Linear Prediction Models with Graph Regularization for Web-page Categorization

Tong Zhang
Yahoo! Inc.
New York City, NY, USA
tzhang@yahoo-inc.com

Alexandrin Popescul
Yahoo! Inc.
Santa Clara, CA, USA
popescul@yahoo-inc.com

Byron Dom
Yahoo! Inc.
Santa Clara, CA, USA
bdom@yahoo-inc.com

ABSTRACT

We present a risk minimization formulation for learning from both text and graph structures which is motivated by the problem of collective inference for hypertext document categorization. The method is based on graph regularization formulated as a well-formed convex optimization problem. We present numerical algorithms for our formulation, and show that such combination of local text features and link information can lead to improved predictive accuracy.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

graph and relational learning, document classification, semi-supervised learning, regularization, collective inference

1. INTRODUCTION

Hyperlinks among web documents provide extra evidence which can improve document classification accuracy. For example, Chakrabarti et al. have shown that classes of graph neighbors seeded with a text only classifier and further refined with an EM-like technique significantly improve Yahoo Directory classification accuracy [3]. Other options exist for aggregating neighborhood class assignments, Macskassy and Provost [8] analyze classification performance with various configurations of local classifiers, relational classifiers, and collective inference methods for propagating evidence through the graph. Also see [7] for a related study by Jensen et al. Methods originating in inductive logic programming have also been applied to classification with hyperlinks. Craven and Slattery use a combination of FOIL and Naive Bayes for classification in the WebKB data [5]. Getoor et al. apply probabilistic relational models to prediction in WebKB and Cora datasets [6].

Such relational learners are designed to exploit structure in the graphs where nodes link to members of their own class or certain other classes with high probability. While we use hyperlink and co-citation relationships, relational learners have also been applied to other data, for example, Neville and Jensen used a relational Bayesian classifier to assign companies into their industry sector [10], Segal et al. used probabilistic relational models to characterize gene expression data [12], Popescul and Ungar used regression over features automatically generated from a multi-relational, including cluster relations, representation of CiteSeer for citation probability estimation and conference community classification [11].

A different approach, which can also take advantage of graph structure, has recently appeared in the semi-supervised learning literature. In this approach, one creates a graph using unlabeled data (often nearest neighbor graph by linking points that are close to each other). Discriminative learning methods such as logistic regression or support vector machines can then be directly applied using a properly defined regularization condition (often referred to as graph-Laplacian) on the graph. For example, see [2, 13, 17, 18]. This framework has drawn significant attention, with a fast growing number of papers appearing in the machine learning literature. An advantage of this approach is its equivalence to a form of standard large margin kernel methods with a specially designed kernel on graph. Therefore it is an extension of the standard kernel method in the graph setting, and the well developed theoretical and algorithmic results for large margin methods can be readily applied here [15]. Moreover, the resulting formulation is a well-formed convex optimization problem, which has a unique and efficiently computable solution. In contrast, previously proposed link-based relational learning models either implement a procedure that does not solve an optimization problem (hence such procedures do not necessarily converge), or requires approximate Bayesian inference (due to the non-convexity of the underlying Bayesian formulation). Therefore at the conceptual level, this different approach for exploiting graph structure has some unique features and advantages.

The purpose of this paper is to adapt the graph regularization framework for large margin discriminative learning to text-categorization problems on the web using text information and hyperlink structures. This leads to a new class of learning methods that complement the earlier approach from the relational learning point of view. In this frame-

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

work, combining link and text information is closely related to combining regularizers or kernels, as we will demonstrate later. Although some theoretical aspects of such combination were discussed in recent work [1], they do not lead to implementable algorithms suitable for large scale text classification problems. The algorithmic and scalability issues will be investigated in this paper. We show that by combining text and link structures, we can improve the state of the art local regularized linear classification methods for document categorization.

The main contribution of this work is to derive these new formulations within the graph regularization framework, and then present scalable numerical algorithms. For simplicity, we will not directly compare performance with the previous relational learning approaches. Some of them start with different classification methods (e.g. not discriminative linear classifier) and make improvements relative to those methods; therefore their baselines are not directly compatible with our framework. Our regularized logistic regression baseline is generally regarded as state of art for text categorization without using link information, and our contribution presents improvements beyond this baseline when graph structure is added within a unified framework.

