield less extreme probabilities). The second classifier is k-Nearest Neighbor (kNN); we empirically set *k*=5. When computing similarity, non-relational features (attributes) had a weight of 1, and relational features had weight W_R. For a manual setting, W_R=1.0. CVPL searched for W_R in [0.01,5]. The specific similarity function, for instances *i* and *j*, was:

$$sim(i,j) = \sum_k w_k (1 - |f_k(i) - f_k(j)|) / \sum_k w_k$$

where w_k is the weight of feature *k*, and $f_k(i)$ is the value of feature *k* for instance *i*. Weighted similarity was used for voting.

Instance Representation. Each instance 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, we considered two different types of relational features: proportion and multinomial. For NB, we experimented with both and found that multinomial consistently performed best across the different algorithms. For kNN, only the proportion features are directly applicable, so we used that. Below, we describe each feature in detail:

- *Proportion* (used by kNN): This type represents the proportion of neighbors that belong to a particular class. There is one such feature f_c per class label *c*, yielding 5 total. The value of a feature $f_c(i)$ = Neighbors$_c$(*i*) / Neighbors(*i*), where Neighbors(*i*) is the number of instances hyperlinked to instance *i* whose values are not *unknown*, and Neighbors$_c$(*i*) is the number whose current label is *c*. If Neighbors(*i*) is zero, then $f_c(i)$ is set to *unknown*.
- *Multinomial* (used by NB): Proportion features, like other features that have been used for CC such as *count* and *exists*, are features that aggregate the labels of an

instance's neighborhood to produce a single value (for proportion, a value between zero and one). In contrast, a multinomial feature uses a multinomial set to fully represent the estimated labels of all of an instance's neighbors. During inference, each label in the set (excluding *unknown* labels) is then separately used to update the conditional probability that an instance has true label *c*. This is the "independent value" approach of Neville, Jensen, and Gallagher [12], which they also used in their later work (e.g., [11]).

Test Procedure. For each control condition (i.e., data generated with a combination of *dh*, *ap*, *ld*, and *lp* values, see Table 2) we ran 25 random trials. For each trial, we generated *training*, *holdout*, and *test* data sets of 250 nodes (i.e., instances) each. The *holdout set*, when not used for parameter learning, was merged with the training set. We measured classification accuracy on the test set.

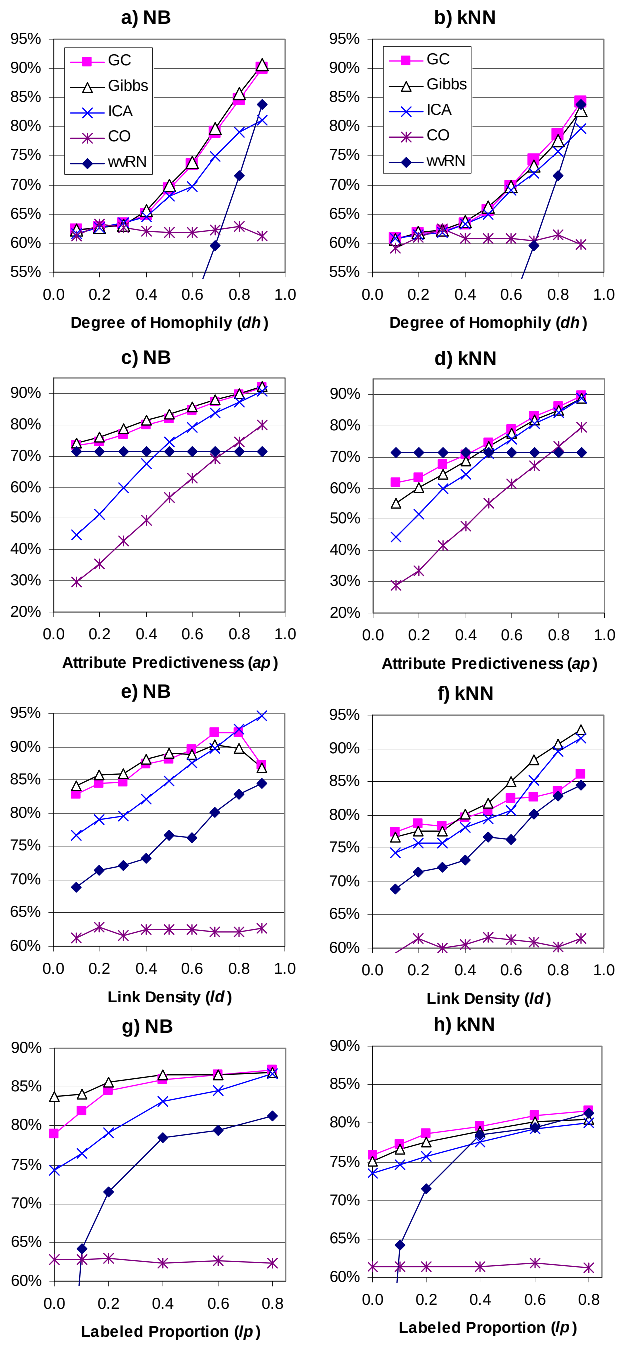
Statistical Analysis. To compare algorithms for a single control condition, we used a one-tailed paired t-test accepted at the 95% confidence level. For hypotheses H1-H4, we compared two algorithms (e.g., GC vs. ICA) for each independent variable (X) (e.g., *ld*) as follows: For each trial, we computed the difference in the algorithms' classification accuracies (e.g., 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 $p<0.05$. Below, we only report results for *lp*=0.2 (the in-sample task). However, regression analyses for results with *lp*=0 (the out-of-sample task) supported the same conclusions. Likewise, we only mention the slope *b* for GC vs. ICA, but Gibbs vs. ICA yielded the same conclusions unless otherwise noted.

