Using Ghost Edges for Classification in Sparsely Labeled Networks

Brian Gallagher† Hanghang Tong⋆ Tina Eliassi-Rad† Christos Faloutsos⋆
†Lawrence Livermore National Laboratory
†{bgallagher, eliassi}@llnl.gov
⋆Carnegie Mellon University
⋆{htong, christos}@cs.cmu.edu

ABSTRACT

We address the problem of classification in partially labeled networks (a.k.a. within-network classification) where observed class labels are sparse. Techniques for statistical relational learning have been shown to perform well on network classification tasks by exploiting dependencies between class labels of neighboring nodes. However, relational classifiers can fail when unlabeled nodes have too few labeled neighbors to support learning (during training phase) and/or inference (during testing phase). This situation arises in real-world problems when observed labels are sparse.

In this paper, we propose a novel approach to within-network classification that combines aspects of statistical relational learning and semi-supervised learning to improve classification performance in sparse networks. Our approach works by adding "ghost edges" to a network, which enable the flow of information from labeled to unlabeled nodes. Through experiments on real-world data sets, we demonstrate that our approach performs well across a range of conditions where existing approaches, such as collective classification and semi-supervised learning, fail. On all tasks, our approach improves area under the ROC curve (AUC) by up to 15 points over existing approaches. Furthermore, we demonstrate that our approach runs in time proportional to $L \cdot E$, where L is the number of labeled nodes and E is the number of edges.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining; I.2.6 [**Artificial Intelligence**]: Learning; I.5.1 [**Pattern Recognition**]: Models - Statistical

General Terms

Algorithms, Design, Performance, Experimentation.

Keywords

Statistical relational learning, semi-supervised learning, collective classification, random walk.

1. INTRODUCTION

We address the problem of within-network classification in sparsely labeled networks. Given a network of both labeled and unlabeled nodes, our goal is to provide labels for the unlabeled nodes. Here, labeling simply means assigning each node a class from among a set of possible classes (see Figure 1).

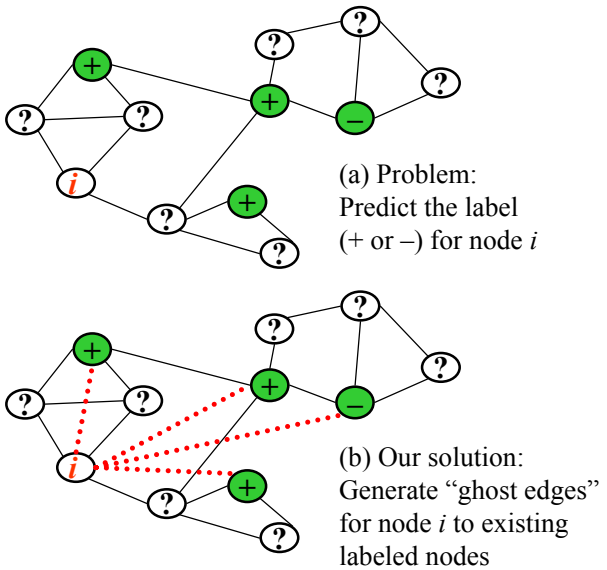

Figure 1: Problem definition and our solution. We address the problem of label sparsity in network classification by creating "ghost edges" using random walks with restarts. These ghost edges connect an unlabeled node to the most relevant labeled nodes.

Suppose we want to identify fraudulent users of a cell phone network. In this case, our set of possible classes for users is {fraud, legit}. If a user is known to be involved in fraud, we assign a label of "fraud." If a user is known to not be involved in fraud, we assign a label of "legit." Otherwise, we leave the node unlabeled. Our goal is then to assign labels ("fraud" or "legit") to the unlabeled users.

Cell phone fraud is an example where networks are often very sparsely labeled. We have a handful of known fraudsters and legitimate users, but the labels are unknown for the vast majority of users. For such applications, it is reasonable to expect that we have access to labels for fewer than 10%, 5%, or even 1% of the nodes.

In addition to being sparsely labeled, cell phone networks are generally anonymized. That is, the nodes often contain no attributes besides class labels. These sparsely labeled, anonymized networks are the focus of this study. Put another way, our work focuses

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.

on *univariate within-network classification in sparsely labeled networks*.

Techniques for statistical relational learning (SRL) have been shown to perform well on network classification tasks because of their ability to exploit dependencies between labels of related nodes [18]. These techniques use labeled data to learn a model of the dependencies between labels of neighboring nodes. The labels of unlabeled nodes can then be inferred by propagating information from labeled nodes throughout the network, according to this learned dependency model (i.e., collective classification). Unfortunately, sparse labels cause a number of problems for relational classifiers. First, collective classification techniques can fail without sufficiently many labels to seed the inference process [15]. Second, fewer labeled nodes means fewer training examples from which to learn dependencies. Finally, even when a node is labeled, many of its neighbors will not be. This is equivalent to having many missing attribute values in a traditional supervised learning setting. The result is that it is very difficult to learn an accurate model of the dependencies present in the data.

Within-network classification can also be viewed as a graph-based semi-supervised learning problem since we have both labeled and unlabeled data available at training time. Although semi-supervised learning (SSL) is generally applied to non-network data, graph-based SSL approaches can be applied to within-network classification, as we show. SSL techniques have the advantage that they do not rely as heavily on labeled training data and can make use of unlabeled data as well. However, graph-based SSL approaches generally do not learn dependencies from data, but instead assume local label consistency (i.e., that nearby points tend to have the same label). If the label consistency assumption is not met, SSL techniques can perform extremely poorly. Figure 2 provides a preview of our results. It plots classifier performance (measured by area under the ROC curve, a.k.a. AUC) versus data set ID. The data sets are ordered according to their local consistency (a.k.a. homophily score).[1] Notice that the two proposed ghostEdge methods are consistently at the top, while the competing methods may perform very poorly depending on the data set's local consistency.