2. PROBLEM SETUP

We define our problem more abstractly as follows. Consider the problem of predicting a real-valued output y based on its corresponding input vector x. In this paper, we are interested in the setting of collective inference where we observe a set of labeled data (X_i, Y_i) for $i = 1, \ldots, n$ and a set of unlabeled data X_j for $j = n+1, \ldots, m$. The true values Y_j for X_j is unknown. Our goal is to estimate a functional relationship $Y_j \approx p(X_j)$ for $j = 1, \ldots, m$ such that the risk $\sum_{j=n+1}^{m} L(p(X_j), Y_j)$ is small, where $L(p, Y)$ is a loss function, and $p(x)$ is a real-valued function. Slightly different from inductive inference, where (X_k, Y_k) for $k = 1, \ldots, n$ are assumed to be selected from an unknown distribution, in the collective inference setting, we assume that X_k ($k = 1, \ldots, m$) are fixed. We randomly draw n out of m samples as labeled, and reveal the corresponding labels. We then test the learning performance on the remaining samples (where the labels are hidden).

In addition to the input vector X, we assume further that a graph structure is observed on the dataset X_k for $k = 1, \ldots, m$. The vertices of the graph are the nodes X_k, with edges defined between node pairs. In the context of web-classification, the nodes are web-pages, and edges are links or co-citations between web-pages. For simplicity, we treat the graph as undirected. The graph structure will be used to construct appropriate regularization conditions for $p(X)$, as described in subsequent sections. In this work, we consider the following interpretation of the graph. If two nodes X_k and X_ℓ are linked, then X_k and X_ℓ are likely to have similar predictive values: $p(X_k) \approx p(X_\ell)$. This information, which we shall use for the purpose of designing a regularization condition, will be explored in the paper.

3. SUPERVISED LEARNING USING LOCAL INFORMATION

In the supervised setting, we use the local input vector X but ignore the graph structure. In text categorization, many classifiers such as decision trees, Naive Bayes, k-NN, have been employed [14]. It was known that sate of the art performance can be achieved by the so-called regularized linear classifiers that include SVM, regularized logistic regression, ridge regression, etc (for example, see [16]). In this paper, we focus on linear classifiers. Consider a feature map $\psi(X)$ and we are looking for linear weight w such that $Y \approx p(X) = w^T \psi(X)$, where w is a parameter we want to estimate from the training data. In text categorization, $\psi(X)$ is often a vector representation of a document X in the vector space model. Given a set of training data (X_i, Y_i), we can compute a linear weight using regularized linear prediction method, where $\hat{p}(x) = \hat{w}^T \psi(x)$, and

$$\hat{w} = \arg\min_{w \in \mathcal{F}} \left[\frac{1}{n} \sum_{i=1}^{n} L(w^T \psi(X_i), Y_i) + \frac{\lambda}{2} w^T w \right], \quad (1)$$

where $\lambda \geq 0$ is an appropriate regularization parameter.

This gives us a predictor $\hat{p}(x)$, which we can evaluate on the test data X_j ($j = n+1, \ldots, m$). In this work we are mainly interested in the binary classification problem, where $Y \in \{\pm 1\}$. Given the weight vector $\hat{w}$ trained from (1), the corresponding class label of X_j can be assigned as $\text{sign}(\hat{w}^T \psi(X_j))$. Common loss functions include $L(f, y) = \ln(1 + \exp(-fy))$ (logistic regression), $L(f, y) = \max(0, 1 - fy)$ (SVM), and $L(f, y) = (f - y)^2$ (least squares).

