LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-235752

Leveraging Structure to Improve Classification Performance in Sparsely Labeled Networks

B. Gallagher, T. Eliassi-Rad

October 23, 2007

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Leveraging Structure to Improve Classification Performance in Sparsely Labeled Networks

Brian Gallagher and Tina Eliassi-Rad
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Box 808, L-560, Livermore, CA 94551
{bgallagher, eliassi}@llnl.gov

Abstract

We address the problem of classification in a partially labeled network (a.k.a. within-network classification), with an emphasis on tasks in which we have very few labeled instances to start with. Recent work has demonstrated the utility of collective classification (i.e., simultaneous inferences over class labels of related instances) in this general problem setting. However, the performance of collective classification algorithms can be adversely affected by the sparseness of labels in real-world networks. We show that on several real-world data sets, collective classification appears to offer little advantage in general and hurts performance in the worst cases. In this paper, we explore a complimentary approach to within-network classification that takes advantage of network structure. Our approach is motivated by the observation that real-world networks often provide a great deal more structural information than attribute information (e.g., class labels). Through experiments on supervised and semi-supervised classifiers of network data, we demonstrate that a small number of structural features can lead to consistent and sometimes dramatic improvements in classification performance. We also examine the relative utility of individual structural features and show that, in many cases, it is a combination of both local and global network structure that is most informative.

Keywords

Statistical relational learning, social network analysis, feature extraction, collective classification.

1. Introduction

Many problem domains are naturally represented as networks of nodes (representing concepts) and links (representing relations between concepts). Examples include communication networks (e.g., phone-call graphs) and informational networks (e.g., citation graphs) to name a few. Networks provide a simple, but powerful representation for many kinds of problems. In this paper, we address the problem of within-network classification.

For within-network classification, we are given a network in which some of the nodes are "labeled" and others are "unlabeled" (see Figure 1). Here, "labeled" simply means that a node has been assigned a "class" from among a set of possible classes. The goal of within-network classification is to assign the correct labels to the unlabeled nodes in the network (i.e., to "classify" them).

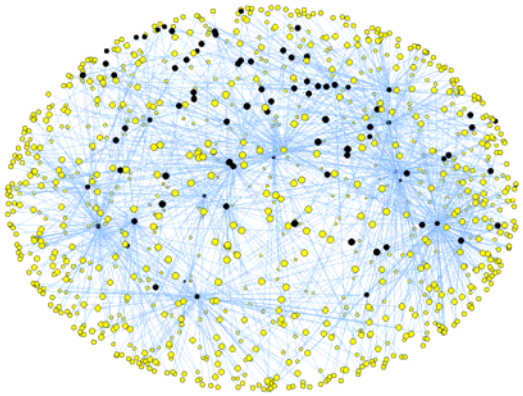

Figure 1: A Portion of the MIT Reality Mining Call Graph. We know the class labels for the black nodes, but do not have labels for the yellow nodes.

Consider the following example. Suppose we want to identify fraudulent users in a cell phone network. In this case, our set of possible classes for users is {fraudulent, legitimate}. If a user, John, is known to be involved in fraud, we assign a label of "fraudulent" to

the node representing John. If a user, Jane, is known to not be involved in fraud, we assign a label of "legitimate" to the node representing Jane. Otherwise, we leave the node unlabeled. Our goal then is to determine the labels ("fraudulent" or "legitimate") of these unlabeled users.

Cell phone fraud is an example of an application where networks are often very sparsely labeled. We may have a handful of known fraudsters and a handful of known legitimate users, but for the vast majority of users, we do not know the correct label. A number of intelligence applications have this characteristic as well (e.g., identifying members of a terrorist organization). For such applications, it is reasonable to expect that we may have access to labels for fewer than 10%, 5%, or even 1% of the nodes. Figure 1 shows an example of a sparsely labeled cell phone network. The pictured network is a subset of the MIT Reality Mining network [4]. The network contains ~32K phone calls between 1K person nodes, but title information (e.g., student, non-student) for only those in the Reality Mining study (~8% of total nodes).

In addition to being sparsely labeled, cell phone networks are generally anonymized. That is, nodes in these networks often contain no attributes besides class labels that could be used to identify them. It is this kind of sparsely labeled, anonymized network that is the focus of this work. Put another way, our work focuses on univariate within-network classification in sparsely labeled networks.

Relational classifiers have been shown to perform well on network classification tasks, because of their ability to make use of dependencies between class labels (or attributes) of related nodes [24]. However, because of their dependence on attributes of neighbors, the performance of relational classifiers can substantially degrade when a large proportion of neighboring instances are also unlabeled. In many cases, *collective classification* provides a solution to this problem, by enabling the simultaneous classification of any number of related instances [22]. However, previous work has shown that the performance of collective classification also degrades when there are too few labels available, eventually to the point where classifiers perform better without it [17, 18].

In this paper, we explore another source of information present in networks: the local and global structural patterns formed by the network's link structure. As we discuss in Section 4, we are not the first to make use of structural characteristics in the context of network classification. Researchers studying complex and social networks have long made use of structural network characteristics to provide information about individuals in a network [19]. To improve classification performance, recent work has also used network structure to (1) improve active inference [21], (2) prune a network down to its 'most informative' links [23], and (3) discover hidden groups [17]. Some classifiers also make explicit use of local structure (i.e., degree of a node) as features [15]. Our main contribution is to explore the use of both local and global network structure as features for relational classifiers in order to answer the following questions:

1. Can network structure make up for the lack of information due to sparsely labeled data?
   Answer: Yes.
2. Does structure provide any information above and beyond that provided by the class labels?
   Answer: Yes.
3. How does the benefit of structural features compare to the benefit of collective classification?
   Answer: When labels are sparse, the benefit of structural features outweighs the benefit of collective classification.
4. Is there a benefit to combining structural modeling with collective classification?
   Answer: No, not consistently.
5. Which structural features are the most useful? Is there generally one feature that is the most predictive or is it a combination of features? Are local and global features equally informative?
   Answer: We found clustering coefficient to be the least informative. A combination of betweenness centrality, number of neighboring nodes, and number of incident links were the most informative.