6 Results

Figure 3 display plots for average classification accuracy. These results all include CVPL except for where it does not apply (CO and wvRN); later we evaluate its effects separately. We consider each of the four hypotheses (see Table 2) in turn, then consider wvRN and CVPL. Percentages given are the raw difference between accuracies (e.g., we report a 6% gain due to improving the accuracy from 60% to 66%, rather a percentage change of 10%).

Result 1: *GC and Gibbs outperform ICA by increasing amounts as the degree of homophily (dh) increases.* The regression analyses all found positive values for the slope *b* (e.g., 0.10 for GC-NB and 0.05 for GC-kNN, all $p < 0.001$), so we accept H1.

Figures 3a and 3b show the details. When homophily is low, CC offers little gain and all algorithms except wvRN perform comparably. As the potential advantages of CC increase with higher homophily, the relative gain of GC and Gibbs increases (e.g., for NB from 4% at *dh*=0.6 to 9-10% at *dh*=0.9). These gains are significant for *dh*≥0.5 for NB (excluding *dh*=0.6 for Gibbs) and for *dh*≥0.7 for kNN.


a) NB
b) kNN
c) NB
d) kNN
e) NB
f) kNN
g) NB
h) kNN
GC
Gibbs
ICA
CO
wvRN
Degree of Homophily (*dh*)
Attribute Predictiveness (*ap*)
Link Density (*ld*)
Labeled Proportion (*lp*)

Result 2: *GC and Gibbs outperform ICA by increasing amounts as attribute predictiveness (ap) decreases*. The regression analyses of *ap* vs. the difference between ICA and either Gibbs or GC found a negative slope *b* (e.g., -0.34 for GC-NB and -0.18 for GC-kNN, all p <0.001), so we accept H2.

Figures 3c and 3d show details. When *ap* is 0.6 (the default), GC and Gibbs outperform ICA by 6-7% for NB and 2-3% for kNN. However, as *ap* decreases to 0.2, label uncertainty increases (as evidenced by the drop for CO), causing the relative gain to increase to 23-25% for NB and 9%-12% for kNN. These gains are significant for all values of *ap* shown, except for Gibbs-kNN at *ap*=0.9.

Result 3 *GC and Gibbs outperform ICA when link density (ld) is low, but have mixed results when ld is high.* The regression analyses found a significant negative slope for most cases (e.g., -0.14 for GC-NB and -0.12 for GC-kNN), but not for Gibbs-kNN. Thus, we do not accept H3.

GC and Gibbs significantly outperform ICA when *ld* is low to moderate (Figures 3e and 3f)). However, the performance of ICA consistently improves with *ld*; enabling it to eventually significantly outperform GC and Gibbs for NB, and GC for kNN. Intuitively, when the link graph is dense, the relational features are relatively unaffected by a few incorrect labels. Thus, if accuracy is high, ICA outperforms GC by simply using all available labels, since almost all are correct. Other results confirm that when attribute predictiveness is low, this effect does not hold and GC outperforms ICA even when *ld* is high. Since Gibbs uses all available information, at high *ld* it performs on par with ICA when using kNN – but not with NB. Instead, when *ld* is very high the NB classifier tends to produce probability estimates that are very close to zero or one, in which case Gibbs sampling is known to perform poorly.