In this paper, we explore a novel approach to within-network classification that capitalizes on the strengths of both statistical relational learning (SRL) and semi-supervised learning (SSL), to produce a classifier that is robust in the face of both sparse labeling and low label consistency. Our contributions are as follows:

- We propose a novel approach to within-network classification, based on the creation of "ghost edges" that enable the propagation of information from labeled to unlabeled nodes.
- We demonstrate that our approach is robust to both sparse labeling and low label consistency, performing well consistently across a range of real world classification tasks where collective classification or semi-supervised learning fail.
- We demonstrate the scalability of our approach. Specifically, we show that our methods run in time proportional to $L \cdot E$ where L is the number of labeled nodes and E is the number of edges in our graph.

The rest of the paper is organized as follows. In Section 2, we review related work. We present our proposed methods in Section 3. Sections 4 and 5 describe our experimental methodology and results. Finally, we offer conclusions in Section 6.

[1]Note that throughout this paper, we use the terms 'homophily', 'positive auto-correlation', 'local consistency' and 'smoothness' interchangeably. We also use the terms 'inverse relationship', 'negative auto-correlation' and 'lack of homophily' interchangeably.

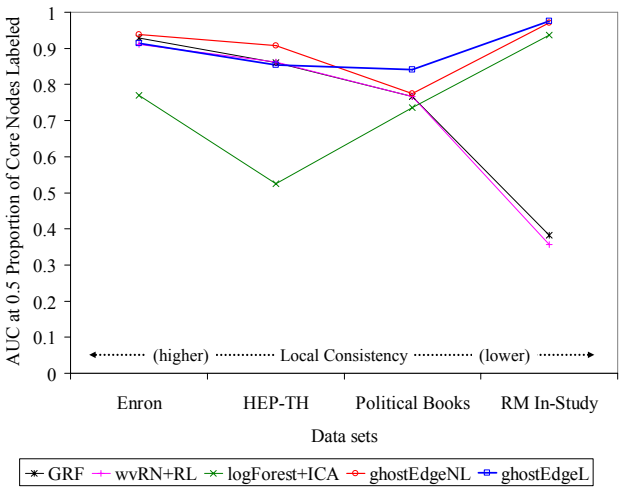

Figure 2: A preview of our results: classification performance of various approaches when half of the nodes in a graph data set are unlabeled. The solid blue (square icon) and red (circle icon) lines on the top represent our ghostEdge approaches, which perform consistently well regardless of the degree of local consistency in the data set.

2. RELATED WORK

In recent years, there has been a great deal of work on models for learning and inference in relational data [7, 10, 11, 16, 18]. For within-network classification tasks where we have sparse labels, we categorize the previous work into two main groups: collective classification and graph-based semi-supervised learning.

Collective classification. Collective classification deals with label sparsity by simultaneously labelling a set of related nodes, allowing estimates of neighboring labels to influence one another. Representative work in this line started with the seminal paper of Chakrabarti et al. on using hyperlinks to for hypertext classification [2]. Sen et al. [17] provide a careful empirical study of the various procedures for collective inference. Macskassy and Provost [12] provide a nice case-study of previous work in learning attributes of networked data. McDowell et al. [14] demonstrate that "cautious" collective classification procedures produce better classification performance than "aggressive" ones. They recommend only propagating information about the top-k most confidently predicted labels. One of the major advantages of collective classification lies in its powerful ability to learn various kinds of dependency structures (positive vs. negative auto-correlation, different degrees of correlation, etc). However, as pointed out in [15], when the labeled data is very sparse, which is quite common in the sparsely labeled networks that we are particularly interested in, the performance of collective classification might be largely degraded due to the lack of sufficient neighbors. This is exactly one major advantage of the proposed method, - we incorporate informative "ghost edges" into the networks to deal with sparsity issues. From this point of view, our method shares the similar spirit as the work by Macskassy [13]. However, in [13], the additional edges are calculated based on attribute-similarity (specifically, text similarity). If such information is not available (which is quite common in many real applications and which is the case we are interested in), the method in [13] is not applicable. On the other hand, we can always leverage our "ghost edges" since they are based on the intrinsic

structure of the networks. Lastly, the algorithm proposed in [13] does not learn the weights, instead it combines the weights through a heuristic.

Graph-based semi-supervised learning. The problem of within-network classification can also be thought of as the graph-based semi-supervised learning problem. Here, the basic idea is to estimate a function on the graph which satisfies two kinds of constrains: (1) the consistency with the label information and (2) the smoothness over the whole graph. The methods in this area mainly vary in the different ways to balance these two constraints. For example, Zhu's Gaussian random field (GRF) method [23] puts a hard constraint on the label consistency and then achieves the smoothness by the harmonic function. Zhou's global and local consistency method [21] combines the two kinds constraints by a regularization parameter and solves a quadratic optimization problem. For more on graph-based semi-supervised learning, we refer the reader to an excellent survey by Zhu [22]. By exploring the global structure (i.e. smoothness) over the whole graph, graph-based semi-supervised learning methods usually outperform the traditional methods, particularly when there are very few labeled nodes in the networks. However, the constraint on smoothness implicitly assumes positive auto-correlation in the graph, that is nearby nodes tend to share the same class labels (i.e., homophily). When such an underlying assumption does not hold (negative auto-correlation, the degree of auto-correlation being small, etc), the performance might be largely affected. This is another advantage of our "ghost edge" method, - it leverages the additional learning stage to recover the intrinsic correlation structure.

3. PROPOSED METHOD

In this section, we explain the motivation behind using ghost edges for within-network classification and discuss how ghost edges are created. We then describe how we take advantage of ghost edges to improve classification performance.