Section 2 describes existing approaches for within-network classification. Section 3 presents our general approach for extending these methods to incorporate information about network structure. Section 4 covers related work. Sections 5 and 6, respectively, present our experimental design and results. We summarize the paper in Section 7.

2. Background on Classifiers for Network Data

In this section, we describe existing techniques for network classification that we build on in this work.

2.1 Relational Classifiers

There has been a great deal of recent work on classifiers that make use of the relationships implicit in structured data such as networks. *Relational classifiers* predict the class labels of a set of nodes, U, given the attributes of the nodes in U and the attributes of neighboring nodes. There are three general types of relational classifiers: *individual conditional classifiers*, *collective conditional classifiers*, and *joint classifiers*

[18, 22]. Figure 2 illustrates a taxonomy for these relational classifiers.

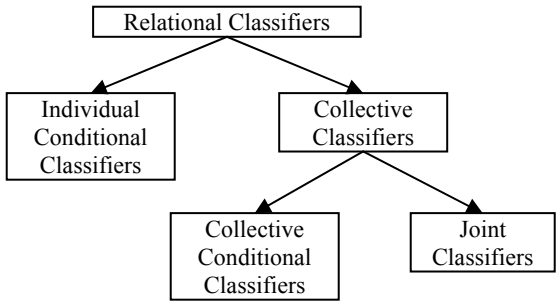

Figure 2: Taxonomy of relational classifiers for network data

Individual conditional classfiers (ICCs) are the simplest type of relational classifier. These classifiers make predictions about the nodes in U one at a time. ICCs can make use of any attributes in the graph that are known at the time classification begins, but they do not make use of the predicted class labels of nodes in U. Naturally, the performance of these classifiers can substantially degrade when a large proportion of neighboring instances are missing class labels. An example of ICC is the *Relational Probability Tree* [15].

Collective classifiers, on the other hand, make use of the predicted class labels of some nodes in U to improve the predictions about other nodes in U. That is, classifications are made "collectively" over all nodes in U "at once." There are two basic types of collective classifiers: collective conditional classifiers and joint classifiers.

Collective conditional classifiers (CCC) combine an ICC with an approximate inference procedure (such as Gibbs sampling [6]). This procedure can be viewed as a message passing algorithm, where each iteration involves a set of messages being passed between a node and its neighbors. Over many iterations, messages propagate throughout the network, allowing even simple relational classifiers to exploit dependencies between nodes separated by many links. An example of CCC is the *Relational Probability Tree* [15] combined with Gibbs sampling.

Joint classifiers attempt to learn a joint probability distribution over all known attributes in a network and then jointly classify missing labels using an approximate inference procedure (such as loopy belief propagation). An example of a joint classifier is the *Relational Markov Network* [24].

There are many different procedures for performing collective inference. The most popular include iterative classification algorithm (ICA), Gibbs sampling, mean-field relaxation labeling, and loopy belief propagation. Sen, et al. [22] and Macskassy and Provost [13] both provide empirical studies of these methods.

As a starting place for our work, we chose two simple conditional classifiers that have been demonstrated to perform well on a variety of tasks: the link-based classifier [11] and the relational neighbor classifier [12]. Since our work deals with univariate relational classification (i.e., class labels are the only attribute), we follow the recent work of Macskassy and Provost [13] and use their network-only version of the link-based classifier and their weighted-vote relational neighbor classifier. Both of these conditional classifiers can operate as individual classifiers or be combined with an approximate inference algorithm (like ICA or Gibbs sampling) to become collective classifiers.

2.2 The Network-Only Link-Based Classifier

The network-only link-based classifier (*nLB*) [13] uses logistic regression to build a discriminative model of node i's class given the class labels of nodes directly linked to i. Since logistic regression expects a fixed-length feature vector, the set of neighboring class labels is summarized by a statistic such as the count or proportion of neighboring nodes of each class.

Our *nLB* implementation uses the count of unique neighbors of each class as features. We also experimented with using normalized counts (i.e., proportion) and with weighting counts by the number of links between neighbors. However, these variations had no substantial effect on the results.

In our experiments, we run variations of this basic *nLB* classifier that make use of structural features and/or collective classification. See Section 5.3 for additional algorithmic details.

2.3 The Weighted-Vote Relational Neighbor Classifier

The weighted-vote relational neighbor classifier (*wvRN*) [12] is a simple non-learning classifier. It does not need to be trained. The *wvRN* simply estimates the probability that node i is of class c as the weighted mean of the probabilities that each of node i's neighbors is of class c:

$$P(C_i = c \mid N) = \frac{1}{Z} \sum_{j \in N} w_{i,j} \cdot P(C_j = c)$$

Here, N is the set of node i's neighbors, $w_{i,j}$ is a weight between node i and its neighbor j, and Z is a normalizer.

Since we do not use collective classification techniques that make use of uncertainty in class label assignments (e.g., relaxation labeling), $P(C_j=c)$ is 1 iff

$C_j=c$ (based on current class label assignments) and $P(C_j=c)$ is 0 otherwise. For all of our classification tasks, we set $w_{i,j}$ to the number of links between nodes i and j. So, the weights loosely represent the strength of the relationship between two nodes.

In our experiments, we run variations of this basic *wvRN* classifier that make use of structural features and/or collective classification. See Section 5.3 for additional algorithmic details.