4. GRAPH BASED COLLECTIVE INFERENCE USING LINK INFORMATION

In graph-based learning, we are interested in finding the predictive values $f_k = p(X_k)$ directly for $k = 1, \ldots, m$. In the class of methods we consider here, this can be achieved by constructing a regularization condition $\mathcal{L}$ on the graph and solve the following problem:

$$\hat{f} = \arg\min_{f \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} L(f_i, Y_i) + \frac{\lambda'}{2} f^T \mathcal{L} f \right]. \quad (2)$$

In this representation, $\mathcal{L}$ is an $m \times m$ matrix. If we have a weighted graph with edges E among the nodes X_k, and weights $c_{k,k'}$ associated with $(k, k') \in E$, then a standard way to define the regularization condition $\mathcal{L}$ is

$$f^T \mathcal{L} f = \sum_{(k,k') \in E} c_{k,k'} (f_k - f_{k'})^2. \quad (3)$$

This regularization condition is often referred to as (unnormalized) *graph Laplacian* in the graph-learning literature. There are different heuristics for choosing weights $c_{k,k'}$, and we just take $c_{k,k'} = 1$. What is a more appropriate definition (and whether it is better to use normalized Laplacian) is not the main topic of this work.

One may reformulate the supervised learning method (1) to have the same form as the collective learning method (2). We define a kernel function $k(x, x') = \psi(x)^T \psi(x')$, and let the kernel gram matrix be $K = [k(X_k, X_{k'})]_{k,k'=1,\ldots,m}$, and

$$\hat{f} = \arg\min_{f \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} L(f_i, Y_i) + \frac{\lambda}{2} f^T K^{-1} f \right]. \quad (4)$$

Then it is known that the supervised learning method (1) and the collective inference method (4) are equivalent on the graph: $\hat{f}_k = \hat{p}(X_k)$ for $k = 1, \ldots, m$. For example, see [15] for a proof. If we replace the regularizer $\mathcal{L}$ by the inverse of the kernel gram matrix K: $\mathcal{L} = K^{-1}$, then the

reformulation (4) of supervised learning and graph learning formulation (2) have the same form. Computationally, (1) is far more efficient than (4). However, the collective inference formulation is conceptually useful when we combine local information $\psi(x)$ in (1) with link information in (2).

5. GRAPH LEARNING WITH LOCAL AND LINK INFORMATION

In the graph learning formulations (2) and (4), the underlying graph is used to form regularization conditions. In (2), a graph Laplacian operator is explicitly constructed, and in (4), kernel $k(x, x') = \psi(x)^T\psi(x')$ is used. Under this framework, a natural way to combine local information in (4) and link information in (2) is to combine these regularization conditions. In this paper, we consider two possible ways to do so: using linear combinations of the regularizers or linear combinations of the associated kernels.

5.1 Linear regularizer combination

A natural way to combine local information with global link structure is to linearly combine the regularizers:

$$\hat{f} = \arg\min_{f\in R^m}\left[\frac{1}{n}\sum_{i=1}^{n} L(f_i, Y_i) + \frac{\lambda}{2} f^T K^{-1} f + \frac{\lambda'}{2} f^T \mathcal{L} f\right].$$

In this formulation, $f_k \approx f_{k'}$ either when k and k' have similar local information contents (that is, $\psi(X_k) \approx \psi(X_{k'})$) or when k and k' are close to each other on the graph. Therefore the resulting method effectively explores the local information in addition to the link structure on the graph.

However, a disadvantage of the above method is that K is a dense $m \times m$ matrix. The computational cost of inverting K is m^3. Even if we use iterative algorithms, the complexity is at least $O(m^2)$, which is not practical for large scale classification. Therefore we need to rewrite it in a form which can be solved more efficiently. In order to address this issue, we shall replace K by $(K + \mu I)$, where $\mu > 0$ is a small tuning parameter that is close to zero. The small parameter μ is often introduced to make the system strictly positive definite, and thus more stable (for example, [17, 15]). In this work, we use a small number $\mu = 0.01$ which is consistent with the literature. We can now consider the following formulation that contains the tuning parameter μ:

$$\hat{f} = \arg\min_{f\in R^m}\left[\frac{1}{n}\sum_{i=1}^{n} L(f_i, Y_i) + \frac{\lambda}{2} f^T (K+\mu I)^{-1} f + \frac{\lambda'}{2} f^T \mathcal{L} f\right]. \quad (5)$$

The following fact is used to reformulate (5).