3.1 Motivation and General Approach

The power of statistical relational learning (SRL) lies in the fact that networks generally exhibit predictable relationships between class labels of related nodes. Therefore, labeled nodes provide a great deal of information about their unlabeled neighbors. Suppose we have an unlabeled node, u, and we have a good understanding of the relationship between the class label of u and the class labels of u's neighbors, N. If all nodes in N are labeled, we should be able to do a very good job of predicting the label of u. However, suppose that very few nodes in N are labeled. There are two ways of looking at this problem:

1. Node u shares edges with plenty of other nodes in the network, but too few of those neighboring nodes are labeled.

2. There are plenty of labeled nodes in the network, but node u shares edges with too few of them.

The first way of looking at the problem is addressed by techniques such as collective classification. Our approach, based on *ghost edges*, addresses the second way of looking at the problem.

The idea behind our approach is to add "ghost edges" between labeled nodes and unlabeled nodes to allow the information from labeled nodes to inform our predictions on unlabeled nodes. Of course, in order for this to work, we must carefully select pairs of nodes to connect via ghost edges. In particular, the success of our approach relies on our ability to choose pairs of nodes with labels that relate to each other in a predictable way. Our conjecture is that nodes which are "closer" in a network will tend to have more predictable relationships between their class labels, and that nodes which are directly connected by an observed edge are simply a special case of this.

Based on this conjecture, we create ghost edges as follows. We create a single ghost edge between every ⟨labeled, unlabeled⟩ pair of nodes in our graph. We then assign a weight to each ghost edge based on the proximity of the nodes that the edge connects. A higher weight indicates that the connected nodes are closer together in the network (i.e., have higher proximity). The following subsection describes our approach to quantifying node proximity.

3.2 Quantifying Node Proximity Using Random Walk with Restart

To calculate RWR scores between each pair of nodes, we use a variation on the fast random walk with restart method proposed by Tong *et al.* [19]. The end result is that the algorithm can quickly give the proximity score $r_{i,j}$, indicating how easy it is to reach node j from node i. In more detail, $r_{i,j}$ gives the steady-state probability to find a particle at node j, when this particle does a random walk with restarts from node i. The score $r_{i,j}$ is high if there are several, high-weighted, short paths from i to j. A random walk with restarts (or, equivalently, *personalized pageRank*) works as follows: a particle starts at node i, moves randomly along the edges of the given graph, and with probability $1 - c$ (say, $c = 0.90$), the particle flies back to the initial node i. For more details, see [19].

3.3 Handling Degrees of Label Consistency

There is a subtle, but very important step in our RWR algorithm. The problem is to handle varying degrees of local label consistency, including cases where labels of neighbors are inversely related. In cases with high label consistency (i.e., nearby nodes have similar labels), all methods tend to do well. In cases where consistency is low, as in, say, a near-bipartite graph of dating relationships with mostly male-female edges, the semi-supervised learning methods will not work, exactly because they have the local consistency assumption hardwired in their optimization functions.

Similarly, if we add ghost edges carelessly, we will add a lot of incorrect edges. What are the right edges to add so that our method can easily handle cases with varying degrees of label consistency?

The idea is to do RWR, but to insist on *even*-length paths. We shall refer to it as the *even-step RWR*. Mathematically, this means that we replace the adjacency matrix $\mathbf{A}$ with its square $B = A * A$. Then, we compute the RWR scores using the B matrix.

Why does this approach work regardless of the degree of label consistency? The reasons are subtle: For the case of inverse class-label relationship, the immediate neighbors are exactly the ones we want to avoid, which is exactly what the even-step RWR does. For the case of high local consistency, social networks typically have triangles and high "clustering coefficient." Thus, even if we only focus on even step paths, our random walk will still give high scores to nodes that are well-connected. In practice, we find that the even-step approach works well for intermediate consistency values as well.

In conclusion, with the subtle technique of *even-step RWR*, our GhostEdge methods can handle varying degrees of local label consistency, as we show in the experiments section.

3.4 Classifier Design

We propose two novel approaches to within-network classification: the ghost edge non-learning classifier (*ghostEdgeNL*) and the ghost edge learning classifier (*ghostEdgeL*). Both approaches are based on propagating class labels throughout a network using ghost edges created via random walk with restart (namely, even-

step RWR). However, the two classifiers make use of ghost edges in different ways. There are two ways in which the approaches differ:

1. *Use of available labels*. GhostEdgeNL is a non-learning method. It simply assumes that neighbors connected by a ghost edge will tend have the same class labels and that this tendency is stronger across edges with higher weights. GhostEdgeL, on the other hand, uses the labeled nodes in a network as training examples to learn the dependencies between class labels of both observed neighbors and ghost neighbors.

2. *Use of RWR scores*. GhostEdgeNL makes use of all ghost edges, although it puts more weight on edges with higher proximity scores. GhostEdgeL bins ghost edges based on their proximity scores and then uses labeled data to learn weights on each bin, based on the predictiveness of its edges.

3.4.1 The GhostEdgeNL Classifier

Ghost edges can be added to any relational classifier. For GhostEdgeNL, we chose to use a simple relational neighbor classifier [11]. This classifier predicts the class of a node based entirely on the class labels of neighboring nodes and performs no learning. It estimates the probability of node u belonging to class c as the weighted proportion of neighboring nodes that belong to class c. GhostEdgeNL uses the proximity score on the ghost edge between nodes as a weight. GhostEdgeNL ignores observed edges.

3.4.2 The GhostEdgeL Classifier

For GhostEdgeL, we chose a learning link-based classifier [10]. This classifier uses logistic regression (LR) to build a discriminative model of node i's class given the class labels of nodes directly connected to i. Since LR expects a fixed-length feature vector, the set of neighboring class labels is summarized by a statistic such as the count or proportion of neighbors of each class.

