Labels or Attributes? Rethinking the Neighbors for Collective Classification in Sparsely-Labeled Networks

Luke K. McDowell
Department of Computer Science
U.S. Naval Academy
Annapolis, MD U.S.A.
lmcdowel@usna.edu

David W. Aha
Navy Center for Applied Research in AI
Naval Research Laboratory, Code 5514
Washington, DC U.S.A.
david.aha@nrl.navy.mil

ABSTRACT

Many classification tasks involve linked nodes, such as people connected by friendship links. For such networks, accuracy might be increased by including, for each node, the (a) *labels* or (b) *attributes* of neighboring nodes as model features. Recent work has focused on option (a), because early work showed it was more accurate and because option (b) fit poorly with discriminative classifiers. We show, however, that when the network is sparsely labeled, "relational classification" based on neighbor attributes often has higher accuracy than "collective classification" based on neighbor labels. Moreover, we introduce an efficient method that enables discriminative classifiers to be used with neighbor attributes, yielding further accuracy gains. We show that these effects are consistent across a range of datasets, learning choices, and inference algorithms, and that using *both* neighbor attributes and labels often produces the best accuracy.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Link-based classification; statistical relational learning; semi-supervised learning; collective inference; social networks

1. INTRODUCTION

Many problems in communications, social networks, biology, business, etc. involve classifying nodes in a graph. For instance, consider predicting a class label for each page (node) in a set of linked webpages, where some node labels are provided for learning. A traditional method would use the attributes of each page (e.g., words in the page) to predict its label. In contrast, *link-based classification* [2, 13] also uses, for each node, the attributes or labels of *neighboring* pages as model features. If "neighbor labels" are used, then an iterative algorithm for *collective inference* is needed, since many labels are initially unknown [3]. If, on the other hand, "neighbor attributes" are used, then a single step of *relational inference* suffices, since all attribute values are known.

Despite the additional complexity of inference, recent work has used collective inference (CI) much more frequently than relational inference (RI) for two reasons. First, multiple algorithms for CI (e.g., belief propagation, Gibbs sampling, ICA) can substantially increase classification accuracy [8, 10]. In contrast, comparisons found RI to be inferior to CI [3] and to sometimes even decrease accuracy compared to methods that ignore links [2]. Second, although RI does not require multiple inference steps, using neighbor attributes as model features is more complex than with neighbor labels, due to the interplay between the larger number of attributes (vs. one label) and a varying number of neighbors for each node. In particular, RI does not naturally mesh with popular, discriminative classifiers such as logistic regression.

Most work on link-based classification assumes a fully-labeled training graph. However, often (e.g., for social and webpage networks) collecting the node attributes and link structure for this graph may be easy, but acquiring the desired labels can be much more expensive [11, 6]. In response, recent studies have examined CI methods with partially-labeled training graphs, using some semi-supervised learning (SSL) to leverage the unlabeled portion of the graph [15, 1, 6]. However, because using neighbor attributes seemed difficult and unnecessary, none evaluated RI.

Our contributions are as follows. First, we provide the first evaluation of link-based classification that compares models based on neighbor labels (CI) vs. models based on neighbor attributes (RI), for sparsely-labeled networks. Unlike prior studies with fully-labeled training networks, we find that RI is often significantly more accurate than CI. Second, we introduce an efficient technique, Multi-Neighbor Attribute Classification (MNAC), that enables discriminative classifiers like logistic regression to be used with neighbor attributes, further increasing accuracy. Finally, we show that the advantages of RI with MNAC remain with a variety of datasets, learning algorithms, and inference algorithms. We find that, surprisingly, RI's gains remain even for data conditions where CI was thought to be clearly preferable [3].

2. LINK-BASED CLASSIFICATION

Assume we are given a graph $G = (V, E, X, Y, C)$ where V is a set of nodes, E is a set of edges (links), each $\vec{x_i} \in X$ is an attribute vector for a 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. We are also given a set of "known" values Y^K for nodes $V^K \subset V$, so that

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
2013 ACM 978-1-4503-2263-8/13/10
http://dx.doi.org/10.1145/2505515.2505628

Table 1: Types of models, based on the kinds of features used. Some notation is adapted from Jensen et al. [3].