PROPOSITION 1. *Let $K = [\psi(X_k)^T\psi(X_{k'})]_{k,k'=1,\ldots,m}$, then*

$$f^T(K+\mu I)^{-1} f = \min_{w}\left[w^2 + \mu^{-1}\sum_{k=1}^{m}(f_k - w^T\psi(X_k))^2\right].$$

PROOF. The minimum on the right hand side can be achieved at

$$w = (XX^T + \mu I)^{-1} X f, \quad \text{where } X = [\psi(X_1), \ldots, \psi(X_m)].$$

Substituting this into the right hand side, we obtain

$$\begin{aligned}
&w^2 + \mu^{-1}\sum_{k=1}^{m}(f_k - w^T\psi(X_k))^2 \\
=&f^T X^T (XX^T + \mu I)^{-2} X f \\
&\quad + \mu^{-1}(f - X^T(XX^T + \mu I)^{-1} X f)^2 \\
=&f^T K (K+\mu I)^{-2} f + \mu^{-1} f^T (I - K(K+\mu I)^{-1})^2 f \\
=&f^T (K+\mu I)^{-1} f.
\end{aligned}$$

□

Using Proposition 1, we can now rewrite (5) as

$$\begin{aligned}
[\hat{f}, \hat{v}, \hat{w}] = \arg\min_{f,v,w}&\left[\frac{1}{n}\sum_{i=1}^{n} L(f_i, Y_i)\right. \\
&\left. + \frac{\lambda}{2}(w^2 + v^2) + \frac{\lambda'}{2} f^T \mathcal{L} f\right] \quad (6) \\
\text{s.t.}\quad & f_k = w^T\psi(X_k) + v_k\sqrt{\mu} \quad (k = 1, \ldots, m).
\end{aligned}$$

This is our final formulation for combining local and link information using a linear combination of the corresponding regularizers. Note that in this approach, the role of the stabilizing parameter μ can be regarded as adding a feature $\sqrt{\mu}$ to each specific node. The method has the following two important properties. First, it avoids the construction of the dense matrix K, and thus can be solved more efficiently. Second, the optimization problem is convex in f and w, which implies there is only one global optimal solution. In contrast, earlier works in relational learning do not lead to a global convex optimization problem, and thus are more difficult to solve.

The formulation (6) integrates the following ideas for the predictive vector f on graph: f fits well on the labeled data: $\sum_i L(f_i, Y_i)$ is small; f is approximately a linear function of the local features; $\mu v^2 = \sum_k (f_k - w^T\psi(X_k))^2$ is small. f is smooth on the graph: $f^T\mathcal{L}f$ is small.

5.2 Linear kernel combination

A second way to combine local information with global link structure is to linearly combine the corresponding kernels, which is a standard practice in the kernel learning literature. Since the kernel matrix is the inverse of the regularization operator (e.g. see remarks at the end of Section 4), we have the following method that linearly combines the kernels:

$$\hat{f} = \arg\min_{f\in R^m}\left[\frac{1}{n}\sum_{i=1}^{n} L(f_i, Y_i) + \frac{1}{2} f^T (\lambda^{-1} K + \lambda'^{-1}\mathcal{L}^{-1})^{-1} f\right]. \quad (7)$$

we use the following result to rewrite this formulation.

PROPOSITION 2. *Let $K = K_1 + K_2$, we have*

$$f^T K^{-1} f = \min_{f_1, f_2 \in R^m}\left\{f_1^T K_1^{-1} f_1 + f_2^T K_2^{-1} f_2 : f_1 + f_2 = f\right\}.$$

PROOF. Consider the optimal solutions f_1 and f_2 on the right hand side. There is a Lagrangian multiplier $\beta \in R^m$

such that f_1 and f_2 minimize the unconstrained problem:

$$f_1^T K_1^{-1} f_1 + f_2^T K_2^{-1} f_2 + 2\beta^T (f_1 + f_2).$$

Taking derivative, we obtain the condition $f_1 = K_1\beta$, $f_2 = K_2\beta$, and thus $\beta = (K_1 + K_2)^{-1} f$. Substituting this representation into the right hand side of the equation in Proposition 2 gives the left hand side. □