We initially implemented GhostEdgeL using a standard LR model. However, we found that LR often failed to appropriately weight features based on their predictiveness. We achieved better results using an ensemble of LR models we refer to as *logForest*. The logForest model is inspired by Breiman's Random Forest classifier [1]. We use a bag of LR classifiers, where each is given a subset of $log(M) + 1$ of the M total features. For this study, our logForest model uses 500 LR classifiers.

GhostEdgeL divides ghost edges into six bin as follows: A contains ghost edges with scores in the top 3%, B gets edges scoring between the top 3%-6%, C between 6%-12%, D between 12%-25%, E between 25%-40%, and F between 40%-80%. There is no overlap between bins.

The GhostEdgeL classifier uses the following features: (1) count of neighbors of each class across observed edges and (2) count of neighbors of each class across ghost edges for each bin. So, for a binary classification problem with six bins, we have 14 features.

Like any relational learning method, GhostEdgeL learns the dependencies between class labels of neighboring nodes. However, in addition, the model learns how much weight to put on observed edges vs. ghost edges with different proximity scores, and the model can potentially learn different dependencies for each of these edge types.

3.5 Scalability

In even-step RWR, the ranking vector $\vec{r}_i = [r_{i,j}]$ for a given labeled node i is defined as:

$$\vec{r}_i(t+1) = c\mathbf{A}^2\vec{r}_i(t) + (1-c)\vec{e}_i$$

where $\vec{e}_i$ is the starting vector for the node i, c is the fly-out probability, t is the iteration number, and $\mathbf{A}$ is the normalized graph Laplacian [3].[2]

We can use the following iterative strategy:

$$\vec{r}_i(1) = \vec{e}_i \tag{1}$$

For $t = 2, 3, ...$, do the following two steps:

$$\vec{r}_i(t) = \mathbf{A}\vec{r}_i(t-1) \tag{2}$$

$$\vec{r}_i(t) \leftarrow c\mathbf{A}\vec{r}_i(t) + (1-c)\vec{e}_i \tag{3}$$

The complexity for each step t is clearly $O(E)$. So, to get one ranking vector, the complexity is $O(\tilde{t} \cdot E)$, where $\tilde{t}$ is the maximum number of iteration needed to reach the steady state. Overall, we need L such ranking vectors (so that we will get all $U \times L$ proximity scores). Therefore, the overall complexity is $O(L \cdot E)$ (omitting the constant $\tilde{t}$).

Next, we will justify that the above iterative procedure will actually converge. To see this, we can re-write $\vec{r}_i(t)$ as:

$$\vec{r}_i(t) = \sum_{t=1}^{\infty}(c^t)\mathbf{A}^{(2t)}\vec{e}_i$$

Since $\mathbf{A}$ is normalized graph Laplacian, we have $-1 \leq \lambda(A) \leq 1$, where $\lambda(A)$ is the eigenvalue of $\mathbf{A}$ [3]. Therefore, $\mathbf{A}^{(2t)}$ is bounded. On the other hand, $c^t \rightarrow 0$ with $t \rightarrow infinity$. Thus, $\|(c^t)\mathbf{A}^{(2t)}\vec{e}_i\|$ goes to 0 as t goes to infinity, which completes the proof.

4. EXPERIMENTAL DESIGN

Our problem setting is within-network classification in sparsely labeled networks. We compare several approaches to overcome label sparsity: (1) collective classification, (2) graph-based semi-supervised learning methods, and (3) our "ghost edge" label propagation approach (ghostEdge). The experiments are designed to answer the following research questions:

- How do the proposed GhostEdge methods do against the competition?
- What is the impact of local label consistency (i.e., homophily) and lack of it?

4.1 Data Sets

We present results on four real-world data sets: political book co-purchases [9], Enron emails [4], Reality Mining cell-phone calls [5], and high-energy physics citations from arXiv (a.k.a. HEP-TH [8]). Our tasks are to identify neutral political books, Enron executives, Reality Mining study participants, and HEP-TH papers with the topic "Differential Geometry," respectively.

Figure 3 summarizes our prediction tasks. The *Sample* column describes the method used to obtain our experimental subset of data from the full data set: use the entire set (*full*), use a time-slice (*time*), or sample a continuous subgraph via breadth-first search (*BFS*). The *Task* column indicates the class label we are trying to predict. The $|V|$, $|L|$, and $|E|$ columns indicate counts of total nodes, labeled nodes, and total edges in each network, respectively. The $P(+)$ column indicates the proportion of labeled nodes that have the class label of interest (e.g., 12% of the political books are neutral).

[2]The normalized graph Laplacian $\mathbf{A} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is the identity matrix, $\mathbf{D}$ is the diagonal degree matrix, and $\mathbf{W}$ is the weighted adjacency matrix.

Note that for the Enron and HEP-TH tasks we have labels for only a subset of nodes (which we refer to as "core" nodes) and can only train and test our classifiers on these nodes. However, unlabeled nodes and their connections to labeled nodes may still be exploited for label propagation.

Data Set	Sample	Task	$\|V\|$	$\|L\|$	$\|E\|$	P(+)
Enron	Full	Executives?	3081	1055	34902	0.02
HEP-TH	Time	Diff. Geometry?	2999	342	36014	0.06
Political Books	BFS	Neutral?	105	105	441	0.12
Reality Mining	BFS	In Study?	1000	1000	31509	0.08

Figure 3: Summary of data sets and prediction tasks.

4.2 Competing methods

From our methods, we use the two versions: (a) **ghostEdgeNL** which is the non-learning ghostEdge-based approach as described in Section 3.4.1 and (b) **ghostEdgeL** which is the learning ghostEdge-based approach described in Section 3.4.2.

The competing methods fall under two categories: (1) collective classification, and (2) graph-based semi-supervised learning methods. Both categories of methods were designed to handle label sparsity.