Model	Self attr.	Neigh. attr.	Neigh. labels
RCI	✓	✓	✓
RI	✓	✓	
CI	✓		✓
SELFATTRS	✓		
NEIGHATTRS		✓	
NEIGHLABELS			✓

$Y^K = \{y_i | v_i \in V^K\}$. Then the *within-network classification task* is to infer Y^U, the values of Y_i for the remaining nodes V^U with "unknown" values ($V^U = V \setminus V^K$).

For example, given a (partially-labeled) set of interlinked university webpages, consider the task of predicting whether each page belongs to a professor or a student. There are three kinds of features typically used for this task:

- **Self attributes:** features based on the the textual content of each page (node), e.g., the presence or absence of the word "teaching" for node v.
- **Neighbor attributes:** features based on the attributes of pages that link to v. These may be useful because, e.g., pages often link to others with the same label.
- **Neighbor labels:** features based on the labels of pages that link to v, such as "Count the number of v's neighbors with label `Student`."

Table 1 characterizes classification models based on the kinds of features they use. The simplest, baseline models use only one kind. First, SELFATTRS uses only self attributes. Second, NEIGHATTRS classifies a node v using *only* v's neighbors' attributes. This model has not been previously been studied, but we use it to help measure the value of neighbor attributes on their own. Finally, NEIGHLABELS uses only neighbor labels. For instance, it might repeatedly average the predicted label distributions of a node's neighbors; this performs surprisingly well for some datasets [5].

Other models combine self attributes with other features. If a model also uses neighbor attributes, then it is performing "relational inference" and we call it RI. A CI model uses neighbor labels instead, via features like the "count `Students`" described above. However, this is challenging, because some labels are unknown and must be estimated, typically with an iterative process of *collective inference* (i.e., CI) [3]. CI methods include Gibbs sampling, belief propagation, and ICA (Iterative Classification Algorithm) [10].

We focus on ICA, a simple, popular, and effective algorithm [10, 1, 6]. ICA first predicts a label for every node in V^U using only self attributes. It then constructs additional relational features X_R using the known and predicted node labels (Y^K and Y^U), and re-predicts labels for V^U using both self attributes and X_R. This process of feature computation and prediction is repeated, e.g., until convergence.

Finally, RCI uses all three kinds of features. Because it uses neighbor labels, it also must use some kind of collective inference such as ICA.

2.1 Prior Work with Neighbor Attributes

Some early work on link-based classification evaluated models that included neighbor attributes [2, 13]. However, recent work has used such models (including RI and RCI) very rarely for two primary reasons. First, prior work found that, while using neighbor labels can increase accuracy, using neighbor attributes can actually *decrease* accuracy [2].

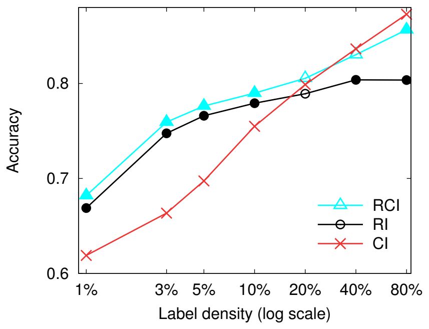

Figure 1: Accuracy for "Gene" using Naive Bayes and SSL-ONCE (see Section 5). Within a column, a circle or triangle that is filled in indicates a result that was significantly different (better or worse) than that of CI.

Later, Jensen et al. [3] compared CI vs. RI and RCI. They found that CI somewhat outperformed RCI and performed much better than RI. They describe how neighbor attributes greatly increased the parameter space, leading to lower accuracy, especially for small training graphs.

Second, it is unclear how to include neighbor attributes in popular classifiers. In particular, nodes usually have a varying number of neighbors. Thus, with neighbor attributes as features, there is no direct way to represent a node in a fixed-sized feature vector as expected by classifiers such as logistic regression or SVMs. With neighbor labels, CI algorithms address this issue with aggregation functions (such as "Count") that summarize all neighboring labels into a few feature values. This works well for labels, which are discrete and highly informative, but is more challenging for attributes, which are more numerous, may be continuous, and are individually less informative than a node's label (thus, this approach fared very poorly in early work [2]).