Now by using Proposition 2 with $K_1 = \lambda^{-1}K$ and $K_2 = \lambda'^{-1}\mathcal{L}^{-1}$, we can rewrite (7) as

$$[\hat{f}, \hat{f}', \hat{w}] = \arg\min_{f,f',w} \left[\frac{1}{n}\sum_{i=1}^{n} L(f_i, Y_i) + \frac{\lambda}{2}w^2 + \frac{\lambda'}{2} f'^T \mathcal{L} f' \right]$$
$$\text{s.t.} \quad f_k = w^T \psi(X_k) + f'_k \quad (k = 1, \ldots, m).$$

Similar to (6), we may introduce a stabilizing parameter μ and obtain the following formulation:

$$[\hat{f}, \hat{v}, \hat{w}] = \arg\min_{f,v,w} \left[\frac{1}{n}\sum_{i=1}^{n} L(f_i, Y_i) + \frac{\lambda}{2}(w^2 + v^2) + \frac{\lambda'}{2}\mu v^T \mathcal{L} v \right]$$
$$\text{s.t.} \quad f_k = w^T \psi(X_k) + v_k\sqrt{\mu} \quad (k = 1, \ldots, m). \tag{8}$$

Equation (8) is similar to (6) except for the graph regularization term involving $\mathcal{L}$. In (6), the regularization is on the function $f = w^T\psi(X_k) + v_k\sqrt{\mu}$, while in (8), the regularization is only on the second component $v_k\sqrt{\mu}$.

Since regularization restricts the parameter space, we know that the formulation (8) is more expressive than (6). Instead of assuming the final predictor to be smooth on the graph as in (6), the method in (8) can be regarded as adding a feature to the predictor that is smooth on the graph. In this sense, when the quality of the graph structure is high, that is, when most links in the graph are within the same class, it is better to use (6). On the other hand, when there are many links that connect different classes, then (8) is preferred.

6. NUMERICAL SOLUTION

There are several ways to derive optimization procedures for this problem. In this paper, we consider the dual training method which has the advantage of minimum storage requirement. Using the representation of $\mathcal{L}$ in (3), both (6) and (8) can be written in the following generalized form:

$$\hat{f}_i = \hat{u}^T \phi_i \quad (i = 1, \ldots, m),$$
$$\hat{u} = \arg\min_{u} \left[\frac{1}{n}\sum_{i=1}^{n} L(u^T\phi_i, Y_i) + \frac{\lambda'}{2}\sum_{(k,k')\in E} c_{k,k'}(u^T\phi_{k,k'})^2 + \frac{\lambda}{2}u^2 \right]. \tag{9}$$

Let e_k be the vector in R^m with zero entries except for the k-th entry which is one. For (6), we can let $u^T = [w^T, v^T]$, $\phi_i^T = [\psi(X_i)^T, \sqrt{\mu}e_i^T]$, and $\phi_{k,k'} = \phi_k - \phi_{k'}$. For (8), we can let $u^T = [w^T, v^T]$, $\phi_i^T = [\psi(X_i)^T, \sqrt{\mu}e_i^T]$, and $\phi_{k,k'} = [0, e_k - e_{k'}]$. It is easy to verify that under such notation, (6) and (8) become special cases of (9). It follows that we only need to develop a computational method for (9).

Since the number of edges in E is often large, we will not be able to store all vectors $\phi_{k,k'}$ in memory. Therefore in our computational procedure, we do not explicitly store the vectors $\phi_{k,k'}$ in order to solve (6). However, our algorithm requires that local vectors ϕ_i can be stored in memory for all $i = 1, \ldots, m$. If this is not possible, then we have to use simplifications. For example, one may simply subsample the graph by removing some nodes. Another possible simplification of (9) is to compute a weight vector $\hat{w}$ based on (1), and then compute a feature $\hat{w}^T\psi(x)$ for each document x. Now, instead of using all features $\psi(x)$, we use a simplified feature $\hat{w}^T\psi(x)$, which requires less memory. Therefore in the following, we assume that the local vectors ϕ_i can be stored in memory for all $i = 1, \ldots, m$, although we may not be able to store all $\phi_{k,k'}$.