2.4 Semi-supervised Classifiers

Although our focus is on supervised approaches for relational learning, the problem of within-network classification can also be thought of as a semi-supervised learning problem, since we have both labeled and unlabeled data available at training time. Therefore, in addition to the supervised methods described above, we examined the semi-supervised Gaussian random field (GRF) approach of Zhu et al. [26]. This is a graph-based semi-supervised learning approach where networks are generally constructed by linking instances with similar attributes. However, the general method can also be applied to networks such as ours where links are based on observed relationships (e.g., person A calls person B).

The GRF method uses a Gaussian random field model to derive a harmonic function f, which assigns a real value to each node in the network (i.e., a label). The function f is derived by minimizing the weighted squared difference between labels of neighboring nodes. The harmonic constraint means that f returns the true value of the label for each labeled node and a weighted average of f over all neighboring nodes for each unlabeled node.

3. Modeling Network Structure

In this section, we describe our approach to modeling characteristics of network structure for use in network classifiers.

3.1 A Taxonomy of Network Features

When classifying a node, i, relational classifiers make use of attributes of node i itself, as well as attributes of nodes in i's relational neighborhood. In addition, some classifiers make use of local network structure (e.g., node degree) [15]. For our work, we would like to incorporate additional network structural information. Eventually, we require all of this information to be packaged into a feature vector so that it may be used as input to a conditional classifier.

If we loosely define a feature to be a function of network observables (i.e., known or predicted attribute values and network structure), we can construct a taxonomy of feature types to capture all of the information of interest to us. Figure 3 presents such a taxonomy.

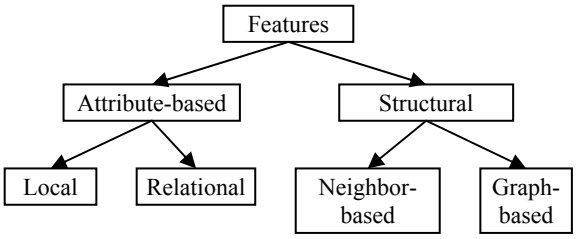

Figure 3: Taxonomy of network features

At the top level, we separate features into attribute-based and structural. Attribute-based features are further divided into: (1) local features, which are intrinsic attributes of a node (e.g., Person.name) and (2) relational features, which are calculated by applying aggregation functions to the set of attribute values of neighboring nodes (e.g., count(NeighborPerson. jobTitle = 'executive')). Structural features are further divided into: (1) neighbor-based features that provide information about the structure of the immediate neighborhood (e.g., neighborCount(Person)), and (2) graph-based features, which leverage information on the structure of a more extended neighborhood, which may even include the entire network (e.g., betweenness(Person)).

Since our work addresses the problem of univariate relational classification, we have no local attribute-based features. Our baseline relational models, *nLB* and *wvRN*, both make use of attribute-based relational features (i.e., the count of neighboring nodes of each class). In the remainder of this section, we discuss the extraction of structural features from a network and explain how we extend the *nLB* and *wvRN* models to make use of these features.

3.2 Network Structural Features

The success of network structural characteristics as predictors of class relies on two basic assumptions: (1) members of different classes play different roles in a network and (2) these roles can be differentiated by measurable structural characteristics. We know from Social Network Analysis that assumption (2) is met in many cases. For instance, popular nodes can be identified by measures such as degree (i.e., the number of connections) and nodes that are "central" to a network can be identified by measures such as betweenness centrality. Whether assumption (1) is met depends on the nature of the class label of interest. For example, suppose that executives tend to be more popular and more central than the average employee in a company communication network. Further suppose that managers of different departments tend to be

similar to one another in terms of popularity and centrality. Then, in this example, we would expect structural features to be more useful for indentifying executives than members of a particular department. The remainder of this section describes the individual structural features we use in our study.

The neighbor-based structural features we use are the degree features used by Neville et al. [15]: (1) the number of neighboring nodes and (2) number of incident links. Note that in multigraphs, these two values are generally different.

Graph-based structural features have never before been explicitly used as features in a conditional classifier. For these features, we use (1) betweenness centrality (which identifies nodes that occur along many paths) and (2) clustering coefficient (which measures neighborhood strength in terms of how connected nodes in a neighborhood are to one another). We formally define betweenness centrality and clustering coefficient next. For more details, we refer the reader to a study by Mark Newman [19].

Betweenness centrality can be defined for nodes or links. For node betweenness, we compute the following function:

$$bet_i = \frac{1}{\frac{1}{2}N(N-1)} \sum_{s \neq i \neq t \in V} \frac{g_i(s,t)}{N_{st}}$$

where $g_i(s; t)$ is the number of shortest paths from node *s* to node *t* that pass through node *i*. N_{st} is the total number of geodesic paths from *s* to *t*. *V* is the set of nodes in the network and *N* is the total number of nodes (i.e., $N = |V|$). A node with high betweenness has great influence over what information flows in the network.

Clustering coefficient for a node *i* is defined as

$$C(i) = \frac{E_i}{k_i(k_i - 1)}$$

where k_i is the number of neighbors of node *i* and E_i is the number of edges between the k_i nodes. Within social networks, the clustering coefficient captures the common belief that a friend of a friend is also a friend.

3.3 Modeling Structure with nLB, wvRN, and GRF

This section describes our general approach to extending the *nLB*, *wvRN*, and *GRF* classifiers to make use of structural features. For a more detailed description of our implementation, see the *nLBStruct*, *wvRNStruct*, and *GRFStruct* classifier descriptions in Section 5.2.

Link-based models, as originally described by Lu and Getoor [11], consist of two separate logistic regression models: one for local attributes and the other for relational attributes. When classifying a node *i*, each of these models will output a probability for each class. The two probabilities for each class are then combined into a single probability for that class by taking their product. We take a similar approach to modeling structural features with the *nLB*. Since we have no local attributes, we replace the local attribute model with a logistic regression model that takes our structural features as input, *logStruct*. Then we combine the probabilities output by the relational feature model and the structural feature model, as in the original link-based classifier.