These two factors have produced a prevailing wisdom that CI based on neighbor labels is better than RI based on neighbor attributes (cf., [10, 9]). This conclusion rested on studies with fully-labeled training graphs, but has been carried into the important domain [11, 6] of sparsely-labeled graphs. In particular, Table 2 summarizes the models used by the most relevant prior work with such graphs. Only one study [15] used models with neighbor attributes (e.g., with RI or RCI), and it did not evaluate whether they were helpful.[1]

We next show that this prevailing wisdom was partly correct, but that, for sparsely-labeled networks, neighbor attributes are often more useful than previously thought.

3. THE IMPACT OF LABEL SPARSITY

To aid our discussion, we first preview our results. Figure 1 plots the average accuracy of RCI, RI, and CI for the Gene dataset (also used by Jensen et al. [3]). The x-axis varies the label density (e.g., fraction of nodes with known labels). When label density is high ($\geq 40\%$), CI significantly outperforms RI, and RCI as well when density is very high. High density is similar to learning with a fully-labeled graph, and these results are consistent with those of Jensen et al.

However, when density is low ($< 20\%$), CI is significantly worse than RI or RCI. To our knowledge, no prior work has reported this effect. Why does it occur?

[1]Prior work with neighbor attributes, including that of Xiang & Neville [15], used methods like decision trees or Naive Bayes that do not require a fixed-length feature vector.

Table 2: Related work on link-based classification that has used some variant of semi-supervised ICA (with partially-labeled networks). The first row is an exception; it used fully-labeled training networks but is included for comparison.

	Classifier type	RCI	RI	CI	SelfAttrs	NeighAttrs	NeighLabels	Inference alg.	Datasets used
Jensen et al. [3]	RBC	✓	✓	✓	✓			Gibbs	Gene, synthetic
Lu & Getoor [4]	Log. reg. (LR)			✓	✓			ICA	Cora, Citeseer
Xiang & Neville [15]	Decision tree	✓					✓	Soft ICA	Gene, synthetic
Bilgic et al. [1]	Log. reg. (LR)			✓	✓			ICA	Cora, Citeseer
Shi et al. [11]	Log. reg. (LR)			✓				ICA	Cora, Citeseer, Gene
McDowell & Aha [6]	Log. reg. (LR)			✓	✓		✓	ICA	Cora, Citeseer, Gene
This paper	NB & LR	✓	✓	✓	✓	✓	✓	ICA, Soft ICA, Gibbs	Cora, Citeseer, Gene, synthetic

First, during inference, many neighbor labels are unknown. Thus, a potential disadvantage of CI vs. RI is that some predicted labels used by CI will be incorrect, while the neighbor attributes used by RI are all known. However, prior work shows that CI—when learning uses a fully-labeled graph—can be effective even when *all* labels in a separate test graph are initially unknown [8, 7]. Thus, having a large number of unknown labels during CI's *inference*, while a drawback, is not enough to explain the substantial differences vs. RI.

The key problem is that, at low label density, CI struggles to *learn* the parameters related to label-based features. Labels can be used for learning such features only where *both* nodes of a link have known labels. In contrast, with neighbor attributes a single node's label makes a link useful for learning, since the node's neighbors' attributes are all known. Thus, with RI a single labeled node can provide multiple examples for learning (one for each of its links).

For example, for the Gene dataset of Figure 1, when the density is 10% (110 known nodes), CI can learn from an average of only 20 links, while RI can use an average of 340. Thus, for sparsely-labeled networks, the *effective training size for features based on neighbor attributes is much larger than for those based on neighbor labels.* This compensates for the greater number of parameters required by neighbor attributes, leading to higher accuracy.

Recent work with partially-labeled networks has also observed these problems with neighbor labels, but did not consider neighbor attributes. Instead, they avoid the problem by discarding methods based on label features and use latent features and/or links [12, 11]. Others have proposed non-learning methods that use label propagation or random walks [5]. Yet others have proposed SSL variants to first predict (noisy) labels for the entire network [15, 1, 6].

We later compare against some of these methods (e.g., with [5]). However, Figure 1's results *already* include some effective SSL methods [1, 6] (see Section 5.3), and yet substantial problems remain for CI at low density. Section 6.2 compares RCI, RI, and CI more completely and shows that neighbor attributes are helpful with or without SSL.

4. LEVERAGING NEIGHBOR ATTRIBUTES

This section explains existing methods for predicting with neighbor attributes, then introduces a new technique that enables the use of discriminative classifiers.

