

Using Graph-based Metrics with Empirical Risk Minimization to Speed Up Active Learning on Networked Data

Sofus A. Macskassy
Fetch Technologies, Inc.
841 Apollo St
El Segundo, CA 90245
sofmac@fetch.com

ABSTRACT

Active and semi-supervised learning are important techniques when labeled data are scarce. Recently a method was suggested for combining active learning with a semi-supervised learning algorithm that uses Gaussian fields and harmonic functions. This classifier is relational in nature: it relies on having the data presented as a partially labeled graph (also known as a within-network learning problem). This work showed yet again that empirical risk minimization (ERM) was the best method to find the next instance to label and provided an efficient way to compute ERM with the semi-supervised classifier. The computational problem with ERM is that it relies on computing the risk for all possible instances. If we could limit the candidates that should be investigated, then we can speed up active learning considerably. In the case where the data is graphical in nature, we can leverage the graph structure to rapidly identify instances that are likely to be good candidates for labeling. This paper describes a novel hybrid approach of using of community finding and social network analytic centrality measures to identify good candidates for labeling and then using ERM to find the best instance in this candidate set. We show on real-world data that we can limit the ERM computations to a fraction of instances with comparable performance.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning—*concept learning*; J.4 [Social and Behavioral Sciences]: Miscellaneous; E.1 [Data Structures]: Graphs and networks; G.2.2 [Discrete Mathematics]: Graph Theory—*graph algorithms*

General Terms

Algorithms, Design, Experimentation, Performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'09, June 28–July 1, 2009, Paris, France.

Copyright 2009 ACM 978-1-60558-495-9/09/06 ...\$10.00.

Keywords

active learning, statistical relational learning, semi-supervised learning, social network analysis, betweenness centrality, closeness centrality, community finding, clustering, empirical risk minimization, within-network learning

1. MOTIVATION

Active learning and semi-supervised learning are both important techniques when labeled data are scarce and unlabeled data are abundant. Active learning targets the situation where it is costly to get more labeled data so which instance(s) should you get labels for in order to get the best learned model. In such a scenario, we let the learning algorithm pick a set of unlabeled instances to be labeled by an oracle (i.e., human), which will then be used as (or to augment) the labeled data set. In other words, we let the learning algorithm tell us which instances to label, rather than selecting them randomly. Active learning is named as such because the learner actively asks for more labels in order to increase its efficacy, thereby minimizing the amount of labeled data needed to get a good model. Semi-supervised learning takes an orthogonal approach to active learning and instead *uses* unlabeled data to help supervised learning tasks. The name “semi-supervised learning” comes from the fact that the data used is between supervised and unsupervised learning. Semi-supervised learning promises higher accuracies with less annotating effort. Various semi-supervised learning methods have been proposed and show promising results. For an overview of methods, there is a good regularly-updated survey available on semi-supervised learning [36]. Combining these two learning frameworks seems intuitively to make sense, yet we are aware of only one such pairing [38].

Recent work in the active learning field suggests an optimal way to identify which instances to label next by minimizing expected classification error [29]. This approach is also known as empirical risk minimization (ERM). The approach, while having very good performance, is also computationally expensive as one need to consider, for each instance, what the classification error would be if one was to know the correct label for that instance. In other words, look at the expected classification error as the instance takes on each of the possible class-labels and weight it by the (predicted) likelihood that the instance takes on that class label. This means that one needs to induce a new classifier for each unlabeled instance in order to estimate its risk.

One recently advocated method for semi-supervised learning on relational data uses Gaussian fields and harmonic functions [37]. This particular method, and graph-based methods like it (e.g., [5, 18, 37, 34]), assume that data is presented in the form of a partially labeled graph, where all the data is interconnected. This setting is also known as *within-network* learning in statistical relational learning [21], although the graph-based methods consider only the univariate case (i.e., the graph consists only of one type of node and edge and the only variable is the class label, whose value is known for a subset of the nodes in the graph).

The Gaussian fields and harmonic functions approach has been used in conjunction with active learning with some success [38]. To our knowledge, this is the only published work where these two learning methodologies have been combined. Various active learning strategies were tested in this work, where they found that using ERM did indeed perform the best among the various strategies. While an efficient way of computing the risk was proposed (to minimize the computational cost), one still needs to consider all the unlabeled points, which is an expensive task. One key observation made in the work was that ERM often picked nodes that were centers of “groups” in the graphs.

This is a key observation which we seek to use here in order to speed up active learning. Specifically, if one considers the case where the data is graph-based, then it seems reasonable that one may use other graph-based methods to identify nodes in the graph that would be good to label next. The key contribution of this paper is an exploration into the use of social network analytic methods to identify which nodes in a partially labeled graph would be the best to label next as part of an active learning strategy. Specifically, we explore the use of various social network analysis centrality metrics (see, e.g., [35]) and community-finding algorithms (see, e.g., [11]) to find central nodes in the graph and in the communities (or clusters) in the graph. As is implied by this approach, we limit the range of query selection to the unlabeled data set, a practice known as poolbased active learning or selective sampling.

As our results will show, we find that using graph-based metrics by themselves do not perform very well as an active learning strategy, but combining them with the risk minimization principle turns out to work quite well. In other words, picking the most central node(s) using centrality measures such as closeness or betweenness does not work very well. Neither does it work well if we first find communities (clusters) and then pick central nodes in the clusters. However, by using these methods to pick a small subset of the unlabeled data as *candidates* for labeling and then using ERM to find which of those candidates to label turns out to be a very good hybrid approach that gives us comparable performance to that of using ERM by itself. The key benefit and contribution is that we can use these methods to pick only a fraction of the unlabeled data on which to perform the expensive ERM calculations. The end result is that we achieve an order of magnitude speedup in active learning. We believe that we can achieve even higher speedups on larger data sets.