4.2.1 Competing methods

On each classification task, we ran seven individual classifiers:

1. **logForest**, an ensemble logistic link-based model without collective classification
2. **logForest+ICA**, an ensemble logistic link-based model, which uses the iterative classification algorithm to perform collective classification
3. **wvRN**, a relational neighbor model without collective classification
4. **wvRN+RL**, a relational neighbor model, which uses relaxation labeling for collective classification
5. **GRF**, the SSL Gaussian random field model
6. **ghostEdgeNL**, our ghostEdge-based classifier without learning
7. **ghostEdgeL**, our ghostEdge-based classifier with learning

We describe each of the competing classifiers next.

logForest is an ensemble of logistic regression classifiers as described in Section 3.4.2. The model takes two features as input: the count of unique neighbors of the positive class and the count of unique neighbors of the negative class. Our base **logForest** classifier does not use collective classification. Therefore, any neighbors with missing class labels are simply ignored.

logForest+ICA uses the base **logForest** classifier, but performs collective classification using the ICA algorithm described in Section 4.2.2. We tried the logForest classifier with both the ICA and RL collective classification algorithms across our range of classification tasks. The performances of the two algorithms were comparable, but ICA performed slightly better overall. This is consistent with previous results [12].

wvRN is the weighted-vote relational neighbor classifier [11]. Given a node i and a set of neighboring nodes, N, the wvRN classifier calculates the probability of each class c for node n as:

$$P(c \mid n) = \frac{1}{Z} \sum_{\{n_j \in N | label(n_j)=c\}} w(n, n_j) \quad (4)$$

where N is the set nodes that neighbor n, $Z = \sum_{n_i \in N} w(n, n_i)$, and $w(n, m)$ is the weight on the edge between n and m. For the baseline wvRN model, $w(n, m)$ is simply the number of observed edges between n and m.

Note that in cases where a node has no labeled neighbors, we will end up with $P(C_i = c) = 0$ for all c. In such cases, we simply assign probabilities to each class based on priors observed in the training data. Our base wvRN classifier does not use collective classification. Therefore, any neighbors with missing class labels are simply ignored.

wvRN+RL uses the base wvRN classifier, but performs collective classification using the RL algorithm described in Section 4.2.2. We tried the wvRN classifier with both the ICA and RL collective classification algorithms across our range of classification tasks. The RL algorithm performed better overall. This is consistent with previous results [12].

GRF uses the Gaussian random field approach of Zhu *et al.* [23]. We ported Zhu's MATLAB code[3] for use in our experimental framework and double checked our results with the original MATLAB code. We made one small modification to Zhu's original code to allow it to handle disconnected graphs. Zhu computes the graph Laplacian as $L = D - cW$, where $c = 1$. We set $c = 0.9$ to ensure that L is diagonally dominant and thus invertible. We found that our change had no substantial impact on classification performance.

4.2.2 Collective Classification

To perform collective classification, we use both the iterative classification algorithm (ICA) and relaxation labeling (RL) [12]. We also ran preliminary experiments using Gibbs sampling [6], which yielded results comparable to ICA. This is consistent with findings of other researchers [12, 17]. In our experiments, the logForest classifier performed better overall using ICA and the wvRN classifier performed better using RL. Therefore, we report results only for these combinations.

ICA initially assigns labels to unlabeled nodes, U, based on what is known in each unlabeled node's local neighborhood. Nodes with no labeled neighbors are temporarily assigned a label of $null$. Then, until either all class labels have stabilized or a certain number of iterations have elapsed, a new label is assigned to each $u_i \in U$, based on the current label assignments of u_i's neighbors. RL is similar to ICA except that instead of each u_i having a current label assignment, u_i has a current probability distribution on the set of labels. Thus, the uncertainty associated with a label assignment is retained until the algorithm terminates and a final label is assigned. Unlabeled nodes are initially assigned the prior distribution, observed in the training data. We perform simulated annealing to catalyze convergence.

4.3 Experimental Methodology

For all results presented here, the basic experimental setup is the same. Each data set contains a set of core nodes for which we have ground truth (i.e., we know the true class labels). In all cases, classifiers have access to the entire data graph during both training and testing. However, not all of the core nodes are labeled. We vary

[3] See http://pages.cs.wisc.edu/ jerryzhu/pub/harmonic_function.m.

the proportion of labeled core nodes from 10% to 90%. Classifiers are trained on all labeled core nodes and evaluated on all unlabeled core nodes.

Our methodology is as follows. For each proportion of core nodes labeled, we run 20 trials and report the average performance. For each trial and proportion labeled, we choose a class-stratified random sample containing (1.0 − proportion labeled)% of the core instances as the test set and the remaining core instances become the training set. Note that for proportion labeled less than 0.9 (or greater than 10 trials), this means that a single instance will necessarily appear in multiple test sets. The test sets cannot be made to be independent because of this overlap. However, we carefully choose the test sets to ensure that each instance in our data set occurs in the same number of test sets over the course of our experiments. This ensures that each instance carries the same weight in the overall evaluation regardless of the proportion labeled. Labels are kept on the training instances and removed from the test instances. We use identical train/test splits for each classifier.

Our experimental framework sits on top of the open source Weka system [20]. We implement our own network data representation and experimental code, which handles tasks such as splitting the data into training and test sets, labeling and unlabeling of data, and converting network fragments into a Weka-compatible form. We rely on Weka for the implementation of logistic regression and for measuring classifier performance on individual training/test trials.

We use the area under the Receiver Operating Characteristic (ROC) curve (AUC) as a performance measure to compare classifiers. We chose AUC because it is more discriminating than accuracy. In particular, most of our tasks have a hight class-skew and accuracy cannot adequately differentiate between the classifiers.

5. EXPERIMENTAL RESULTS

In this section, we discuss the results of our experiments. We assessed significance of the results using paired t-tests[4] (p-values ≤ 0.05 are considered significant).