The original link-based classifier aggregates probabilities from the two constituent classifiers by taking their product. We opted to use a weighted sum instead to allow us to control the amount we rely on attribute-based vs. structural features. Using a fixed weight of 0.5 (i.e., equal weights for attributes and structure), our experiments showed no substantial difference in performance due to the use of sum vs. product as a probability aggregator. However, for the experiments presented in this paper, we actually learn the weight *w* as described below. We then calculate the probability of each class as:

$$P(C) = w \cdot P_{nLB}(C) + (1 - w) \cdot P_{logStruct}(C)$$

Since the *wvRN* model is not a learning method, it cannot take advantage of structural features directly. Therefore, we take the same basic approach as for *nLB* and use a logistic regression model of the structural features. Then we take a weighted sum of the probabilities returned by the *wvRN* model and the structural feature model, as in the *nLB* case.

Like *wvRN*, the semi-supervised *GRF* approach is not feature-based. So, again, we take the same basic approach as for *nLB* and use a logistic regression model of the structural features. Then we take a weighted sum the probabilities returned by the *GRF* model and the structural feature model. In addition to our weighted sum approach, we also tried the approach suggested by Zhu et al. for incorporating external classifiers into their method. Their approach is to attach a "dongle" node to each unlabeled node in the graph that is given the label assigned by the external (e.g., structural) classifier. The transition from node *i* to its dongle is assigned a probability of η and all other transitions from *i* are discounted by 1 - η. This "dongle" approach did not yield any improvements over the weighted sum approach. So, we use the weighted sum approach for consistency with the other classifiers.

For all approaches, we calculate *w* based on the relative performance of attribute-only classifier (*attrOnly*) and *logStruct* on the training data. More specifically, we perform 10-fold cross validation on

the training set using each of *attrOnly* and *logStruct* separately. We calculate AUC for each fold and then obtain an average AUC score for each classifier, AUC_{attr} and AUC_{struct}. We then set w as follows:

$$w = \frac{AUC_{attr}}{AUC_{attr} + AUC_{struct}}$$

Thus, we put weight on each feature type (attribute-based vs. structural) proportional to the estimated predictiveness of that feature type, based on the training data for a particular task. Note that AUC is our performance measure of choice for this work. See Section 5.4 for further information on AUC.

4. Related Work

In recent years, there has been a great deal of work on models for learning and inference in relational data [7, 11, 12, 15, 16, 18]. Many use some sort of feature construction to incorporate attribute-based relational information. However, to our knowledge, no previous approach uses structural information from the extended neighborhood as features for classification. As we discuss below, several researchers have made use of network structure indirectly for network classification.

Note that most of these models use only a single relational feature at a time. Relational Probability Trees (RPTs) [15] use several features concurrently. However, they construct only binary features. RPTs also use neighbor-based structural features (i.e., neighboring node and link counts), but they do not use graph-based structural features such as betweenness or clustering coefficient. In addition, their work does not specifically consider the impact of using structural features on classifier performance.

In order to make simultaneous use of multiple feature types (e.g., attribute-based, structural, temporal), Gallagher and Eliassi-Rad [5] advocate the use of random forests [1], which are well suited to making sense of large feature sets.

Perlich and Provost [20] provide a nice study on aggregation of relational attributes, based on a hierarchy of relational concepts. They do not consider structural features.

Singh et al. [23] use descriptive attributes and structural properties (i.e., node degree and betweenness centrality) to prune a network down to its 'most informative' affiliations and relationships for the task of attribute prediction. They do not use the structural properties as input to their classifiers.

Rattigan et al. [21] use network structure to decide which nodes to label in an active learning setting. In addition, they utilize fast, approximate calculations for network measures such as betweenness.

Neville and Jensen [17] use spectral clustering to group instances based on their link structure (where link density within a group is high and between groups is low). This group information is subsequently used in conjunction with attribute information to learn classifiers on network data.

There are many recent papers on collective classification [2, 8, 13, 14, 16, 18, 22, 24]. Sen et al. [22] provide a careful empirical study of the various procedures for collective inference. Macskassy and Provost [13] 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 confident predicted labels. Lastly, previous work confirms our observation that collective classification's performance suffers when labeled data is very sparse [17, 18].

As discussed in Section 2.4, the problem of within-network classification can also be thought of as a semi-supervised learning problem. The graph-based approaches to semi-supervised learning are particularly relevant here. For more on semi-supervised learning, we refer the reader to an excellent survey by Zhu [26].

5. Experimental Design

We have designed our experiments to answer the following questions:

1. Can network structure make up for the lack of information due to sparsely labeled data?
2. Does structure provide any information above and beyond that provided by the class labels?
3. How does the benefit of structural features compare to the benefit of collective classification?
4. Is there a benefit to combining structural modeling with collective classification?
5. Which structural features are the most useful? Is there generally one feature that is the most predictive or is it a combination of features? Are local and global features equally informative?

To avoid confounding effects as much as possible, we focus on univariate binary-classification problems, and extend simple existing classifiers to incorporate structural information.

5.1 Data Sets

We present results on four real-world data sets: political book purchases [10], Enron emails [3], Reality Mining cell phone calls [4], and theoretical high-energy physics publications (HEP-TH) from *arXiv* [9].

The political books data set consists of 105 books labeled as liberal, conservative, or neutral. Links between books indicate that both books were purchased by the same customer. There are 441 co-purchase links in this data set. Our task is to identify the neutral books (Pr(*neutral*) ≈ 0.12). This is our only data set that is fully labeled to begin with.

From the Enron data set, we use a subset containing all data collected during a 32 day period, from 6/8/2001 to 7/10/2001. This subset consists of approximately 9K people nodes and 50K email links. We explore the task of identifying executives among Enron employees (Pr(*exec*) ≈ 0.018). For this task, we only have ground truth for a subset of 1.6K nodes (~18% of total nodes). So, our training and test sets are drawn from among these 1.6K nodes. However, we can still use the remainder of the graph (i.e., the unlabeled neighboring nodes) to calculate structural features of the labeled nodes.