The remainder of the paper describes related work in Section 2, the semi-supervised algorithm used in this paper in Section 3, the active learning strategies we explore in Section 4. Section 5 describes our experimental study and results. We conclude by discussing our findings in Section 6.

2. RELATED WORK

There has been a great deal of research in active learning. For example, [32] select queries to minimize the version space size for support vector machines; [12] minimize the variance component of the estimated generalization error; [16] employ a committee of classifiers, and query a point whenever the committee members disagree. Most of the active learning methods do not take further advantage of the large amount of unlabeled data once the queries are selected. The work by [22] is an exception, where EM with unlabeled data is integrated into active learning. Another exception is [25], which uses a semi-supervised learning method during training. In addition to this body of work from the machine learning community, there is a large literature on the closely related topic of experimental design in statistics; [10] give a survey of experimental design from a Bayesian perspective.

Semi-supervised learning is similarly an active research area. We here briefly consider methods that are directly relevant to the graph-based setting for our study. We refer to that setting as the univariate within-network setting. For a good review of semi-supervised learning in general, we refer the reader to a regularly-updated survey [36].

Of particular relevance to the work presented in this paper are semi-supervised methods that work on graphs or networked data (see e.g., [5, 18, 37, 6, 34]). In these published works, the setting is not initially one of network classification. Rather, these techniques are designed to address semi-supervised learning in a transductive setting [33], but their methods have direct application to certain instances of the univariate network-classification. All the above-mentioned references induce graphs over initially non-graph data by connecting the data into a weighted network. This is done by adding edges (in various ways) based on similarity between instances.

Previous experiments [6] have shown that a randomized min-cut method empirically does not perform as well as the Gaussian fields and harmonic functions method [37]. In this latter work, the induced network is treated as a Gaussian field (a random field with soft node labels) constrained such that the labeled nodes maintain their values. The value of the energy function is the weighted average of the function’s value at the neighboring points. The result is a principled, independent development of the weighted-voting relational neighbor classifier (wvRN) from statistical relational learning [21, 20]. Experimental results show these two procedures to yield almost identical generalization performance, albeit the matrix-based procedure is much slower than the iterative wvRN. This suggests that wvRN can be used on larger data sets where the matrix-based procedure is not tractable to use, especially as part of an active learning strategy. We explore the use of the matrix-based method in this paper to extend prior work in that area and leave an exploration of wvRN for future studies.

3. SEMI-SUPERVISED LEARNING

We here employ the semi-supervised learning framework using Gaussian random fields and harmonic functions from recent work [37].

Following the semi-supervised paradigm, we assume that we are given l labeled instances $L = \{(x_1, y_1), \dots, (x_l, y_l)\}$ and u unlabeled instances $U = \{x_{l+1}, \dots, x_{l+u}\}$. Usually $l \ll u$. In our setting, we further assume that we are given

a set of edges between instances, $E = \{w_{ij}|i, j = 1 \dots (l + u), i \neq j\}$, where w_{ij} denote the weight (strength) of the edge between instances i and j and $w_{ij} = 0$ if there is no edge connecting the two. We note that while $w_{ij} = w_{ji}$ for most data (and for the data we use on our study below), this is not a necessary condition. We can then represent the data as a graph: $G = (V, E)$, where $V = \{x_i|i = 1 \dots (l + u)\}$ and E is as just defined.

The goal is to compute a real-valued function $f : V \rightarrow \mathbb{R}$, over G and then to assign labels according to f . We would like unlabeled points that are near each other to have the same label. Implicit in this is that we are assuming that the simple-yet-ubiquitous principle of homophily [4, 24] is true (cf. assortativity, [27], and relational autocorrelation [17]). This assumption motivates the use of the following quadratic energy function on f :

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} (f(i) - f(j))^2 \quad (1)$$

The minimum energy function $f = \operatorname{argmin}_{f \parallel L=f_l} E(f)$ is *harmonic*, meaning that it satisfies $\Delta f = 0$ on unlabeled data points U and is equal to f_l on the labeled points L . Here Δ is the *combinatorial Laplacian*, given in matrix form as $\Delta = D - W$, where D is the diagonal matrix with diagonal entries $d_i = \sum_j w_{ij}$ (non-diagonal entries are 0), and $W = [w_{ij}]$ is the weight matrix.

The harmonic property means that that value of f at each unlabeled data point is the average of f at neighboring points:

$$f(j) = \frac{1}{d_j} \sum_{i \sim j} w_{ij} f(i), \text{ for } j = l + 1, \dots, l + u. \quad (2)$$

We can compute the solution to f explicitly in terms of matrix operations. We first partition the Laplacian matrix into Δ blocks for labeled and unlabeled nodes,

$$\Delta = \begin{bmatrix} \Delta_{ll} & \Delta_{lu} \\ \Delta_{ul} & \Delta_{uu} \end{bmatrix} \quad (3)$$

and let $f = \begin{bmatrix} f_l \\ f_u \end{bmatrix}$, where $f_l = y_L$ and f_u denotes the mean values on the unlabeled data points. The solution is then given by:

$$f_u = -\Delta_{uu}^{-1} \Delta_{ul} f_l \quad (4)$$

f is an $C \times n$ matrix, where $n = l + u$ and C is the number of classes such that $f_{ic|L} = 1$ for $y_i = c$ and $f_{ic|L} = 0$ for all $c \neq y_i$ over the labeled instances, and $f_{jc|U} = p(y_j = c|G)$, namely the predicted probability that instance j belongs to class c .