5.1 Effects of Ghost Edges

Figures 4 and 5, respectively, compare the performance of wvRN (with and without RL) to ghostEdgeNL and logForest (with and without ICA) to ghostEdgeL. We see a consistent and often dramatic increase in performance over the baseline wvRN and logForest models due to the use of ghost edges. GhostEdgeNL significantly outperforms wvRN on Enron ≤ 0.7, Political Books $0.3 - 0.7$, and HEP-TH and Reality Mining for all proportions labeled. GhostEdgeL significantly outperforms logForest on Enron and Reality Mining ≤ 0.7, Political Books ≥ 0.3, and HEP-TH at all proportions labeled.

In many cases, the ghostEdge classifiers also outperform collective classification. GhostEdgeNL significantly outperforms wvRN+RL on Enron ≤ 0.5, HEP-TH ≤ 0.7, and Reality Mining for all proportions labeled. GhostEdgeL significantly outperforms logForest+ICA on Enron ≤ 0.7, HEP-TH at all proportions labeled, Political Books ≥ 0.3, and Reality Mining ≤ 0.5.

Figure 2 (previewed in Section 1) compares the performance of various approaches to handle label sparsity with 50% of core nodes labeled: ghost edges, collective classification, and Gaussian random fields. This figure demonstrates the robustness of the ghostEdge methods across a range of data sets with varying degrees of local consistency among labels (see Figure 8). Both ghostEdgeNL and ghostEdgeL are consistently high performers across all tasks. All of the other methods perform poorly (i.e., $\geq$ 30 AUC points from the top) on at least one of the data sets.

Figure 6 presents a more complete comparison of the approaches as the proportion of labeled nodes varies. Here, we see that the ghostEdge methods perform well in comparison to other approaches, regardless of the proportion of nodes labeled. For all data sets and proportions labeled, one of the ghostEdge classifiers is always the top performer (or tied at the top). We note that there are occasionally substantial differences in performance between ghostEdgeL and ghostEdgeNL. We present a more detailed discussion of learning and non-learning methods in Section 5.2.

5.2 Effects of Learning

Figure 6 reveals a couple of interesting things about learning vs. non-learning classifiers. First, learning methods in general are hurt more than non-learning methods by a smaller proportion of labeled nodes because learning methods rely on training examples to generate an accurate dependency model. Figure 7 shows the average number of training examples available for each data set at each proportion labeled. The Political books and HEP-TH data sets have very few training examples available at the lower proportions of labeled nodes (11 and 33, respectively at 0.1 labeled). Correspondingly, it is on these data sets that we see a dip in the learning methods relative to the non-learning methods at lower proportions of labeled data.

The second thing to note is that the performance of both GRF and wvRN+RL on the Reality Mining task actually decreases as more labels are made available. This is because there is an inverse relationship between class labels of neighbors (see auto-correlation scores in Figure 8). So, these non-learning methods take whatever truth they are given and use it to make exactly the wrong decision. The more information they get, the worse they perform. We see this same effect in Figure 4 with wvRN. GhostEdgeNL overcomes this problem by using even-step RWR, as described in section 3.3. The relative performance of logForest+ICA increases with respect to GRF and wvRN+RL as label consistency decreases since the logForest model is able to learn dependencies among neighboring labels (Figure 2).

5.3 Effects of Collective Classification and Semi-Supervised Learning

Figures 2 and 6 allows us to compare the performance of the collective classification approaches (i.e., wvRN+RL and logForest+ICA) and the GRF semi-supervised approach across data sets. On all tasks, the performance of wvRN+RL and GRF is essentially equivalent, although GRF does perform significantly better than wvRN+RL in terms of AUC on all data sets except HEP-TH. We do not report results for wvRN+ICA, but we found that wvRN+RL performed much better than wvRN+ICA overall. On the other hand, the logForest classifier demonstrated roughly equivalent performance regardless of the collective inference procedure used. These results are consistent with previous findings [12].

6. CONCLUSIONS

We focus on the problem of predicting node labels in a large graph, when (a) there are few labeled nodes and (b) local consistency (a.k.a. homophily) does not necessarily hold. To address the first problem, we introduce 'ghost edges', by judiciously adding edges between nodes, according to RWR proximity. To address the

[4]It is an open issue whether the standard significance tests for comparing classifiers (e.g., t-tests, Wilcoxon signed-rank) are applicable for within-network classification, where there is typically some overlap in test sets across trials. It remains to be seen whether the use of such tests produces a bias and the extent of any errors caused by such a bias. This is an important area for future study that will potentially affect a number of published results.