For the Reality Mining data set, we use a connected subgraph of the full graph, obtained via breadth-first sampling. The subgraph was obtained by starting from a random node in the graph and expanding out in a breadth-first fashion until 1000 nodes had been touched. The final subgraph includes all nodes and links touched during the breadth-first search. The sampled subgraph consists of approximately 1K people nodes and 32K phone call links. We explore two classification tasks using the Reality Mining data set. The first task is to identify which people nodes represent study participants (Pr(*study*) ≈ 0.08). For this task, we have labels for all 1K nodes. The second task is to identify which of the 84 study participants are students (Pr(*student*) ≈ 0.62). For this task, we know the labels for each participant, but the remaining nodes are unlabeled. As in the Enron graph, we can still use the unlabeled nodes to calculate structural features of labeled nodes.

Our final data set is a network of theoretical high-energy physics publications (HEP-TH) from arXiv. Our test network was obtained via the BFS sampling method described above and contains 3K articles connected by 36K citation links. The task is to identify papers with the topic "Differential Geometry" (Pr(*DG*) ≈ 0.055). Again, this is a data set for which only a small number of nodes are labeled to begin with. In our subgraph, we have 342 labeled nodes (~11% of total nodes) from which to construct our training and test sets. Again, the unlabeled nodes may still be used to calculate structural features of the labeled nodes.

5.2 Classifiers

On each classification task, we ran ten individual classifiers: four variations of *nLB*, four variations of *wvRN*, and two variation of *GRF*. We describe each of these classifiers in detail here.

nLB is the network-only link-based classifier described previously. It is a logistic regression classifier that 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 *nLB* classifier does not use collective classification. Therefore, any neighbors with missing class labels are simply ignored.

nLBStruct is a classifier composed of two separate logistic regression models. The first model is the *nLB* described above. The second model, *logStruct*, is a logistic regression classifier that takes our four structural features as input. The *nLBStruct* classifier calculates the probability of each class as:

$$P(C) = w \cdot P_{nLB}(C) + (1 - w) \cdot P_{logStruct}(C)$$

where *w* is calculated as described in Section 3.3. Like *nLB*, *nLBStruct* does not use collective classification.

nLBCol uses the base *nLB* classifier, but performs collective classification using the *ICA* algorithm described in Section 5.3.

nLBStructCol uses the base *nLBStruct* classifier, but performs collective classification using the *ICA* algorithm described in Section 5.3.

wvRN is the weighted-vote relational neighbor classifier described previously. Given a node *i* and a set of neighboring nodes, *N*, the *wvRN* classifier calculates the probability of each class for node *i* as:

$$P(C_i = c \mid N) = \frac{1}{L_i} \sum_{j \in N} \begin{cases} w_{i,j} \text{ if } C_i = c \\ 0 \text{ otherwise} \end{cases}$$

where $w_{i,j}$ is the number of links between nodes *i* and *j* and L_j is the number of links connecting node *i* to labeled nodes. 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.

wvRNStruct is a classifier composed of two separate classifiers. The first classifier is the *wvRN* described above. The second classifier, *logStruct*, is a logistic regression classifier that takes our four structural

features as input. We use logistic regression here to model structure because *wvRN* is not a learning method and cannot model the structural features directly. The *wvRNStruct* classifier calculates the probability of each class as:

$$P(C) = w \cdot P_{wvRN}(C) + (1-w) \cdot P_{logStruct}(C)$$

where w is calculated as described in Section 3.3. Like *wvRN*, *wvRNStruct* does not use collective classification.

wvRNCol uses the base *wvRN* classifier, but performs collective classification using the *ICA* algorithm described in Section 5.3.

wvRNStructCol uses the base *wvRNStruct* classifier, but performs collective classification using the *ICA* algorithm described in Section 5.3.

GRF uses the Gaussian random field approach of Zhu et al. [26] described in Section 2.4 above. We ported Zhu's MATLAB code[1] 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 this change had no substantial impact on classification performance.

GRFStruct combines graph structure with *GRF* in the same way as *nLBStruct* and *wvRNStruct*. That is, *GRFStruct* calculates the probability of each class as:

$$P(C) = w \cdot P_{GRF}(C) + (1-w) \cdot P_{logStruct}(C)$$

where w is calculated as described in Section 3.3.

Note that we do not use the *GRF* classifiers with collective classification since the *GRF* method performs label propagation implicitly.

5.3 Collective Classification

To perform collective classification, we use the basic iterative classification algorithm described by Macskassy and Provost [13], with one modification. Macskassy and Provost allow nodes to be temporarily classified as null if all of their neighbors are unlabeled. Instead, we use our relational classifier to assign a label to these nodes, just as any other node. For relational classifiers that do not make use of network structure, these nodes end up being assigned the most prevalent class, based on the training data. In cases where the relational classifier takes network structure into account, these assigned labels are based on structure as well. We found that this approach achieved better overall classification performance, regardless of whether structural features were used. In addition, this approach generally converged quickly (i.e., within 10 trials), whereas the "null classification" approach often took the full 1000 trials. Figure 4 shows pseudo-code for our ICA algorithm, using the notation of Macskassy and Provost [13].

Let V^U be the set of unlabeled nodes in our graph.

Repeat
1. Generate a random order, O, of nodes in V^U.
2. For each node $v_i \in O$:
   a. Apply the relational classifier to v_i, using all currently assigned labels. Note: during the first iteration, some nodes will have no label. Thereafter, all nodes will have a label assigned.
   b. Assign v_i the class label with the highest probability according to the relational model.

Until 1000 iterations have elapsed or no node receives a new class label.