4. ACTIVE LEARNING STRATEGIES

This section describes the specific active learning strategies we explore in this paper and our approach for using the cheap label strategies to identify a small subset of candidate instances that can then be analyzed by the more computationally expensive risk minimization strategy.

4.1 Empirical Risk Minimization (ERM)

Statistical models generate probability distribution over the class labels. If the classifier is certain of an instance belonging (not belonging) to a class, then the class probability is closer to 1 (0). If the classifier is uncertain, then

the probability gets closer to 0.5. In other words, the larger the margin, the more certain the classifier is. The empirical risk strategy seeks to label the instance that is most likely to increase the margin the most.

We define the estimated *empirical risk* \hat{R} or *margin* as:

$$\hat{R}(f) = \sum_c \sum_{i=1}^n \min(f_{\max}(i), 1 - f_c(i)), \quad (5)$$

where $f_c(i)$ is the probability that instance i belongs to class c according to the function f and $f_{\max}(i) = \operatorname{argmax}_c f_c(i)$.

The empirical risk minimization (ERM) principle seeks to find the instance that minimizes the risk (maximizes the margin). We do not know the true label for any of the instances, but we can use the classifier to estimate the likelihood that each unlabeled instance belongs a specific class. We define the estimated risk after knowing instance k as:

$$\hat{R}(f^{+k}) = \sum_c f_c(k) \cdot \sum_{i=1}^n \min(f_{\max}^{+k}(i), 1 - f_c^{+k}(i)), \quad (6)$$

ERM picks the instance k that minimizes risk:

$$k = \operatorname{argmin}_k \hat{R}(f^{+k}) \quad (7)$$

Computing f^{+k} using the Gaussian fields approach would normally involve computing the new solution, which is $O(n^3)$ as we need to invert the Δ matrix. However, we note that [38] describes an efficient way ($O(n)$) to compute the change in risk after labeling instance k as class c . This change is computed as follows:

$$f_c^{+k}(i) = f_c(i) + (y_c(k) - f_c(k)) \frac{(\Delta_{uu}^{-1})_{-k}}{(\Delta_{uu}^{-1})_{kk}} \quad (8)$$

where $(\Delta_{uu}^{-1})_{-k}$ is the k -th column of Δ_{uu}^{-1} matrix, $(\Delta_{uu}^{-1})_{kk}$ is its k -th diagonal element and $y_c(k) = 1$ when k is labeled as c and 0 otherwise. This computation is linear since Δ_{uu}^{-1} is already computed when computing the solution f . This makes it possible to compute $\hat{R}(f^{+k})$ in $O(Cn)$, where C is the number of classes.¹ Finding the best label then takes $O(Cn^2)$ time.

This strategy is the gold standard we will compare against.

4.2 Uncertainty

One strategy that intuitively seems that it should work well is that of labeling instances where the classifier is most uncertain. In other words, find the instances that are closest to the margin. We define the uncertainty, $U(i)$, of a prediction as:

$$U(k) = f_{\max}(k) - f_{\max'}(k), \quad (9)$$

where $f_{\max'}(k)$ is the second most likely class label for instance k .

It has been shown that this is often a sub-optimal labeling strategy, but we note that this actually performs quite well on some of our benchmark data and therefore include it in the library of strategies that we will use in our study below.

4.3 Graph-based Metrics

We now turn to one of the key contributions of this paper: graph-based selection strategies. When the data is relational

¹For clarity, the computation we show is $O(C^2n)$. It is straight forward to optimize the risk minimization computation to be $O(Cn)$.

in nature and can be expressed as a graph, then it seems obvious that one might be able to leverage the graph structure to identify the nodes in the graph that are important (central or influential) from a communication flow perspective. Intuitively one would expect such central nodes to also be good candidates to label. In fact, node centrality is a core concept in the analysis of social networks (see, e.g., [35]), from which a variety of centrality measures have been proposed. For example *degree centrality* computes the degree of a node: the more nodes you are connected to, the higher your centrality score. Of particular interest here are two particular and often-used metrics known as shortest-path betweenness centrality and shortest-path closeness centrality.

We also consider an important graph-based observation in the original work that combined semi-supervised learning and active learning: the nodes that ERM picked (on synthetic data sets with clear differently labeled clusters) were generally central nodes in the clusters. This suggests that clustering the graph and then finding central nodes in the clusters may be a good alternative way of finding candidate nodes. We explore the use of *community finding algorithms* (cf. [28]) to cluster the graph and then the use of centrality metrics within each cluster to find central nodes in the cluster to label.

4.3.1 Betweenness Centrality

Graph betweenness centrality is perhaps one of the most prominent measures of centrality [1, 15]. It measures the degree to which a node in the network is in a position of brokerage by summing up the fractions of shortest paths between other pairs of vertices that pass through it. This is therefore also known as *shortest-path* betweenness. Some centrality measures impose that a network need to be undirected and/or unweighted. This has not been imposed on the definition and computation of betweenness centrality.

Betweenness measures the degree of brokerage (or mediation [7]) for each node in the graph, meaning that it measures how much information is propagated through each node. In graph-based semi-supervised learning, we often *propagate* label information through the network, and identifying key nodes for such propagation seems a good choice for label selection in active learning.