A simple idea to avoid storing $\phi_{k,k'}$ is to use the stochastic gradient descent algorithm, where we look at one data point i, or one edge (k, k') at a time, and update the weight vector u based on the gradient of the cost function at the examined point. A closely related method, which we shall employ here, is to solve the dual formulation of (9). It can be shown that the weight vector $\hat{u}$ in (9) can be obtained using the following dual formulation:

$$[\hat{u}, \hat{\alpha}] = \arg\min_{u,\alpha} \left[\frac{1}{n}\sum_{i=1}^{n} L_D(-\lambda n\alpha_i, Y_i) + \frac{\lambda^2}{2\lambda'}\sum_{(k,k')\in E} c_{k,k'}^{-1}\alpha_{k,k'}^2 + \frac{\lambda}{2}u^2 \right] \tag{10}$$
$$\text{s.t.} \quad u = \sum_{i=1}^{n}\alpha_i\phi_i + \sum_{(k,k')\in E}\alpha_{k,k'}\phi_{k,k'}.$$

The function $L_D(a, y)$ is the convex dual of L, defined as $L_D(a, y) = \sup_{f\in R}[af - L(f, y)]$. The set of variables α is often referred to as the dual variable, while u is the primal variable. The idea of dual algorithm is to vary one α_i or $\alpha_{k,k'}$ at a time, while keeping the remaining dual variables fixed. We always set the primal variable $u = \sum_{i=1}^{n}\alpha_i\phi_i + \sum_{(k,k')\in E}\alpha_{k,k'}\phi_{k,k'}$. There is a dual variable for each node, and a dual variable for each edge. There are two kind of dual updates, corresponding to node dual-variable update and edge dual-variable update. We have the algorithm in Table 1 for solving equation (10). The parameter $\eta \in (0, 1)$ is introduced to make the algorithm more stable. The parameter is not critical, and we use 0.1.

Table 1: Dual Algorithm

initialize primal weight vector $u \leftarrow 0$

initialize dual variables $\alpha_i \leftarrow 0$, $\alpha_{k,k'} \leftarrow 0$

for $\ell = 1, \cdots, L$

  for each $i = 1, \ldots, n$

    approximately solve

    $\min_{\Delta\alpha_i} \left[L_D(-\lambda n(\alpha_i + \Delta\alpha_i), Y_i) + \frac{\lambda n}{2}(u + \Delta\alpha_i\phi_i)^2 \right]$

    update $\alpha_i \leftarrow \alpha_i + \Delta\alpha_i$

    update $u \leftarrow u + \Delta\alpha_i\phi_i$

  for each $(k, k') \in E$

    let $\Delta\alpha_{k,k'} \leftarrow -\eta\frac{\lambda\alpha_{k,k'} + \lambda' c_{k,k'} u^T\phi_{k,k'}}{\lambda + \lambda' c_{k,k'}\phi_{k,k'}^T\phi_{k,k'}}$

    update $\alpha_{k,k'} \leftarrow \alpha_{k,k'} + \Delta\alpha_{k,k'}$

    update $u \leftarrow u + \Delta\alpha_{k,k'}\phi_{k,k'}$

let $\hat{f}_i = u^T\phi_i$ $(i = 1, \cdots, m)$

7. EXPERIMENTS

We use three document collections: two hyperlinked web page collections, WebKB (http://www.cs.cmu.edu/ webkb/) [4] and Yahoo! Directory (http://www.yahoo.com/), and Cora (http://www.cs.umass.edu/ mccallum/code-data.html) — a collection of scientific articles with citation information [9]. We have found that co-citation graphs derived from the original directed hyperlink (citation) graphs often give better performance within our framework, and in the rest of the paper it is assumed we use co-citation graphs, unless otherwise noted. Two documents are connected by an (undirected) co-citation edge, if there exists a third page which links to both documents. The exact meaning of "exists" varies across the datasets we use, and will be described below. The word features are formed (after removing html tags when necessary) by converting upper-case characters to lower-case characters, removing punctuation, tokenizing by white space characters and removing stop words. We use full word forms.