Figure 4: Pseudo-code for Iterative Classification Algorithm (ICA)

We chose iterative the classification algorithm because (1) it is simple, (2) it has been shown to have consistently good performance on a variety of collective classification tasks, and (3) it converges more quickly than other approaches. We also ran preliminary experiments using Gibbs sampling [6], which yielded comparable results. This is consistent with experiments done by other researchers [13, 22].

5.4 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 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.

[1] Zhu's original MATLAB code is available at http://pages.cs.wisc.edu/~jerryzhu/pub/harmonic_function.m.

Our methodology is essentially the same as the one used by Macskassy and Provost [13] for their study of within-network classification, except that we ensure that each instance in the data set is given equal weight in the overall evaluation. For each proportion labeled, we run 30 trials. 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. As Macskassy and Provost note, 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 [25]. We implement our own network data representation and experimental code, which handles tasks such as splitting the data into training and test sets, calculation of network structural features, 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. Since most of our tasks have a class-skew problem, accuracy cannot adequately differentiate between the classifiers.

6. Experimental Results and Discussion

In this section, we describe and discuss the results of our experiments. We assessed significance of the results using paired t-tests. When we use the term "significant" in the text, we mean a p-value ≤ 0.05.

6.1 Effects of Learning Label Dependencies

The *nLB* classifier is a supervised-learning based approach. It uses labeled nodes as training examples to build a model of the dependencies between class labels of neighboring nodes. The *wvRN* and *GRF* classifiers, on the other hand, do not attempt to learn these dependencies, but simply assume that class labels of neighboring nodes will tend to be the same. In cases where this assumption is met, non-learning methods can perform well. For example, *GRF* performs very well on the Enron and Reality Mining position tasks (Figure 5), both of which have a high positive correlation between class labels of neighboring nodes. However, in cases where there are more complex dependencies between neighboring class labels, non-learning methods can perform poorly. For instance, on the Reality Mining study participant task, both *wvRN* and *GRF* perform extremely poorly and actually perform worse with more known labels than fewer known labels. This is because there is actually a negative relationship between neighboring class labels in this task (i.e., non-participants never communicate directly with each other in this network). The *nLB* classifier performs well on this task because it is able to learn the correct dependencies by using available labeled nodes as training data.

6.2 Effects of Network Structure

Figure 5 shows the results of our core experiments on the effects of network structure. There are several interesting observations to note regarding Figure 5. In general, the performance of the structural classifiers degrades more slowly than that of the corresponding base classifiers as fewer instances are labeled. The exception to this is on the Enron task, in which *GRF* and *GRFStruct* are statistically tied, except at 0.1 labeled. This indicates that in general the information provided by the network structure is able to make up, at least in part, for the information lost due to missing attributes. Note that there are three separate effects that lead to performance degradation as the number of labeled instances decreases: (1) There are fewer labeled instances available for inference. This factor impacts the quality of the attribute-based features available at inference time, but has no impact on the quality of the structural features. (2) Fewer labels at training time mean that (labeled) training examples have fewer labeled neighbors. This impacts the quality of the attribute-based features available at training time and, hence, the quality of the resulting model. Again, there is no impact on the quality of the structural features. (3) Fewer labeled instances means less training data. This impacts model quality regardless of the type of features used. Note that *wvRN* and *GRF* are affected only by the first factor, since these methods do not make use of training data.

With only a few exceptions, the structural models outperform the corresponding base models. Differences are significant for *nLB/wvRN/GRF* on political books ≤ 0.7/0.9/0.7 labeled, Enron ≤ 0.5/0.5/--, Reality Mining position ≤ 0.3/≥ 0.3/≥ 0.3, Reality Mining participants ≤ 0.7/0.9/0.9, and HEP-TH ≤ 0.5/0.7/0.5. There is only one case where the use of structure significantly

Figure 5: Classification results on political books, Enron, Reality Mining, and HEP-TH data sets. The classifiers are: *nLB*, *nLBStruct*, *nLBCol*, *nLBStructCol*, *wvRN*, *wvRNStruct*, *wvRNCol*, *wvRNStructCol*, *GRF*, and *GRFStruct*. See Section 5.2 for a detailed description of these classifiers.

degrades performance. That is using *GRF* on the Enron task at 0.1 labeled. The *GRF* classifier does so well on this task on its own that adding the additional (structural) information simply adds complexity without adding additional predictive information. However, even here, the performance decrease is small compared to the gains on other tasks

The fact that structure improves performance on several tasks, including political books, up to 90% labeled suggests that the structural features may provide information above and beyond that provided by neighboring class labels. Recall that the political books network is the one data set that is fully labeled to begin with. This indicates structural features may have more general applicability beyond sparsely labeled data. In particular, there may be tasks (similar to political books) for which both (1) network structure is informative and (2) collective classification is helpful. In such cases, there may be an increased benefit to combining the two approaches.

In the next section, we discuss the effects of collective classification in our experiments.

6.3 Effects of Collective Classification

As described previously, the Enron, Reality Mining, and HEP-TH data sets all have large amounts of unlabeled data due to the fact that we simply do not have ground truth available for many of the nodes. In these cases, there are two reasonable approaches to collective classification: (1) performing collective classification over the entire graph and (2) performing collective classification over the core set of nodes only (i.e., the training and test sets).

In our experiments, the first approach produced results that were often dramatically worse that the non-collective base classifier, even when we utilized 90% of the available class labels. We hypothesize that this is due to an insufficient quantity of labeled instances to effectively seed the collective classification process. Remember that for most of our networks, there are large amounts of unlabeled data outside of the training and test sets. Other researchers have also reported cases where collective classification hurts performance due to a lack of labeled data [17, 18]. We found that the second approach (i.e., collectively classifying the core nodes only) consistently and often dramatically outperformed the first approach. Therefore, we report results using the second approach. Note that because we remove nodes from the network in the second approach, the network may become disconnected, which can adversely affect the performance of techniques that propagate labels (e.g., collective classification and SSL). However, we still found that this approach performed better than trying to perform inference over the entire network, due to the large amount of unlabeled data in many of these networks.