Result 4: *GC and Gibbs outperform ICA more as the labeled proportion (lp) decreases.*. All regression analyses found negative valued slopes (p < 0.002). Thus, we accept H4.

As expected, Figures 3g and 3h show that the greater uncertainty caused by small *lp* yields larger gains for the more cautious algorithms. In addition, for GC and Gibbs (but not for ICA), CC accuracy improves by only 2-3% from *lp*=0.2 to *lp*=0.8, suggesting that the more cautious CC algorithms are effective in replicating the gain possible if almost all neighbors were known, even when few are actually known.

wvRN: wvRN's performance depends on homophily, link density, and *lp*. In our study, wvRN was competitive with the other CC algorithms only when homophily and/or *lp* was high, or when the attributes were not very predictive. For tasks in which a high *lp* or weak attributes are presumed, wvRN's accuracy should be more competitive. On the other hand, wvRN requires that some labels are known in the test set, so it is not applicable when *lp* = 0 (the out-of-sample task).

CVPL: The results described above all used CVPL. We found that when some test labels were known (*lp* > 0), CVPL had only a small impact. When all labels were unknown (*lp* = 0), however, the use of CVPL was sometimes important. Using CVPL almost always increased accuracy by at least a few percentage points. Moreover, for each algorithm there was *some* type of data for which not adjusting for CC training biases (i.e., not using CVPL) led to very poor performance. Using CVPL corrected for these problems. For instance, CVPL improved performance dramatically for the NB classifier (e.g., by 15-33% with `Gibbs`) when link density was high and for the kNN classifier (e.g., by 6-9% with `ICA`) when homophily was moderate. Thus, applying CVPL in all of our other experiments seemed advisable for maximizing performance and ensuring the fairest comparisons.

7 Discussion and Explanation of Prior Results

Our results enable us to explain the unanswered questions from Section 1:

- *Why did Sen et al. [15] find no consistent difference between Gibbs and ICA?* In contrast, Gibbs had worked well in other work, and in this paper we found that Gibbs (and GC) often did significantly increase 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 particular "snowball" sampling method that we found produced an effective labeled proportion of at least 0.5 – a region where we showed that the use of caution has little impact. Also, they did not vary attribute predictiveness, which we show is a significant factor in the relative performance of more cautious CC algorithms.
- *Why did McDowell et al. [8] find that GC outperformed Gibbs, while the results of this paper generally found Gibbs performing at least as well as GC?* To investigate, we re-ran the earlier experiments (which used three real-world datasets), but with two variations informed by our now refined understanding of caution. First, we used CVPL with both the NB and kNN classifiers. Second, we changed the NB classifier to use multinomial features (instead of proportion). The use of multinomial features is in some sense more cautious, since it retains all label info rather than aggregating, and our results showed it yielded better accuracy for NB (it didn't apply for kNN). With these enhancements, Gibbs's relative performance improved, so that GC and Gibbs both significantly outperformed ICA, but Gibbs and GC were not significantly different from each other. Thus, more careful learning and representation choices yields results for the three previously-used datasets that are highly consistent with those reported here for synthetic data.
- *When will cautious algorithms outperform more aggressive variants?* We found that using more cautious CC frequently and sometimes dramatically increased accuracy. In general, cautious CC performs comparatively better whenever there is more uncertainty in the estimated relational feature values (e.g., as occurs when the attribute predictiveness is low) or when the effect of any such uncertainty is magnified (e.g., when homophily 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. To increase the robustness of our conclusions, we performed additional tests using an alternative graph generator [11], an alternative attribute generator (a Naïve Bayes model learned directly from the raw Cora data), and different relational feature choices (neighbor counts thresholded to yield a binary value). The results were consistent with those reported here.