4.1 Existing Methods for Neighbor Attributes

Let $\mathcal{N}_i$ be the set of nodes that are adjacent to node v_i, i.e., in the "neighborhood" of v_i (for simplicity, we assume undirected links here). Furthermore, let $X_{\mathcal{N}_i}$ be the set of attribute vectors for all nodes in $\mathcal{N}_i$ ($X_{\mathcal{N}_i} = \{\vec{x_j} | v_j \in \mathcal{N}_i\}$).

Suppose we wish to predict the label y_i for v_i based on its attributes and the attributes of $\mathcal{N}_i$. As described above, the variable size of $\mathcal{N}_i$ presents a challenge. To address this general issue, prior studies (with neighbor labels) often assume that the labels of nodes in $\mathcal{N}_i$ are conditionally independent given y_i. This assumption is not necessarily true, but often works well in practice [3, 8, 7]. In our context (with neighbor attributes), we can make the analogous assumption that the attribute vectors of the nodes in $\mathcal{N}_i$ (and the attribute vector $\vec{x_i}$ of v_i itself) are conditionally independent given y_i. Using Bayes rule and this assumption yields

$$\begin{aligned} p(y_i|\vec{x_i}, X_{\mathcal{N}_i}) &= p(y_i)\frac{p(\vec{x_i}, X_{\mathcal{N}_i}|y_i)}{p(\vec{x_i}, X_{\mathcal{N}_i})} \\ &= p(y_i)\frac{p(\vec{x_i}|y_i)}{p(\vec{x_i}, X_{\mathcal{N}_i})}\prod_{v_j \in \mathcal{N}_i} p(\vec{x_j}|y_i) \\ &\propto p(y_i)p(\vec{x_i}|y_i)\prod_{v_j \in \mathcal{N}_i} p(\vec{x_j}|y_i) \qquad (1) \end{aligned}$$

where the last step drops all values independent of y_i.

To use Equation 1, we must compute $p(\vec{x_i}|y_i)$ and $p(\vec{x_j}|y_i)$. The same technique works for both; we now explain for the latter. We further assume that all attribute values for v_j (e.g., the values inside $\vec{x_j}$) are independent given y_i. If nodes have M attributes, then

$$\begin{aligned} p(\vec{x_j}|y_i) &= p(x_{j1}, x_{j2}, ..., x_{jM}|y_i) \\ &= \prod_{k=1}^{M} p(x_{jk}|y_i). \qquad (2) \end{aligned}$$

Plugging this equation (and the equivalent one for $p(\vec{x_i}|y_i)$) into Equation 1 yields a (relational) Naive Bayes classifier [8]. In particular, the features used to predict the label for v_i are v_i's attributes and v_i's neighbors' attributes, and these values are assumed to be conditionally independent. Jensen et al. [3] used this simple generative classifier for RI, and a simple extension to add the labels of neighboring nodes as features yields the equations needed for RCI.

4.2 Multi-neighbor Attribute Classification

The method described above can predict with neighbor attributes, and can increase classification accuracy, as we show later. However, it has two potential shortcomings. First, it ignores dependencies among the attributes within a single node. Second, it requires using probabilities like $p(\vec{x_j}|y_i)$, whereas a discriminative classifier (e.g., logistic regression) would compute $p(y_i|\vec{x_j})$. Thus, the set of classifiers that can be used is constrained, and overall accuracy may suffer.

A new idea is to take Equation 1 and apply Bayes rule to each conditional probability separately. This yields

$$p(y_i|\vec{x_i}, X_{\mathcal{N}_i}) \propto p(y_i)\frac{p(y_i|\vec{x_i})p(\vec{x_i})}{p(y_i)} \prod_{v_j \in \mathcal{N}_i} \frac{p(y_i|\vec{x_j})p(\vec{x_j})}{p(y_i)}$$

$$\propto p(y_i|\vec{x_i}) \prod_{v_j \in \mathcal{N}_i} \frac{p(y_i|\vec{x_j})}{p(y_i)} \quad (3)$$

where the last step drops all values independent of y_i.