Figure 5 shows the effects of collective classification on our tasks. On its own, collective classification appears to have only a small effect on these tasks. For some tasks, we see a small, but significant improvement over the base classifiers due to collective classification at the higher proportions of labeled instances. For the Reality Mining study participant task, we see larger improvements due to collective classification. However, there are several cases, generally at the lower proportions of labeled instances, where we see performance degradation due to collective classification. The *nLB* model significantly outperforms *nLBCol* at some proportion labeled on Enron and *wvRN* significantly outperforms *wvRNCol* for some proportion labeled on all tasks, except identifying Reality Mining study participants.

Due to the relatively poor performance of collective classification, the comparison between the *xxCol* and *xxStruct* classifiers is not particularly illuminating. There are only two cases where the *xxCol* classifier significantly outperforms the *xxStruct* classifier (i.e., *wvRN* at 0.7 and 0.9 on Enron). There are 30 cases where *xxStruct* significantly outperforms *xxCol*.

Another effect demonstrated by Figure 5 is the interaction between the use of network structure and collective classification. Although there are a few cases where the collective structural model significantly outperforms the simple structural model, the results do not demonstrate a consistent improvement due to collective classification over and above the use of network structure on its own.

6.4 Effects of Local and Global Network Structure

Since our classifiers model a number of structural network characteristics, we want to understand which of these characteristics contribute to the observed performance gains and to what extent. To shed some light onto these issues, we ran a series of experiments on the same classification tasks described above, but using different subsets of structural features. Specifically, we ran logistic regression models with different combinations of the four structural features: each feature on its own, leave-one-feature out for each feature, neighbor-based features only, graph-based features only, and all features. This gives us 11 classifiers in all. Each classifier uses structural features only (i.e., no neighboring class labels). We ran experiments varying the amounts of labeled data available, as we had done previously. However, since the results demonstrate the same effects regardless of the proportion of labeled data, we present results only for 50% labeled.

Figures 6 and 7 show the results of these experiments. Figure 6 shows AUC results for models using each structural feature on its own as well as a model using all features combined. This demonstrates the predictive power of each feature in the absence of any other information. The model using all features serves as a reference point. Figure 7 shows the increase in AUC due to adding the specified feature to a classifier that already has access to all of the other structural features. In other words, the y-axis of this plot represents the AUC of a classifier that uses all structural features minus the AUC of a classifier that uses all features except for the specified one. This demonstrates the power of each feature when combined with the other features.

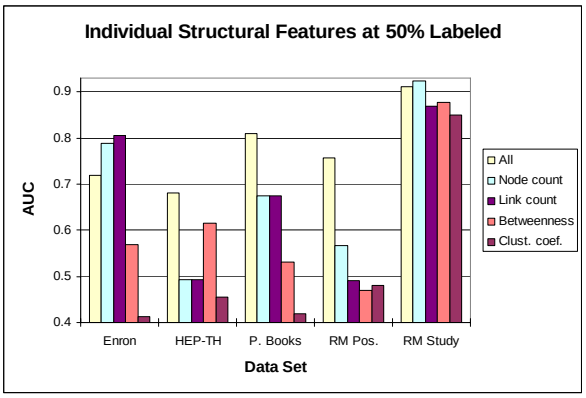

Figure 6: Comparison of the performance of structural features in isolation: node count, link count, betweenness centrality, clustering coefficient, and a combination of all four features (All).

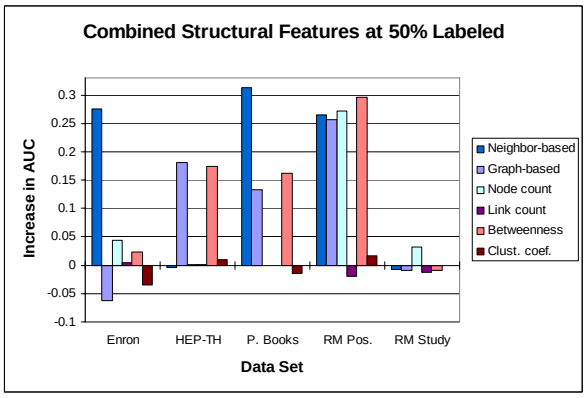

Figure 7: Comparison of the performance of structural features in combination: node count, link count, betweenness centrality, clustering coefficient, both neighbor-based features (node count and link count), and both graph-based features (betweenness centrality, clustering coefficient).

There are several interesting observations to note here. First, all of the features appear to be useful for at least some of the tasks. Clustering coefficient is the least useful overall, improving AUC only slightly on two of the tasks and degrading AUC slightly on the other three. Second, in all cases, it is a combination of at least three features that is most informative, rather than a single feature or a pair of features. Third, features that are not informative on their own can combine to provide powerful predictive information. For instance, on the Reality Mining position task, *node count*, *betweenness*, and *clustering coefficient* produce AUCs of 0.57, 0.49, and 0.48 on their own, respectively. However, when combined, these three produce an overall AUC of 0.78. Just *betweenness*, which performs worse than random (i.e., AUC = 0.5) on its own, provides a boost of 0.32 AUC when added to a classifier using *node count* and *clustering coefficient*.

For the Enron and political books tasks, the local (i.e., neighbor-based) structural features provide more predictive power than the global (i.e., graph-based) structural features. For HEP-TH, global features, specifically *betweenness*, are more important. For the Reality Mining tasks, classification performance suffers roughly equally in the absence of local or global structural features. For the HEP-TH and political book tasks, *node count* and *edge count* are identical since there is no more than one link between any pair of nodes. Therefore, in these cases, either feature can be removed without hurting performance. However, if both features are removed, this has a major effect on performance in the political books task. For the Enron and Reality Mining tasks, *node count* provides more of a boost than *edge count* when combined with the global features.