The WebKB dataset consists of 8,275 web pages crawled from university web sites and belonging to seven functional, as opposed to topical, categories (course, department, faculty, project, staff, student and other). The vocabulary consists of 20,000 most frequent words. To obtain a denser co-citation graph we have used Yahoo!'s database to retrieve current inlinks, where pages used to derive the co-citation graph do not necessarily belong to the set of WebKB pages. The number of (subsampled) edges in the co-citation graph used in our experiments is 1,143,716. Because we use additional information in graph generation, the results are not comparable to earlier published experiments with this dataset, and should be interpreted only in the context of comparison to word features within the proposed framework.

The Yahoo! Directory dataset is the largest of three, and consists of 22,969 web pages with assignments into one of the 13 top level topical directory categories (for example, arts, business and education). The vocabulary consists of 50,000 most frequent words. The number of (subsampled) edges in the co-citation graph is 1,170,029. As in the previous dataset generation, the in-neighbor pages used in creating the Yahoo! Directory co-citation graph do not have to necessarily belong to the initial set of 22,969 pages and were used for graph generation only.

The Cora collection contains 30,714 computer science papers with available class assignments. We experiment with the class structure containing 10 top level categories. The vocabulary consists of 20,000 most frequent words. The co-citation graph has 259,298 edges. Since in-coming citations from external papers were unavailable, and in contrast to the previous two datasets, the edges were formed using only the citations among the documents in this dataset. In addition to the above Cora data which we refer to as Cora-cocit, we also include a version of Cora with direct citations, which we call Cora-direct. There are 225,026 documents in this data, and graph contains 714,266 directed edges. The text and class labels of documents in Cora-direct which do not appear in Cora-cocit are not available.

We randomly split the labeled data into two parts: 50% for training and another 50% for testing. We draw five runs and report test set averages and standard deviations. We use the logistic regression loss function $L(f, y) = \ln(1 + \exp(-fy))$. The best λ value can be found through cross validation in the training data using (1) without the graph structure. We then fix this λ for all other configurations (with graph structure). We take $\mu = 0.01$ as mentioned earlier. In the experiments, we simply set λ' such that $\lambda' n = 0.01$ without additional optimization. Therefore better results might be possible with optimized λ'. We compare the following methods: reg-comb (reg): method to combine text features with link structures using (6); ker-comb (ker): method to combine text features with link structures using (8); text-only (txt): method that only uses text features (1); graph-only (gra): method that only uses link structures (2). The computational method presented in Table 1 easily scales to the datasets used in this paper. A typical run for each dataset takes well under an hour on a standard PC. With appropriate engineering, the method is capable of handling datasets at a significantly larger scale.

Table 2: Classification accuracy (mean ± std-dev %)

	WebKB	Yahoo! Dir	Cora-cocit	Cora-direct
reg	90.9 ± 0.4	72.0 ± 0.4	78.1 ± 0.2	79.6 ± 0.5
ker	89.2 ± 0.5	66.5 ± 0.3	77.8 ± 0.2	82.0 ± 0.2
txt	89.2 ± 0.5	66.2 ± 0.3	73.9 ± 0.3	73.9 ± 0.2
gra	64.0 ± 0.5	40.1 ± 0.3	55.8 ± 0.4	78.8 ± 0.1

Table 2 shows the multi-category classification accuracy for different methods across datasets. We noticed the reg-comb method consistently outperforms either text-only or graph-only methods. The performance of the ker-comb method is not as consistent as that of reg-comb. However, it never decreases the performance, while it significantly improves the text-only or graph-only baselines for the two Cora datasets. It is also not surprising that text often carries more information than the link structure. Therefore the text only method usually out-performs the graph-only method. The only exception is the Cora-direct dataset, where link information turns out to be more useful than text information.