We define shortest-path betweenness as:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}, \quad (10)$$

where $\sigma(s,t)$ is the number of shortest paths between nodes s and t in the graph ((s,t) -paths), and $\sigma(s,t|v)$ is the number of (s,t) -paths that go through node v . By convention, if $s = t$, $\sigma(s,t) = 1$ and if $v \in \{s,t\}$, then $\sigma(s,t|v) = 0$. By convention, we let $\frac{0}{0} = 0$.

In unweighted networks, the shortest path is defined in terms of the number of edges that connect two nodes. In weighted networks, we define shortest path as the sum of edge weights for the edges that connect two nodes.

We need to compute all-pairs shortest-paths to get the centrality measure for all nodes. However, we note that there are efficient ways for computing this [8]. In fact, the all-pairs shortest-path computation can be done in $O(nE)$ time, where E is the number of edges and n is the number of nodes in the graph. We will not include the algorithm here, but refer the reader to the original document [8] for details.

We use this metric directly as a labeling strategy as well

as part of our hybrid approach. We note that this is a global measure and this strategy therefore imposes a consistent ranking of nodes regardless of any initial state (i.e., any initial labels will have no effect on this label strategy). This is actually a nice property because it means that it is robust to the initial labeled state as it ignores any initially known labels.

4.3.2 Closeness Centrality

Graph closeness centrality is one other very popular measure of centrality [2, 30]. It measures how close a node is to all other nodes in the network as defined by the shortest path from the source node to the destination node. As with betweenness, this metric inherently handles weighted and directed networks. Intuitively it seems a good idea to label nodes that are close to other unlabeled nodes as they would have a high impact on the certainty the classifier would have in its classification.

To obtain large values for small sums of distances, shortest-path closeness is defined as the inverse of the total distance:

$$c_C(v) = \frac{1}{\sum_{t \in V} d(v,t)}, \quad (11)$$

where $d(v,t)$ is the (weighted, directed) distance from node v to node t in the graph. These nodes are important because they can easily reach (and possibly be reached by) other nodes in the network.

As with betweenness, we need compute all-pairs shortest-paths to get the centrality measure for all nodes. We note that the same core algorithm can be used to compute both betweenness and closeness.

We intended to use this metric directly, but preliminary experiments suggests that this does not perform very well by itself. We evaluated the strategy both in global terms as well as conditioned on the distances to unlabeled nodes. However, it turns out that we can use the strategy as part of the community-finding strategy described below.

4.3.3 Community Finding

If we are correct that instances of the same class tend to cluster together, then it makes a lot of sense to cluster a graph and selecting nodes within each found cluster. We note that the semi-supervised algorithms that are graph-based in fact assume that nodes that are near each other in the graph ought to have the same label. We therefore explore the use of a clustering algorithm to help us identify such nodes.

We note that a class of modularity-based algorithms for detecting community structure have over the recent years received a lot of attention (see, e.g., [28, 11]). These algorithms seek to optimize a particular modularity function that maximizes within-cluster connectivity while minimizing across-cluster connectivity. They are generally based on using edge-betweenness heuristics to remove edges one at a time until the optimal modularity is found. The particular algorithm that we use here is a modified version of a fast community detection algorithm which works in reverse: it starts with each node in a separate cluster and iteratively merges clusters until the optimum modularity is reached using a greedy search [11]. This fast algorithm is shown to run in $O(n \log^2 n)$ time on sparse networks (which most real world networks are). We use a modified algorithm which can handle weighted networks as the published version works

only on unweighted networks. We refer the reader to the literature for a description of this algorithm.

The way clustering is used here is to identify clusters that do not yet have any labeled nodes and then find a central node in that cluster that should be labeled. As we likely have multiple clusters, we then rank the central nodes based on how large they are. We would like to first label the larger clusters as that would presumably give the classifier a larger cluster that it could become more certain of. To do this, we modify the closeness centrality measure to be group specific:

$$c_C(v) = \frac{1}{\sum_{t \in C_v} d(v, t)}, \quad (12)$$

where C_v is the set of nodes in the cluster that node v belongs to.

The advantage in our setting to using the community finding algorithm is three-fold: (1) the algorithm automatically detects an optimal number of clusters, (2) the clustering is hierarchical such that we can look at sub-clusters, and (3) it is quite fast—almost linear for sparse graphs.

The community finding algorithm finds the optimal clusters initially and we start by getting labels for those clusters: We identify any of these top-level clusters that have no known label and first get labels for those clusters. Once all those clusters have at least one label, then we go to their sub-clusters (all sub-clusters of all the top-level clusters) and repeat using a breadth-first approach.

We explored using this strategy directly, but as with the closeness strategy, preliminary experiments suggests that this does not perform very well by itself. However, closer analysis of its performance revealed that while its top-picks in each cluster were generally not the best, the top $k \geq 3$ nodes in each cluster often had a node that ERM would have picked. We therefore do not evaluate this strategy but use it as part of the hybrid approach described below.

4.4 Hybrid Approach

Our initial experiments showed that while betweenness often worked well, it was not consistent and it was not as good as ERM. Further, neither the closeness nor the community-finding strategy worked well by themselves, although the community-finding algorithm often was in the right ballpark as it often had a good selection if one were to consider more than just the best node at each iteration.