On all tasks besides Enron, performance improves or else degrades only slightly due to the inclusion of all four structural features. In the Enron case, *clustering coefficient* appears to mislead the classifier to the point where it is better to use either *node count* or *edge count* individually than to use all features. This is one case where it appears that we might benefit from a more selective base classifier. Figure 8 shows a performance comparison between a logistic regression classifier using all four structural features and a random forest classifier [1] using the same features. We see that the random forest is able to make use of the informative features without being misled by the uninformative ones to the extent that we see with logistic regression. Gallagher and Eliassi-Rad [5] present additional results on random forests with a variety of relational features and compare them to several relational classifiers on different tasks.

7. Conclusion

In this paper, we addressed the problem of within-network classification in sparsely labeled networks. We presented a novel approach for modeling structural characteristics of networks as features for classification and demonstrated the value of our approach empirically. Our experiments revealed a number of interesting findings.

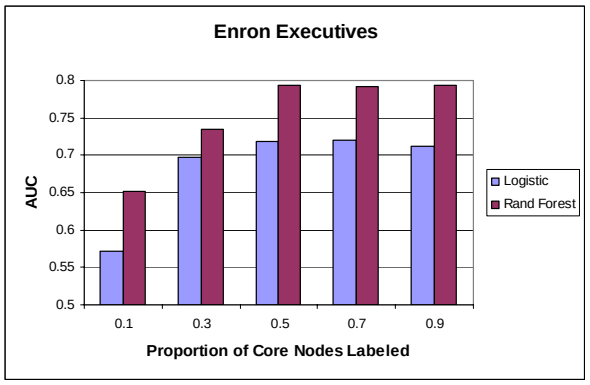

Figure 8: Comparison of logistic regression and random forest classifiers using four structural features: node count, link count, betweenness centrality, and clustering coefficient.

We discovered that network structural information can make up for vast amounts of missing class labels. We observed that structure can also provide information above and beyond that provided by class labels alone. We found that when class labels are sparse, the benefits of structural features can far outweigh the benefits of collective classification. We did not observe a consistent benefit to combining structural features with collective classification. However, we expect that in data sets where labels are less sparse, a combination of structural features and collective classification may provide an additional benefit over either technique on its own. Finally, we found that there is a benefit to combining a number of local and global structural features.

Although many of our networks have a temporal component, we have not yet made use of this information. Future work includes exploration of the dynamics of network structure in time-evolving networks. We conjecture that time-dependent structural features will further improve classification performance.

Acknowledgments

We would like to thank Luke McDowell for his insightful comments. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344. UCRL-TR-235752.

References

[1] L. Breiman, "Random forests," *Machine Learning*, 45(1), 2001, pp. 5-32.

[2] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, 1998, pp. 307-318.

[3] W.W. Cohen, "Enron email data set," http://www.cs.cmu.edu/~enron/.

[4] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Journal of Personal and Ubiquitous Computing*, 10(4), 2006, pp. 255-268.

[5] B. Gallagher and T. Eliassi-Rad, "Leveraging network structure to infer missing values in relational data," Technical Report UCRL-TR-231993, Lawrence Livermore National Laboratory, June 2007.

[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, 1984. pp. 721-741.

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

[8] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," In *Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (*KDD*), 2004, pp. 593-598.

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

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

[11] Q. Lu and L. Getoor, "Link-based classification," In *Proc. of the 20th Int'l Conf. on Machine Learning* (*ICML*), 2003, pp. 496-503.

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

[13] S. Macskassy and F. Provost, "Classification in networked data: a toolkit and a univariate case study," *Journal of Machine Learning Research*, 2007 (to appear).

[14] L. McDowell, K.M. Gupta, D.W. Aha, "Cautious inference in collective classification," In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*, 2007, pp. 596-601.

[15] J. Neville, D. Jensen, L. Friedland, and M. Hay, "Learning relational probability trees," In *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (*KDD*), (2003), pp. 625-630.

[16] J. Neville, D. Jensen, and B. Gallagher, "Simple estimators for relational Bayesian classifiers," In *Proc. of the 3rd IEEE Int'l Conf. on Data Mining* (*ICDM*), 2003, pp. 609-612.

[17] J. Neville and D. Jensen, "Leveraging relational autocorrelation with latent group models," In *Proc. of the 5th IEEE Int'l Conf. on Data Mining* (*ICDM*), 2005, pp. 322-329.

[18] J. Neville and D. Jensen, "Relational dependency networks," *Journal of Machine Learning Research*, 8, 2007, pp. 653-692.

[19] M.E.J. Newman, "The structure and function of complex networks," *SIAM Review*, 45, 2003, pp. 167-256.

[20] C. Perlich and F. Provost, "Aggregation-based feature invention and relational concept classes." In *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (*KDD*), 2003, pp. 167 – 176.

[21] M. Rattigan, M. Maier and D. Jensen, "Exploiting network structure for active inference in collective classification," Technical Report 07-22, University of Massachusetts, Amherst, MA, 2007.

[22] 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.

[23] L. Singh, L. Getoor, and L. Licamele, "Pruning social networks using structural properties and descriptive attributes," In *Proc. of the 5th IEEE Int'l Conf. on Data Mining* (*ICDM*), 2005, pp. 773-776 .

[24] B. Taskar, P. Abbeel, and D. Koller, "Discriminative probabilistic models for relational data," In *Proc. of the 18th Conf. on Uncertainty in AI* (*UAI*), 2002, pp. 485-492.

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

[26] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions." In proc. *of the 20th Int'l Conf. on Machine Learning* (*ICML*), 2003.