In our experiments, the reg-comb method usually achieves better performance than the ker-comb method. As we have pointed out, we did not optimize the regularization parameter λ' using cross validation on the training data. This might have affected the performance of ker-comb. One may also study the quality of the underlying graph to obtain additional insights on what method works better with what kind of graph. Table 3 shows the binary-classification performance of the graph-only method (and text-only method) at the zero decision threshold, micro-averaged over the categories. We use the standard performance measures for text categorization: precision, recall, and F-measure. The micro-averaged performance measures are computed using statistics summed over the confusion matrices of all categories. The table shows that the graphs have relatively high precision, which implies that the links are reliable. As we commented earlier, the reg-comb method will be able to take advantage of such graphs to achieve good performance. We also observe that the graph-only method yields low recalls, implying that although the graph structures can be used to reliably label part of the document collections, many documents cannot be reliably labeled by graph alone (and thus text information becomes helpful here). A special case is Cora-direct, which has relatively high precision and recall. We believe that high recall is one reason for ker-comb to perform well on this data. The precision and recall figures

of the graph-only classifier give good characterizations of the underlying graph structure. They have implications on the performance of different combination methods. A useful direction is to develop better characterizations of graph-structures, and then design suitable combination methods accordingly. For comparison, we also include the precision (P), recall (R), and F-measure (F) figures for the text-only method in Table 3.

Table 3: Micro-averaged Binary Classification Performance (mean ± std-dev %)

Graph-only Method				
	WebKB	Yahoo! Dir	Cora-cocit	Cora-direct
P	99.6 ± 0.2	90.1 ± 0.5	79.8 ± 0.5	83.0 ± 0.3
R	58.0 ± 0.4	30.0 ± 0.4	33.4 ± 0.4	74.0 ± 0.3
F	73.3 ± 0.3	45.0 ± 0.5	47.1 ± 0.4	78.3 ± 0.3

Text-only Method				
	WebKB	Yahoo! Dir	Cora-cocit	Cora-direct
P	90.8 ± 0.6	73.8 ± 0.5	89.1 ± 0.4	88.7 ± 0.3
R	82.8 ± 0.6	50.0 ± 0.3	58.4 ± 0.3	57.8 ± 0.3
F	86.6 ± 0.6	59.6 ± 0.4	70.5 ± 0.2	70.0 ± 0.2

Table 4 shows the predictive performance of the text-only method versus the graph-only method, where TT is the percentage of test data predicted correctly by both methods; TF is the percentage of test data only predicted correctly by the text-only method; FT is the percentage of test data only predicted correctly by the graph-only method; and FF is the percentage of test data predicted incorrectly by both methods.

Table 4: Text-only versus Graph-only (%)

	WebKB	Yahoo! Dir	Cora-cocit	Cora-direct
TT	57.9	28.3	48.3	64.5
TF	31.3	37.9	25.6	9.4
FT	6.2	11.8	7.5	14.3
FF	4.6	22.0	18.6	11.8

8. CONCLUSION

This paper introduces a novel method for learning from both text and graph structures, applicable to the problem of collective inference for hypertext document categorization. This method is based on the graph regularization formulation of kernel methods, recently proposed in the machine learning literature. An advantage of this method compared to earlier approaches from the relational learning literature is that it leads to a well-formed convex optimization problem. Therefore it has a global optimal solution that can be efficiently computed. Moreover, the method is equivalent to the standard kernel method, with an appropriately defined kernel based on the graph. Therefore existing theoretical and algorithmic results can be directly applied. Experimental results show that combining text features and link information leads to improved accuracy in the tasks we studied.

9. REFERENCES

[1] A. Argyriou, M. Herbster, and M. Pontil. Combining graph Laplacians for semi-supervised learning. In *NIPS'05*, 2006.

[2] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, Special Issue on Clustering:209–239, 2004.

[3] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD'98*, 1998.

[4] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Poceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[5] M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43:97–119, 2001.

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

[7] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *KDD'04*, 2004.

[8] S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. Technical Report CeDER-04-08, Stern School of Business, New York University, 2004. Submitted to Journal of Machine Learning Research.

[9] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000. www.research.whizbang.com/data.

[10] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[11] A. Popescul and L. Ungar. Cluster-based concept invention for statistical relational learning. In *Proceedings of SIGKDD-2004*, 2004.

[12] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 1:1–9, 2001.

[13] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *NIPS 2001*, 2002.

[14] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, 1:69–90, 1999.

[15] T. Zhang and R. K. Ando. Analysis of spectral kernel design based semi-supervised learning. In *NIPS'05*, 2006.

[16] T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2001.

[17] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schlkopf. Learning with local and global consistency. In *NIPS 2003*, pages 321–328, 2004.

[18] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML 2003*, 2003.