We call classification based on Equation 3 *Multi-Neighbor Attribute Classification* (MNAC). This approach requires two conditional models, $p(y_i|\vec{x_i})$ and $p(y_i|\vec{x_j})$. The first is a standard (self)attribute-only classifier, while the second is more unusual: a classifier that predicts the label of a node based on the attributes of *one* of its neighbors. Because the prediction for this latter model is based on just a single attribute vector, any probabilistic classifier can be used, including discriminative classifiers such as logistic regression.

MNAC's derivation is simple, but it has not been previously used for link-based classification. McDowell & Aha [6] used a somewhat similar technique to produce "hybrid models" that combine two classifiers, but the derivation is different and they did not consider neighbor attributes.

5. EXPERIMENTAL METHOD

5.1 Datasets and Features

We use all of the real datasets used in prior studies with semi-supervised ICA, and some synthetic data (see Tables 2 & 3). We removed all nodes with no links.

Cora (cf., [10]) is a collection of machine learning papers. **Citeseer** (cf., [10]) is a collection of research papers. Attributes represent the presence of certain words, and links indicate citations. We mimic Bilgic et al. [1] by ignoring link direction, and also by using the 100 top attribute features after applying PCA to all nodes' attributes.

Gene (cf., [3]) describes the yeast genome at the protein level; links represent protein interactions. We mimic Xiang & Neville [15] and predict protein localization using four attributes: Phenotype, Class, Essential, and Chromosome. When using LR, we binarized these, yielding 54 attributes.

We create synthetic data using Sen et al.'s graph generator [10]. *Degree of homophily* is how likely a node is to link to another node with the same label; we use their defaults of 0.7 with a link density of 0.2. Each node has ten binary attributes with an *attribute predictiveness* of 0.6 (see [7]).

5.2 Classifiers and Regularization

All models shown in Table 1 (except NeighLabels) require learning a classifier to predict the label based on self attributes and/or a classifier to predict based on neighbor attributes. We evaluate Naive Bayes (NB), because of its past use with neighbor attributes [3], and logistic regression (LR), because it usually outperformed NB [10, 1]. For neighbor attributes, LR uses the new MNAC method.

RCI and CI also require a classifier to predict based on neighbor labels. McDowell & Aha [6] found that NB with "multiset" features was superior to LR with "proportion" features as used by Bilgic et al. [1]. Thus, we use NB for neighbor labels, and combine these results with the NB or LR classifiers used for attributes (described above), using the "hybrid model" method mentioned in Section 4.2 [6].

Table 3: Data sets summary.

Characteristics	**Cora**	**CiteSeer**	**Gene**	**Syn.**
Total nodes	2708	3312	1103	1000
Total links	5278	4536	1672	1250
Class labels	7	6	2	5
% dominant class	16%	21%	56%	20%

For sparsely-labeled data, regularization can have a large impact on accuracy. We used five-fold cross-validation on the labeled data, selecting the value of the regularization hyperparameter that maximized accuracy on the held-out labeled data. We used a Gaussian prior with all LR's features, a Dirichlet prior with NB's discrete features, and a Normal-Gamma prior with NB's continuous features.

5.3 Learning and Collective Inference

We chose default learning and inference algorithms that performed well in prior studies; Section 6.2 considers others.

For learning, we use the SSL-Once strategy: first, learn the classifiers using the attributes and the known labels. Next, run inference to predict labels for every "unknown" node. Finally, learn new classifiers, using the attributes, known labels, and newly predicted labels. With LR, we also use "label regularization" [6] which biases the learning towards models that yield sensible label distributions (it does not apply to NB). McDowell & Aha [6] found that these choices performed well overall and had consistent accuracy gains compared to not using SSL. The three baseline models (see Table 1) do not use SSL or label regularization.

For inference, CI and RCI use 10 iterations of ICA. For NeighLabels, we use a common baseline based on relaxation labeling (wvRN+RL [5], see Section 2).

6. RESULTS

We report accuracy averaged over 20 trials. For each, we randomly select some fraction of V (the "label density" d) to be "known" nodes V^K; those remaining form the "unknown label" test set V^U. We focus on the important sparsely-labeled case [11, 6], e.g., $d \leq 10\%$. To assess results, we use paired t-tests with a 5% significance level, with appropriate corrections because the 20 test sets are not disjoint [14].