This suggests that perhaps a better approach is to use the graph-based strategies to prune down the list of possible nodes that one should look at in more detail using ERM. This approach constitutes the main contribution of this paper: using a hybrid strategy to first prune candidate nodes and then using ERM on this smaller selection without hurting performance and yet speeding up active learning significantly.

The hybrid strategy works by asking three strategies for their top picks, and then using ERM to pick the best pick among the union of these top picks. The strategies the hybrid approach uses are: uncertainty, betweenness and cluster-finding. The hybrid approach asks each strategy for its top 3 picks (for a maximum of 9 nodes, although there is often some amount of overlap). It then uses ERM to find the top candidate node among this set. We explored using other settings and get similar behavior although we need to get more than one pick per strategy for this to get good performance.

As we limit each iteration to having to do at most 9 ERM node-computations, we achieve a significant amount of speedup. If we consider the amount of computations needed at each iteration, we can see that this is quite significant. The uncertainty labeling strategy takes $O(n)$ to find the top k for small k (although our implementation was not optimized and actually sorted the list to give an $O(n \log n)$ computational cost). Using the betweenness centrality strategy is likewise relatively cheap. It has an initial cost of $O(nE + n \log n)$, but is then $O(1)$ as the sorted list of nodes will not change over the runs. The community-finding algorithm is more complex and takes more time. It takes an initial $O(n \log^2 n)$ to generate the clusters (although this can get arbitrarily close to $O(n^3)$ as the graph becomes more dense. However, since we need to invert a matrix for the semi-supervised learning step, this is at worst the same as one learning iteration). After that, we compute the most central node in each cluster, which is $O(n_C^2)$ per cluster, where n_C is the number of nodes in that cluster. However, we only compute this for a limited set of clusters in each iteration, making $n_C \ll n$. However, this is the most costly operation after the ERM step. As we shall see, this is still significantly faster than using ERM by itself.

5. EXPERIMENTAL STUDY

We performed an experimental study on 10 benchmark networked data sets to evaluate the efficacy of our proposed hybrid approach in terms of performing comparably to ERM as well as being significantly faster in terms of time spent. We here describe our study, which consists of a description of the data, our experimental methodology and our result.²

5.1 Data

We consider in this study 11 data sets from three domains: computer science department websites (WebKB), industry news, and academic papers.³

5.1.1 WebKB

The first domain we draw from is based on the WebKB Project [13].⁴ It consists of sets of web pages from four computer science departments, with each page manually labeled into 7 categories. As with other work [26, 19], we ignore pages in the “other” category except as described below.

From the WebKB data we produce eight networked data sets, two for each of the four universities, ranging in size from 338 to 434. Four networks contained six classes, although two classes had so few instances that this, in effect, turned into a four-class problem. The second set of four networks simplified the classes into a student/not-student binary classification task.

Following prior work on web-page classification, we link two pages by co-citations (if x links to z and y links to z , then x and y are co-citing z) [9, 19]. To weight the link between x and y , we sum the number of hyperlinks from x to z and separately the number from y to z , and multiply these two quantities. This weight represents the number of

²We used NetKit-SRL for all of these studies, as it provided an easy way to ensure the same experimental environment across all runs. NetKit-SRL is available at <http://netkit-srl.sourceforge.net>

³All data sets are available at <http://netkit-srl.sourceforge.net>

⁴We use the WebKB-ILP-98 data.

possible co-citation paths between the pages. Finally, to get a sparser and cleaner graph, we keep only the top 5 most highly weighted edges from every node.

To produce the final data sets, we removed pages in the “other” category from the classification task, although they were used as “background” knowledge—allowing 2 pages to be linked by a path through an “other” page.

5.1.2 Industry Classification

The second domain we draw from involves classifying companies by industry sector. Companies are linked via cooccurrence in text documents. We use two different data sets, representing different sources and distributions of documents and different time periods (which correspond to different topic distributions).

INDUSTRY CLASSIFICATION (YH)

As part of a study of activity monitoring, [14] collected 22,170 business news stories from the web between 4/1/1999 and 8/4/1999. Following a prior study, we placed an edge between two companies if they appeared together in a story [3]. The weight of an edge is the number of such cooccurrences found in the complete corpus. The resulting network comprises 1798 companies that cooccurred with at least one other company. To classify a company, we used Yahoo!’s 12 industry sectors.

INDUSTRY CLASSIFICATION (PR)

The second industry classification data set is based on 35,318 PR Newswire press releases gathered from April 1, 2003 through September 30, 2003. We used the same data preparation as above, resulting in a network of 2189 companies that cooccurred with at least one other company. We use the same classification scheme as above.

5.1.3 Cora

The last domain is academic papers. We used the cora data set [23] comprises 4240 computer science research papers. It includes the full citation graph as well as labels for the topic of each paper (and potentially sub- and sub-sub-topics). Following a prior study [31], we focused on papers within the machine learning topic with the classification task of predicting a paper’s sub-topic (of which there are seven).

Papers can be linked if they share a common author, or if one cites the other. Following prior work [19], we link two papers if one cites the other. The weight of an edge would normally be one unless the two papers cite each other (in which case it is two—there can be no other weight for existing edges).

5.2 Methodology

The main objective of this study is to test the hybrid approach for performance and speed. We start off, for each data set, by randomly sampling one instance of each class and then run 100 iterations of active learning using various strategies. We compute time spent and classifier accuracy at each iteration. We then average over 10 runs.