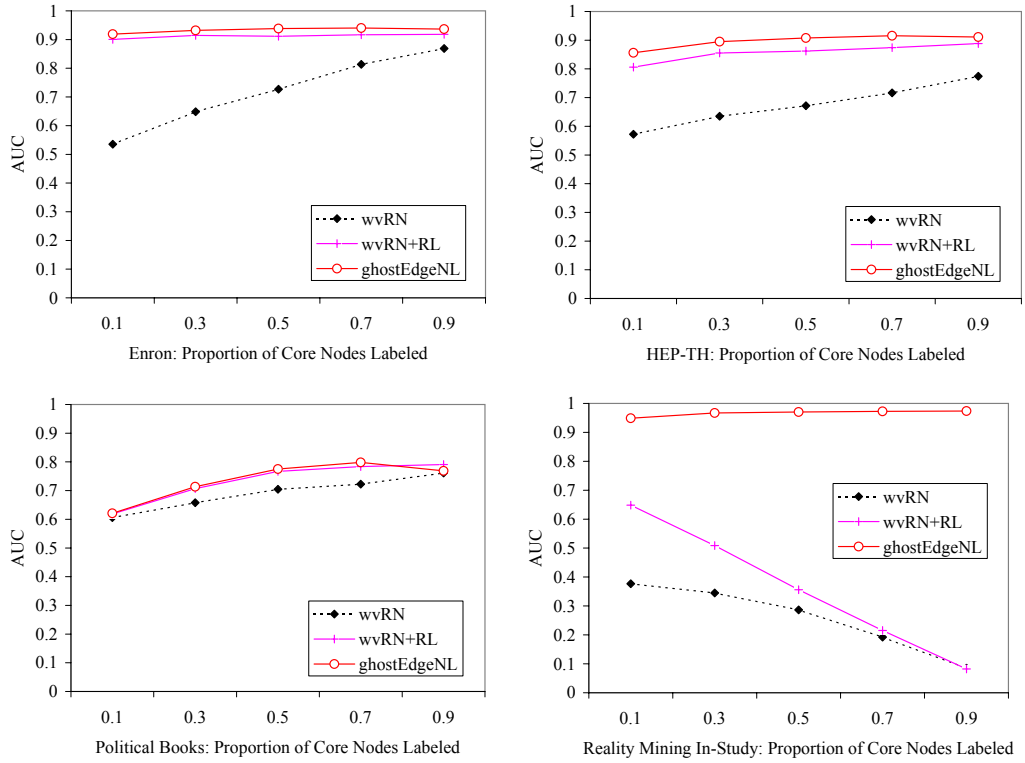

Figure 4: Comparisons of wvRN, wvRN+RL, and ghostEdgeNL. Adding ghost edges boosts performance on all data sets.

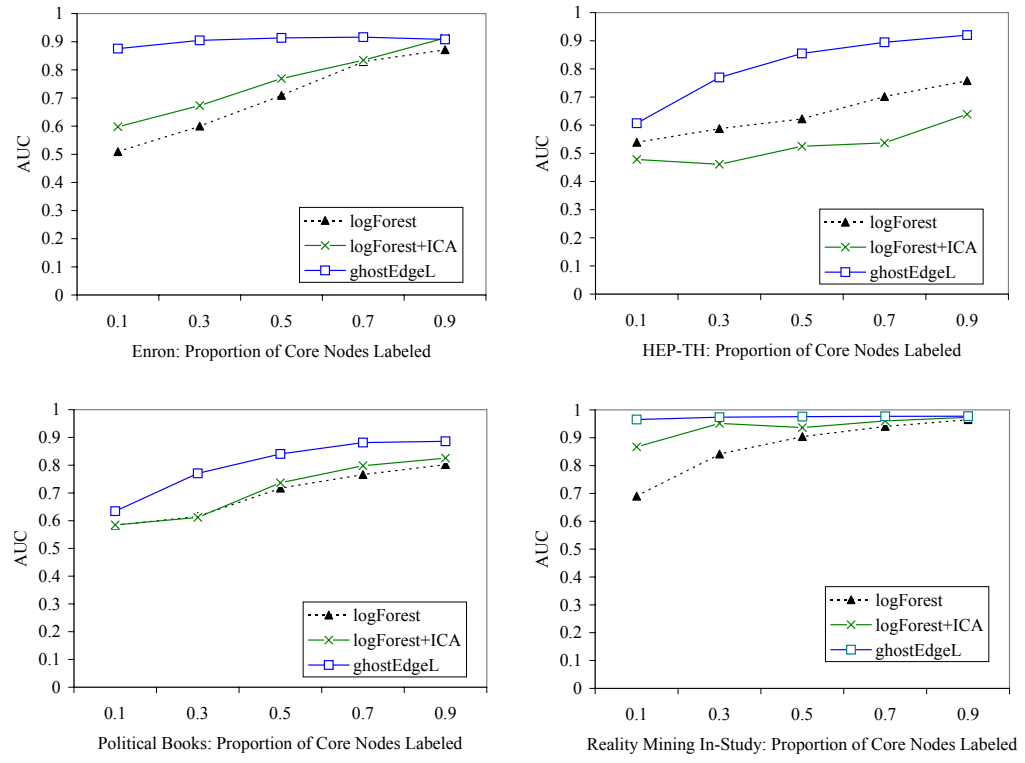

Figure 5: Comparisons of logForest, logForest+ICA, and ghostEdgeL. Adding ghost edges boosts performance on all data sets.