6.1 Key Results with Real Datasets

Figure 2 shows average classification accuracy for the real datasets. The solid lines show results with RCI, RI, or CI, using NB as the attribute classifier. The results for Gene were already discussed in Section 3, and match closely the trends for Cora and Citeseer. In particular, for high label density d, CI (and sometimes RCI) performs best; but, when labels are sparse, using neighbor attributes (with RCI or RI) is essential. When $d < 10\%$, the gains of RCI and RI vs. CI are, with one exception, all significant (see caption). Results without SSL-Once (not shown) showed very similar trends.

With LR (see Table 4 for partial details), the trends are very similar: CI is significantly better than RI for high d, but the opposite holds for low d. Fortunately, RCI (using neighbor labels *and* attributes) yields good relative performance regardless of d, as was true with NB.

For comparison, the dashed lines in Figure 2 show results for RCI with LR (using MNAC). For Cora and Citeseer, this method beats RCI with NB in all cases. The gains are substantial (usually 8-10%) for very sparse graphs, and smaller but still significant for higher d. Here, LR outperforms NB

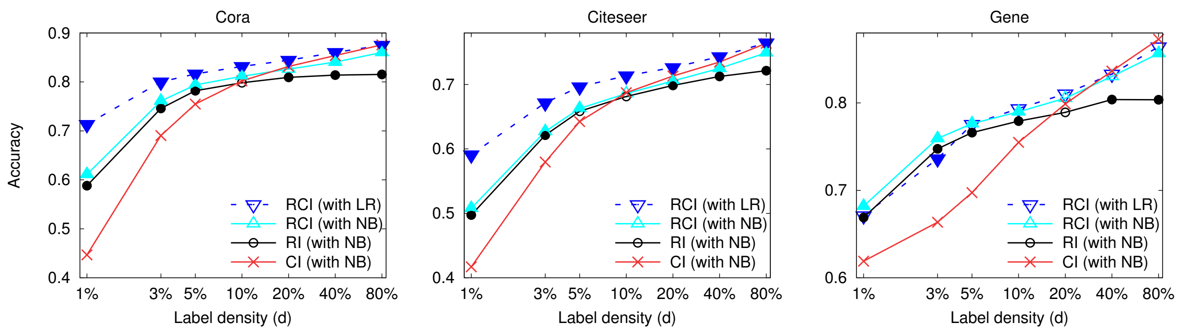

Figure 2: Accuracy vs. label density. Solid lines show results with NB; for these, filled (not hollow) symbols indicate sig. differences vs. CI. Dashed lines show RCI+LR for comparison; for these, filled symbols mean sig. differences vs. RCI+NB. These results (as with all others, unless otherwise stated) use SSL-Once and, where needed, ICA for inference.

Table 4: Average accuracy for different combinations of models, learning, and inference algorithms, using LR. Within each column, we bold the best result *and* all values that were *not* significantly different from that result. For $d > 10\%$ (not shown), learning and inference choices usually affected accuracy by 1% or less.

	Cora				Citeseer				Gene			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	1%	3%	5%	10%
RCI+No-SSL	58.1	74.7	78.9	**82.6**	51.5	63.7	67.1	**71.2**	63.4	70.6	75.7	**78.5**
RCI+SSL-Once	71.3	**80.0**	**81.6**	**83.2**	59.1	67.2	**69.7**	**71.4**	67.1	73.6	77.6	**79.4**
RCI+SSL-EM	**73.5**	**79.6**	**81.3**	82.7	**63.1**	**68.2**	**69.8**	**71.2**	**70.6**	**75.6**	**78.0**	**78.9**
RCI+SSL-Once+Gibbs	69.7	**79.7**	**81.3**	**83.0**	59.1	65.1	67.6	69.0	67.6	**74.2**	**78.2**	**79.8**
RI+No-SSL	57.4	73.1	77.1	80.6	51.2	63.1	66.8	70.5	63.1	68.8	73.6	76.3
RI+SSL-Once	67.8	78.7	80.5	81.8	57.9	66.7	**69.4**	**71.3**	65.9	71.6	75.7	78.0
RI+SSL-EM	**70.2**	78.9	80.2	81.3	**62.9**	**68.4**	**69.7**	**71.2**	68.3	71.4	76.4	76.8
CI+No-SSL	40.9	63.1	70.6	79.5	44.3	59.5	63.4	69.6	60.5	67.8	72.2	75.7
CI+SSL-Once	57.2	75.1	78.3	81.3	52.5	64.2	67.7	69.6	60.9	66.7	70.6	74.2
CI+SSL-EM	64.6	78.2	80.2	82.0	57.4	65.8	68.0	69.6	62.8	67.0	75.3	77.2
CI+SSL-Once+Gibbs	36.6	62.8	71.2	78.0	44.3	54.8	53.2	60.8	61.0	67.0	70.4	74.8
SelfAttrs	37.1	54.1	59.5	65.8	40.7	55.8	61.8	66.7	59.8	65.5	68.4	71.6
NeighAttrs	52.1	70.9	74.6	77.8	45.1	58.4	63.5	65.8	59.4	65.9	69.5	71.8
NeighLabels	41.2	63.4	72.3	77.9	31.5	47.9	52.6	55.8	55.7	62.6	66.6	71.6