5.3 Results

The primary objective in this work was to significantly speed up active learning and yet keep a performance comparable to using empirical risk minimization (ERM). We first compare ERM to uncertainty, betweenness and uniform random sampling [29] to explore the difference in per-

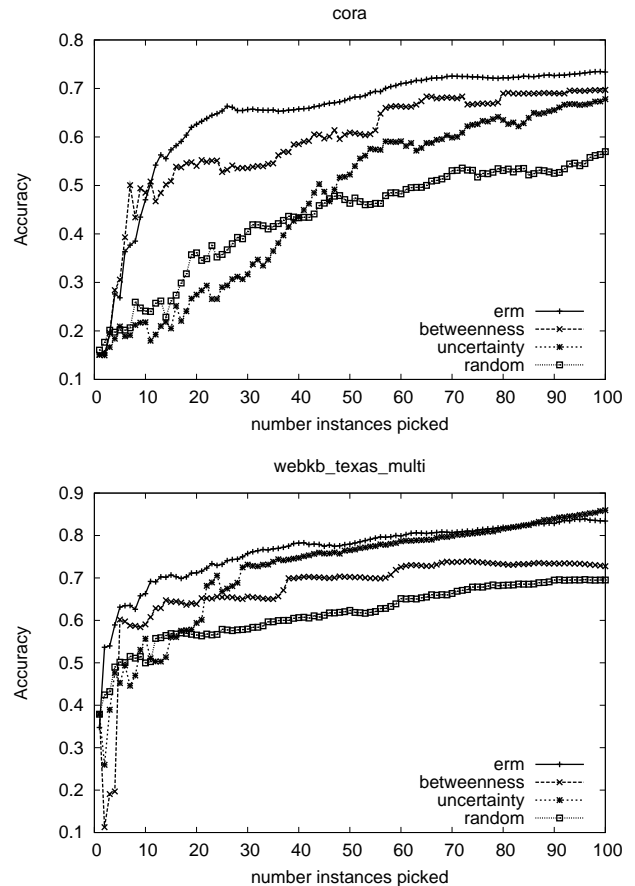


Figure 1: Two prototypical performance signatures of ERM, uncertainty labeling, labeling based on betweenness centrality and uniform random sampling. As we can see, ERM is consistently a good performance while the other two have higher variance among different data sets. We tested community finding and closeness as well, but both performed so bad that they are not included here.

formance between these four active learning strategies and to verify that ERM is in fact consistently the better strategy. Figure 1 shows prototypical performance signatures for the strategies on two of the 11 data sets we explored. As we can see, uncertainty labeling and betweenness labeling both turn out to perform close to ERM some of the time but they are not consistently as good as ERM. Also, we see that random sampling was significantly worse than ERM. In the 11 data sets, betweenness only beat ERM once in the beginning but then became worse as we sampled more points. Betweenness was otherwise consistently worse than ERM. Uncertainty labeling was somewhat comparable to ERM in 3 cases, but significantly worse in the other 8. Random sampling was always worse than ERM. These results show that ERM is the best consistent strategy to use.

We now turn to the main evaluation of this study, namely that of the hybrid approach versus ERM. Specifically, we want to see that the hybrid approach has similar performance to that of ERM and yet is significantly faster. Figure 2 shows three prototypical runs of ERM against the hy-

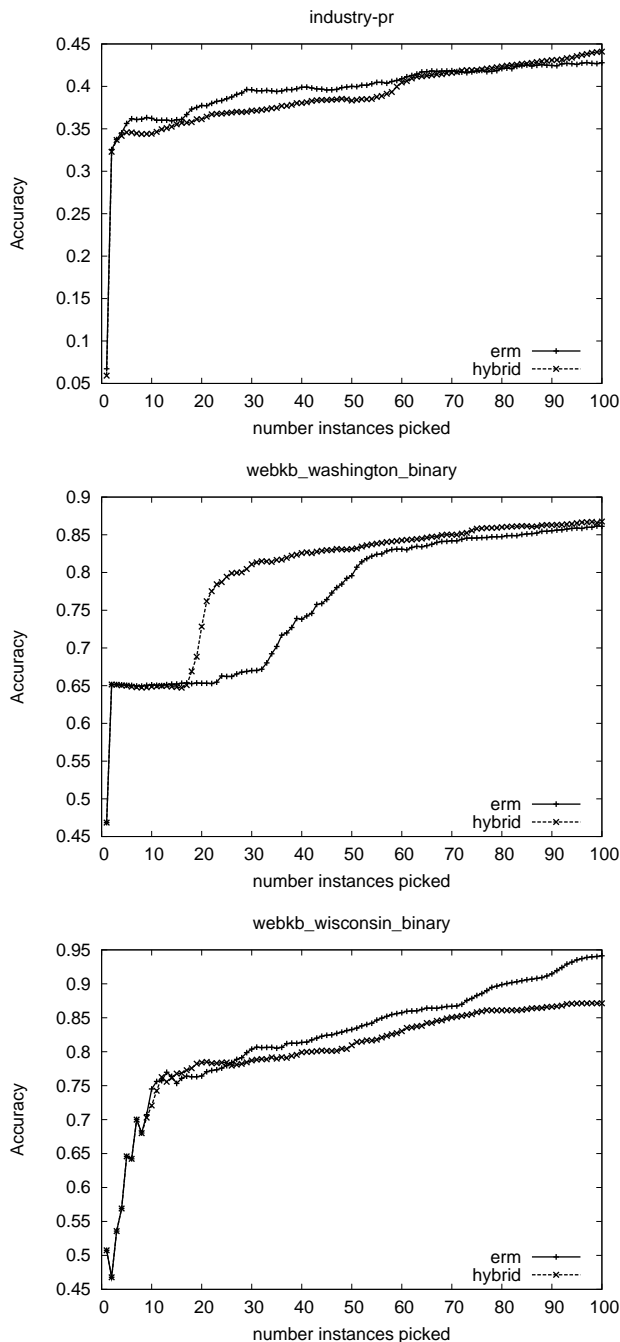


Figure 2: Three prototypical performance signatures of ERM as compared against the hybrid strategy. As we can see, they perform comparably. In fact, the most prototypical run is at the top, where they are almost on top of each other.