This paper also examined several types of cautious behavior for CC. In particular, GC and Gibbs used different types of caution: favoring confident information (Type 1) vs. re-sampling based on label distributions (Type 2). Because Gibbs more fully uses the available information, we expected Gibbs to perform better than GC in some situations. Instead, we found that, while Gibbs sometimes had a small gain over GC, they performed remarkably similarly given their large algorithmic differences. This is particularly interesting because, while they require similar execution time per iteration, deterministic GC typically needs several orders of magnitude fewer iterations to achieve good results. Thus, GC's simplicity and speed may make it a promising alternative to Gibbs.

Regarding Type 3 caution, we found that CVPL generally increased accuracy, sometimes substantially. It is well-known that parameter tuning is important for learning non-relational classifiers. We show for the first time that it can be especially critical for CC due to CC's reliance on uncertain labels during testing. For example, further results showed that when link density was high, Gibbs-NB with a naïve α (prior hyperparameter) of 1.0 attained 99% of the accuracy attainable with any α – *if* most test labels were known (e.g., *lp*=0.8). However, when *lp*=0 this strategy's accuracy was just 61% of optimal. Using CVPL instead increased accuracy, as we also found with ICA and GC, and expect to find with other classifiers and CC algorithms.

8 Conclusion

We identified and investigated three types of "cautious" algorithmic behaviors that address the problem of using estimated test set labels with collective classification (CC). Each behavior had been seen before in some context, but none had been subjected to analysis that explained its behavior in the context of cautious algorithms in general and that identified the data characteristics for which it outperformed more aggressive variants like ICA. In particular, we showed that cautious techniques produce accuracy gains that are especially large when the data characteristics are such that there is greater uncertainty in the values of the relational features. These results enabled us to answer several important unanswered questions from previous work.

We showed that these cautious techniques work well with two base classifiers and expect that they will provide comparable benefits with others, although this needs to be demonstrated empirically. Also, our investigation should be extended to compare with other CC techniques that exercise caution including LBP and Markov Logic Networks [13]. Finally, we plan to investigate techniques that will improve cautious CC's performance on data with high link density or other extreme conditions.

References

1. Bollobas, B., Borgs, C., Chayes, J., & Riordan, O. (2003). Directed scale-free graphs. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (pp. 132-139). Baltimore, MD: ACM.

2. Chakrabarti, S., Dom, B., and Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *Proceedings of the International Conference on Management of Data* (pp. 307-318). Seattle, WA: ACM.
3. Heckerman, D. (1995) *A tutorial on learning with Bayesian networks* (Technical Report MSR-TR-95-06). Redmond, WA: Microsoft Research.
4. Jensen, D., Neville, J., and Gallagher, B. (2004). Why collective inference improves relational classification. *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining* (pp. 593-598). Seattle, WA: ACM.
5. Kohavi, R., and John, G.E. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273-324.
6. Lu, Q., and Getoor, L. (2003). Link-based classification. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 496-503). Washington, DC: AAAI.
7. Macskassy, S., and Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, **8**, 935-983.
8. McDowell, L., Gupta, K. M., and Aha, D.W. (2007). Cautious inference in collective classification. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*. Vancouver, Canada: AAAI Press.
9. Neville, J., and Jensen, D. (2000). Iterative classification in relational data. In L. Getoor and D. Jensen (Eds.) *Learning Statistical Models from Relational Data: Papers from the AAAI Workshop* (Technical Report WS-00-06). Austin, TX: AAAI.
10. Neville, J., and Jensen, D. (2005). Leveraging relational autocorrelation with latent group models. *Proceedings of the Fifth International Conference on Data Mining.* Houston, TX: IEEE.
11. Neville, J., and Jensen, D. (2007). Relational dependency networks. *Journal of Machine Learning Research,* **8**, 653-692.
12. Neville, J., Jensen, D., and Gallagher, B. (2003). Simple estimators for relational bayesian classifiers. *Proceedings of the Third IEEE International Conference on Data Mining*. Melbourne, FL: IEEE.
13. Richardson, M., and Domingos, P. (2006). Markov logic networks. *Machine Learning,* **62**(1-2), 107-136.
14. Sen, P., and Getoor, L. (2006). Empirical comparison of approximate inference algorithms for networked data. In A. Fern, L. Getoor, and B. Milch (Eds.) *Open Problems in Statistical Relational Learning: Papers from the ICML Workshop.* Pittsburgh, PA: www.cs.umd.edu/projects/srl2006.
15. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. To appear in *AI Magazine*.
16. Taskar, B., Abbeel, P., and Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence* (pp. 485-492). Edmonton, Canada: Morgan Kaufmann.