because the attributes are continuous (a challenging scenario for NB) and because LR does not assume that the (many) attributes are independent. For Gene, however, NB is generally better, likely because NB can use the 4 discrete attributes, whereas LR must use the 54 binarized attributes.

Overall, for sparsely-labeled graphs, neighbor attributes appear to be much more useful than previously recognized, and our new LR with MNAC sometimes significantly increases accuracy. For simplicity, below we only consider LR.

6.2 Varying Learning & Inference Algorithms

The results above all used SSL-Once for learning and, where needed, ICA for collective inference. Table 4 examines results where we vary these choices. First, there are different learning variants: No-SSL, where SSL is not used (though label regularization still is), and SSL-EM, where SSL-Once is repeated 10 times, as with McDowell & Aha [6]. Second, for RCI and CI we consider Gibbs sampling instead of ICA. Gibbs has been frequently used, including in the RI vs. CI comparisons of Jensen et al. [3], but sometimes has erratic behavior [7].

These choices can impact accuracy. For instance, SSL-Once+Gibbs sometimes beats SSL-Once with ICA, but always by less than 1%, and decreases accuracy substantially in several cases with CI. Using SSL-EM instead of SSL-Once leads to more consistent gains that are sometimes substantial, especially for CI.

Thus, for different datasets and classifiers, the best SSL and inference methods will vary. However, our default use of ICA with SSL-Once was rarely far from maximal when RCI, the best model, was used. Moreover, the values in bold show that, regardless of the learning and inference choices, RCI or RI yielded the best overall accuracy (at least for the data of Table 4, which focuses on sparse networks). Also, using RCI or RI with SSL-Once almost always outperformed CI with *any* learning/inference combination shown.

For comparison, the bottom of Table 4 also shows results with three baseline algorithms. NeighAttrs performs best on average, indicating the predictive value of neighbor attributes, even when used alone.

6.3 Varying the Data Characteristics

We compared RCI, RI, and CI on the synthetic data. First, varying the label density produced results (not shown) very similar to Gene in Figure 2. Second, in Figure 3(a), as homophily increases, both RI and CI improve by exploiting greater link-based correlations. RCI does even better by using both kinds of link features, except at low homophily where using the (uninformative) links decreases accuracy.

Figure 3(b) shows the impact of adding some random attributes for each node. We expected CI's relative performance to improve as these were added, based on the results and argument of Jensen et al. [3]: CI has fewer parameters than RI, and thus should suffer less from high variance due to the random attributes. Instead, we found that RI's (and RCI's) gains over CI only increase, for two reasons.

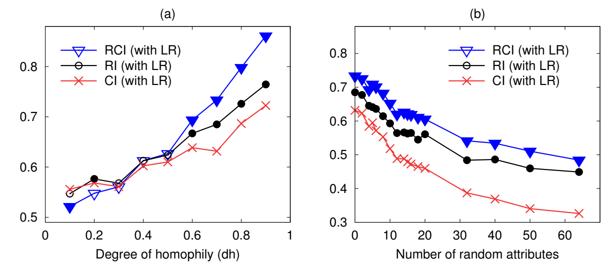

Figure 3: Synthetic data results, using LR and 5% label density. Filled symbols indicate sig. differences vs. CI.