brid strategy. As we can see, the hybrid approach has very similar performance overall. It interestingly outperformed ERM early in the runs on the *webkb_washington_binary* data set, but they did end up converging again later on. We see that on the *webkb_wisconsin_binary* data set that ERM actually starts to outperform the hybrid approach towards the end, but still the two strategies have very similar perfor-

dataset	% picked			# multiple strategies
	CF	US	BC	
cora	11.0%	12.3%	76.9%	1
industry-pr	9.0%	36.8%	54.4%	2
industry-yh	1.9%	30.4%	68.9%	12
cornell-binary	12.6%	62.3%	36.9%	118
cornell-multi	3.5%	75.5%	29.9%	89
texas-binary	9.8%	40.1%	59.2%	91
texas-multi	16.3%	68.5%	28.5%	113
washington-binary	20.6%	72.5%	28.0%	211
washington-multi	25.7%	72.0%	25.1%	228
wisconsin-binary	38.8%	62.0%	27.4%	282
wisconsin-multi	24.0%	69.8%	25.4%	192

CF: Community finding strategy

US: Uncertainty sampling

BC: Betweenness centrality

Table 1: How often was each strategy picked for each data set (out of 1000, except for cora, where it was out of 700)? Last column shows how often more than one strategy picked the node ultimately selected by ERM.

mance curves. We note that the most prototypical comparison is on the *industry-pr* data, where the two strategies are almost on top of each other. This is the behavior we see on 7 of the 11 data sets. ERM was slightly better (1 – 2%) on the remaining data sets.

One might wonder how much each of the three strategies in the hybrid approach contributes to this performance. In particular, do we need all three strategies and what would happen if we were to use only a subset of them? Table 1 shows how often each strategy was picked as well as how often more than one strategy picked the node selected by ERM. Three interesting observations can be made from this: First, all three strategies are often used a significant amount of the time; Second, the three larger data sets relied significantly more on betweenness than the smaller data sets, which relied heavily on uncertainty sampling; Three, community finding was generally the least-used strategy. We explored whether removing betweenness or the community finding strategy would have any effect on the performance and removing either strategy did consistently make the hybrid method perform worse.

We finally turn to how long it took to do these runs.⁵ Figure 3 shows two runs, the one with the best and the one with the worst performance gain. As we can see, we achieve a tenfold speedup in the *industry-yh* data set, which is the larger of the two. We also get a tenfold speedup on the *industry-pr* data set, whereas we generally see a 3 – 5 times speedup on the *webkb* data sets. This is not all that surprising given the size of these data sets. The *webkb* data are only around 400 and its is therefore very quick to compute ERM even for all unlabeled nodes. As the data sets increase in size, the $O(Cn^2)$ cost of ERM becomes more significant as we see on the *industry* data sets. However, we run into a limitation of the classifier as the data sets become even larger as in the *cora* data set (4240 instances).

⁵We emphasize that we did not spend much time optimizing the hybrid strategy computations whereas we used the efficient ERM computations from the literature. The comparisons here are therefore not in our favor.

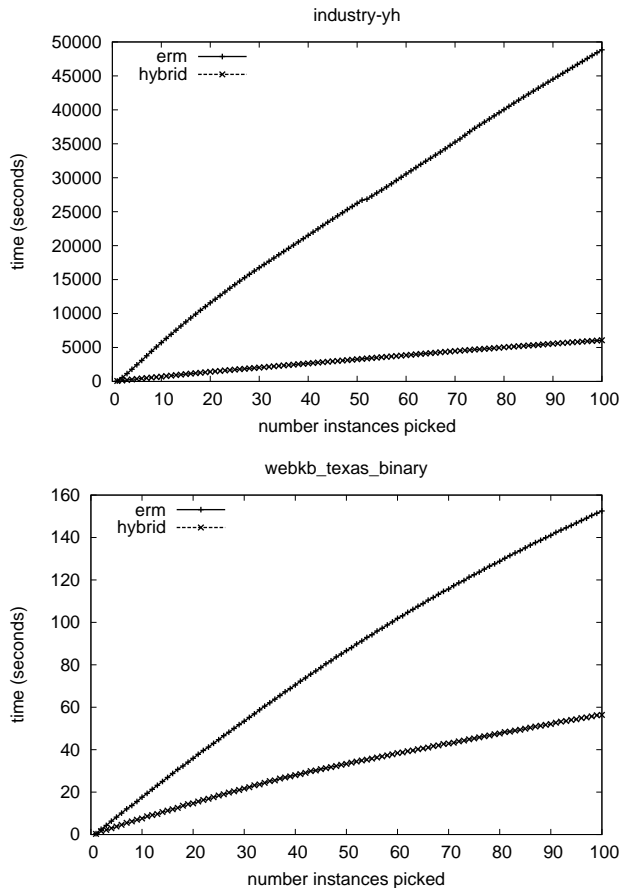


Figure 3: Two prototypical timed runs of ERM and the hybrid strategy. We pick these two as they show the two most extreme cases. In industry-yh (the larger data set), we see a ten-fold speedup, whereas we get only a three times speedup on the texas data set.