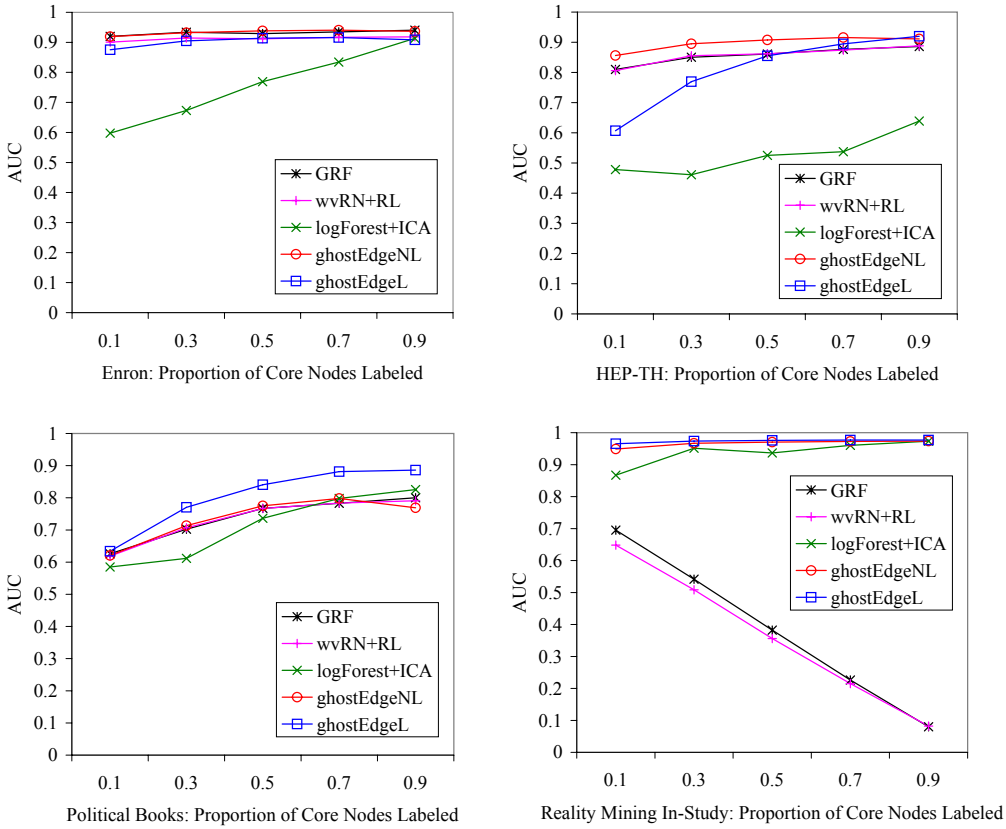

Figure 6: Comparisons of our approaches (ghostEdgeL and ghostEdgeNL) to a purely semi-supervised approach (GRF), and two collection classification approaches (logForest+ICA and wvRN+RL) as the proportion of labeled nodes varies.

second problem, we propose to bypass all the activation-spreading methods (which implicitly assume homophily), and instead we use a classifier on a carefully chosen set of features from observed, as well as 'ghost-edge' neighbors. A subtle, but vital point is that we consider RWR not on the original matrix, but on its square. This change makes our method robust, regardless of the degree of homophily. In other words, our method does well even when the local consistency assumption is not met.

We performed experiments on several real, publicly available data sets, measuring the AUC. The competitors were carefully chosen to be the state of the art. Our method is very robust, performing as well as or better than the best competitor across tasks. All other classifiers we evaluated perform poorly in some cases, depending on the degree of homophily. We also showed that the complexity of our approach is $O(L \cdot E)$, where L is the number of labeled nodes and E is the number of edges. Therefore, the approach is suitable for large data sets, provided that known labels and edges are sufficiently sparse.

7. ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-CONF-404625), and based upon work supported by the National Science Foundation under Grant No. IIS-0534205. This work is also partially supported by the Pennsylvania Infrastructure Technology Alliance (PITA), an IBM Faculty Award, a Yahoo Research Alliance Gift, with additional funding from Intel, NTT and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

8. REFERENCES

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

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

[3] F. Chung. Spectral graph theory. *Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Sociaty*, 1997.

[4] W. W. Cohen. Enron email data set. http://www.cs.cmu.edu/ enron/, 2004.

[5] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[6] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6:721–741, 1984.

[7] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

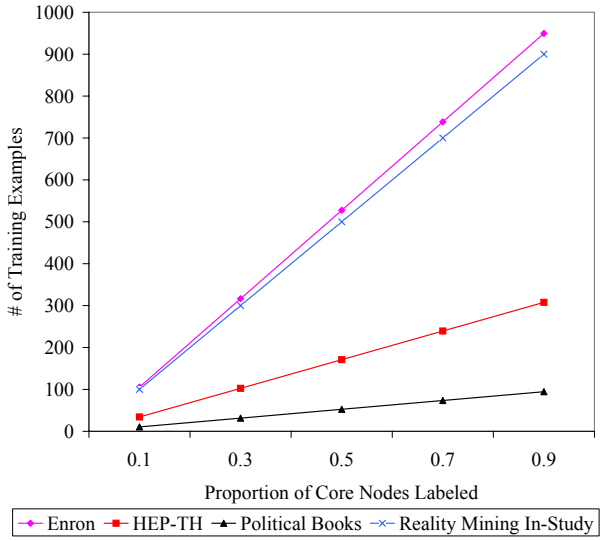

Figure 7: Number of training examples vs. proportion of core nodes labeled for each data set used in our experiments.

Data	Task	$\|L\|/\|V\|$	Avg Degree for a Node in L	Avg Degree for a Node in V	Label Consistency	Auto-correlation	P(+)
Enron	Executives?	0.34	4.31	9.10	0.94	0.21	0.02
HEP-TH	Diff. Geometry?	0.11	3.81	24.00	0.94	0.23	0.06
Political Books	Neutral?	1	8.4	8.4	0.87	0.16	0.12
Reality Mining	In Study?	1	2.75	2.75	0.07	-0.86	0.08

Figure 8: Detailed information prediction tasks. $|L|/|V|$ gives the proportion of core nodes labeled at 1.0. We define local consistency as the percentage of links connecting labeled nodes that have the same label at each endpoint. Auto-correlation is the Pearson correlation measure on the links connecting labeled nodes. Average degree is computed as the mean of the number of neighboring nodes in a set of nodes (either L or V). P(+) is the proportion of labeled nodes that have the class label of interest.

[8] D. Jensen. Proximity HEP-TH database. http://kdl.cs.umass.edu/data/hepth/hepth-info.html, 2003.

[9] V. Krebs. Books about U.S. Politics. http://www.orgnet.com/, 2004.

[10] Q. Lu and L. Getoor. Link-based classification. In *ICML*, pages 496–503, 2003.

[11] S. Macskassy and F. Provost. A simple relational classifier. In *Notes of the 2nd Workshop on Multi-relational Data Mining at KDD*, 2003.

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

[13] S. A. Macskassy. Improving learning in networked data by combining explicit and mined links. In *AAAI*, pages 590–595, 2007.

[14] L. McDowell, K. M. Gupta, and D. W. Aha. Cautious inference in collective classification. In *AAAI*, pages 596–601, 2007.

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

[16] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *KDD*, pages 625–630, 2003.

[17] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine (Special Issue on AI and Networks)*, forthcoming.

[18] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, pages 485–492, 2002.

[19] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[20] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[21] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

[22] X. Zhu. Semi-supervised learning literature survey. *Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison*, 2005.

[23] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.