First, Jensen et al. had fully-labeled training data; for our sparse setting, RI has a larger effective training size (see Section 3), reducing variance. Second, unlike Jensen et al. we use cross-validation to select regularization parameters for the features. The regularization reduces variance for RI and CI, and is especially helpful for RI as random attributes are added. If we remove regularization and increase label density, the differences between CI and RI decrease markedly.

6.4 Additional Experiments

Due to lack of space, we can only summarize two additional experiments. First, we evaluated "soft ICA", where continuous label estimates are retained and used at each step of ICA instead of choosing the most likely label for each node [15]. This slightly increased accuracy in some cases, but did not change the relative performance of RCI and RI vs. CI. Second, we compared our results (using a single partially-labeled graph) vs. "ideal" results obtained using a fully-labeled learning graph *or* a fully-labeled inference graph. We found that ideal *learning* influenced results much more than ideal *inference* for all methods, but that CI was (negatively) affected by realistic (non-ideal) learning much more than RI or RCI were. Thus, as argued in Section 3, RI's and RCI's gains vs. CI for sparsely-labeled networks seem to arise more because of CI's difficulties with learning rather than with inference.

7. CONCLUSION

Link-based classification is an important task, for which the most common methods involve computing relational features. Almost no recent work has considered neighbor attributes for such features, because prior work showed they performed poorly and they were not compatible with many classifiers. We showed, however, that for sparsely-labeled graphs using neighbor attributes (with RI) often significantly outperformed neighbor labels (with CI), and that using both (with RCI) yielded high accuracy regardless of label density. We also introduced a new method, MNAC, which enables classifiers like logistic regression to use neighbor attributes, further increasing accuracy for some datasets. Naturally, the best classifier depends upon data characteristics; MNAC greatly expands the set of possibilities.

Our findings should encourage future researchers to consider neighbor attributes in models for link-based classification, to include RI and RCI as baselines in comparisons, and to re-evaluate some earlier work that did not consider such models (see Table 2). For instance, Bilgic et al. [1] studied active learning with sparse graphs; their optimal strategies could be quite different if RI or RCI were considered, since they could tolerate learning with fewer and more widely-dispersed labels. Likewise, Shi et al. [11] used semi-supervised ICA with CI as a baseline, but these results could change substantially with RCI instead.

Our results need to be confirmed with additional datasets and learning algorithms. Also, the best methods should be compared against others discussed in Section 2. Finally, we intend to explore these effects in networks that include nodes with different types and some missing attribute values.

8. ACKNOWLEDGMENTS

Thanks to Gavin Taylor and the anonymous referees for comments that aided this work. This work was supported in part by NSF award number 1116439 and a grant from ONR.

9. REFERENCES

[1] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *Proc. of ICML*, pages 79–86, 2010.

[2] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of SIGMOD*, pages 307–318, 1998.

[3] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proc. of KDD*, pages 593–598, 2004.

[4] Q. Lu and L. Getoor. Link-based classification using labeled and unlabeled data. In *ICML Workshop on the Continuum from Labeled to Unlabeled data*, 2003.

[5] S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *J. of Machine Learning Research*, 8:935–983, 2007.

[6] L. McDowell and D. Aha. Semi-supervised collective classification via hybrid label regularization. In *Proc. of ICML*, pages 975–982, 2012.

[7] L. McDowell, K. Gupta, and D. Aha. Cautious collective classification. *J. of Machine Learning Research*, 10:2777–2836, 2009.

[8] J. Neville and D. Jensen. Relational dependency networks. *J. of Machine Learning Research*, 8:653–692, 2007.

[9] R. Rossi, L. McDowell, D. Aha, and J. Neville. Transforming graph data for statistical relational learning. *J. of Artificial Intelligence Research*, 45:363–441, 2012.

[10] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[11] X. Shi, Y. Li, and P. Yu. Collective prediction with latent graphs. In *Proc. of CIKM*, pages 1127–1136, 2011.

[12] L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proc. of KDD*, pages 817–826, 2009.

[13] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *Proc. of IJCAI*, pages 870–878, 2001.

[14] T. Wang, J. Neville, B. Gallagher, and T. Eliassi-Rad. Correcting bias in statistical tests for network classifier evaluation. In *Proc. of ECML*, pages 506–521, 2011.

[15] R. Xiang and J. Neville. Pseudolikelihood EM for within-network relational learning. In *Proc. of ICDM*, pages 1103–1108, 2008.