In the cora data we achieve a speedup of 5 over using ERM. This is because we use the matrix-based classifier ($O(n^3)$) with an optimized ERM computation ($O(Cn^2)$). We would expect that as the data set increases, the classifier itself becomes the bottle neck and any speedup in the sampling becomes less significant. We remind the reader that we used the matrix-based algorithm here in order to extend existing literature. This finding suggests that using a faster approximate method such as vvRN-RL as described in Section 2 is critical in order to scale active learning to larger networked data sets.

6. CONCLUSION

We observed that recent work on combining semi-supervised learning and active learning used a learning method that was inherently graph-based and, in effect, used label propagation to complete label a partially labeled graph. This setting is identical to the univariate within-network framework described in the statistical relational learning literature, and we observed that if the data is in fact networked then the structure of the network should be able to help in identifying important nodes that should be

labeled in order to achieve maximum prediction accuracy. The use of network structure to help in node selection in active learning is a novel idea.

First, we reviewed current literature on combining semi-supervised learning and active learning, noting that the current state-of-the-art active learning strategy uses empirical risk minimization (ERM), but it is computationally expensive. We therefore suggested using graph-based centrality measures from social network analysis to either replace ERM or help it by suggesting a small fraction of nodes that should be considered, thereby significantly speeding up the active learning process.

We showed on 11 benchmark graph-based data sets that although graph-based measures did not perform as well as ERM by themselves, a hybrid approach of using the graph-based centrality measures to identify a small subset of candidate nodes did in fact perform comparably to (and sometimes even beating) ERM while dramatically speeding up the learning process by more than an order of magnitude.

We note that this is an initial exploration of using the structure of the graph and that our results suggest that this research direction has potential to dramatically improve active learning in terms of speed and perhaps even performance. For example, there are numerous centrality and community finding algorithms available that are good candidates for exploring better candidate selection. We explored one way of doing so with good success. However, there is still much that can be done in terms of optimizing node selection. The work presented here shows the potential for the use of such methods.

Acknowledgments

Thanks to reviewers for suggesting improvements to this paper. Thanks to Foster Provost for initially suggesting to look at active learning in the context of within-network learning. This work was sponsored in part by the Office of Naval Research under award number N00014-07-C-0923. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Office of Naval Research or the U.S. Government.

7. REFERENCES

- [1] J. Anthonisse. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, 1971.
- [2] M. A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10:161–163, 1965.
- [3] A. Bernstein, S. Clearwater, and F. Provost. The Relational Vector-space Model and Industry Classification. In *Proceedings of the Learning Statistical Models from Relational Data Workshop (SRL) at the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 8–18, 2003.
- [4] P. M. Blau. *Inequality and Heterogeneity: A Primitive Theory of Social Structure*. New York: Free Press, 1977.
- [5] A. Blum and S. Chawla. Learning from Labeled and Unlabeled Data using Graph Mincuts. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 19–26, 2001.

- [6] A. Blum, J. Lafferty, R. Reddy, and M. R. Rwebangira. Semi-supervised learning using randomized mincuts. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.
- [7] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28:466–484, 2006.
- [8] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, May 2008.
- [9] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced Hypertext Categorization Using Hyperlinks. In *ACM SIGMOD International Conference on Management of Data*, 1998.
- [10] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:237–304, 1995.
- [11] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70, 2004. 066111.
- [12] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.
- [13] M. Craven, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and C. Y. Quek. Learning to Extract Symbolic Knowledge from the World Wide Web. In *15th Conference of the American Association for Artificial Intelligence*, 1998.
- [14] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.
- [15] L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.
- [16] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.
- [17] D. Jensen and J. Neville. Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 2002.
- [18] T. Joachims. Transductive Learning via Spectral Graph Partitioning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.
- [19] Q. Lu and L. Getoor. Link-Based Classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.
- [20] S. A. Macskassy and F. Provost. A Simple Relational Classifier. In *Proceedings of the Multi-Relational Data Mining Workshop (MRDM) at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.
- [21] S. A. Macskassy and F. Provost. Classification in Networked Data: A toolkit and a univariate case study. *Journal of Machine Learning Research (JMLR)*, 8(May):935–983, 2007.
- [22] A. McCallum and K. Nigam. Employing em in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 350–358, 1998.
- [23] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, 2000.
- [24] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27:415–444, 2001.
- [25] I. Muslea, S. Minton, and C. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of 19th International Conference on Machine Learning*, pages 435–442, 2002.
- [26] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning Relational Probability Trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.
- [27] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67, 2003. 026126.
- [28] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
- [29] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning*, pages 441–448, 2001.
- [30] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966.
- [31] B. Taskar, E. Segal, and D. Koller. Probabilistic Classification and Clustering in Relational Data. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 870–878, 2001.
- [32] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the 17th International Conference on Machine Learning*, pages 999–1006, 2000.
- [33] V. N. Vapnik. *Statistical Learning Theory*. John Wiley, NY, 1998.
- [34] F. Wang and C. Zhang. Label propagation through linear neighborhoods. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 985–992, 2006.
- [35] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge: Cambridge University Press, 1994.
- [36] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison, 2005.
- [37] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 12th International Conference on Machine Learning*, 2003.
- [38] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.