Ranking on Graph Data

Shivani Agarwal SHIVANI@CSAIL.MIT.EDU
Computer Science and AI Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Abstract

In ranking, one is given examples of order relationships among objects, and the goal is to learn from these examples a real-valued ranking function that induces a ranking or ordering over the object space. We consider the problem of learning such a ranking function when the data is represented as a graph, in which vertices correspond to objects and edges encode similarities between objects. Building on recent developments in regularization theory for graphs and corresponding Laplacian-based methods for classification, we develop an algorithmic framework for learning ranking functions on graph data. We provide generalization guarantees for our algorithms via recent results based on the notion of algorithmic stability, and give experimental evidence of the potential benefits of our framework.

1. Introduction

The problem of ranking, in which the goal is to learn a real-valued ranking function that induces a ranking or ordering over an instance space, has recently gained much attention in machine learning (Cohen et al., 1999; Herbrich et al., 2000; Crammer & Singer, 2002; Freund et al., 2003). In developing algorithms for ranking, the main form of data that has been considered so far is vector-valued data, *i.e.*, the algorithm receives as input a finite number of objects in some Euclidean space $\mathbb{R}^n$, together with examples of order relationships or preferences among them, and the goal is to learn from these examples a ranking function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that orders future objects accurately. (A real-valued function $f : X \rightarrow \mathbb{R}$ is considered to order/rank $x \in X$ higher than $x' \in X$ if $f(x) > f(x')$, and vice-versa.)

In this paper, we consider the problem of learning a ranking function when the data is represented in the form of a graph. Formally, we wish to develop ranking algorithms that can take as input a (weighted) graph $G = (V, E, w)$, where V is a set of vertices in which each vertex corresponds to a data point, $E \subseteq V \times V$ a set of edges connecting related data points, and $w : E \rightarrow \mathbb{R}^+$ a weight function encoding connection strengths; and, given examples of order relationships among a small number of elements in V, can learn a good ranking function $f : V \rightarrow \mathbb{R}$ over V.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

Graph representations of data are important for many applications of machine learning. For example, such representations have been shown to be useful for data that lies in a high-dimensional space but actually comes from an underlying low-dimensional manifold (Roweis & Saul, 2000; Tenenbaum et al., 2000; Belkin & Niyogi, 2004). More importantly, graphs form the most natural data representation for an increasing number of application domains in which pair-wise connections (similarities) among objects matter, and/or are easily characterized. For example, computation of similarities between biological sequences has been a central focus in several computational biology applications; indeed, in a protein ranking problem considered toward the end of this paper, the input data consists of similarity scores between pairs of proteins.

There have been several developments in theory and algorithms for learning over graph data, in the context of classification and (to some extent) regression. In our work we build on some of these recent developments – in particular, developments in regularization theory for graphs and corresponding Laplacian-based methods for classification over graph data (Belkin & Niyogi, 2004; Belkin et al., 2004; Zhou & Schölkopf, 2004; Zhou et al., 2005) – to develop an algorithmic framework for learning ranking functions on graphs, both undirected and directed.

After some preliminaries in Section 2, we describe our algorithmic framework, first for undirected graphs in Section 3, then for directed graphs in Section 4. In Section 5 we show that our algorithms can be viewed as performing regularization within a reproducing kernel Hilbert space (RKHS) derived from the graph Laplacian; this allows us to provide generalization guarantees via recent results based on the notion of algorithmic stability (Bousquet & Elisseeff, 2002; Agarwal & Niyogi, 2005). We give experimental evidence of the potential benefits of our framework in Section 6, and conclude with a discussion in Section 7.

2. Preliminaries

We are given a weighted *data graph* $G = (V, E, w)$, where $V = \{v_1, \ldots, v_n\}$ is a finite set of vertices corresponding to data points, $E \subseteq V \times V$ a set of edges, and $w : E \to \mathbb{R}^+$ a weight function, together with a small number of examples of order relationships among vertices in V (data points). The set of order relationships among elements of V can be represented as a (directed) weighted graph of its own, which we shall call the *order graph* and denote by $\Gamma = (V, \Sigma, \tau)$, where $\Sigma \subseteq V \times V$ and $\tau : \Sigma \to \mathbb{R}^+$; the interpretation is that if $(v_i, v_j) \in \Sigma$, then v_i is to be ranked higher than v_j, and the penalty for mis-ordering such a pair is given by $\tau(v_i, v_j)$. The order graph can thus be thought of as providing 'preference labels' for ranking.

The goal is to learn from $\langle G, \Gamma \rangle$ a 'good' ranking function $f : V \to \mathbb{R}$. We note that since V is assumed to be finite and known, and our goal is to learn a ranking function only over this set, our formulation of the problem of learning ranking functions on graphs falls under the setting of transductive learning (Vapnik, 1998; Joachims, 2003; Belkin & Niyogi, 2004; Zhou et al., 2004)[1]. We also note that since $|V| = n$, we can represent any function $f : V \to \mathbb{R}$ as a column vector $\mathbf{f} \in \mathbb{R}^n$ with ith element $f_i = f(v_i)$. We shall use these two representations interchangeably in the rest of the paper.

A good ranking function is a function that makes few ranking mistakes. Assuming that ties are broken at random, the (expected) ranking loss incurred by a function $f : V \to \mathbb{R}$ on a pair $(v_i, v_j) \in \Sigma$ can be written as

$$\tau(v_i, v_j) \cdot \ell(f; v_i, v_j)\,,$$

where

$$\ell(f; v_i, v_j) \;=\; \begin{cases} 1 & \text{if } f(v_i) < f(v_j) \\ \frac{1}{2} & \text{if } f(v_i) = f(v_j) \\ 0 & \text{if } f(v_i) > f(v_j)\,. \end{cases} \qquad (1)$$

The (empirical) ranking error of f with respect to the order graph Γ can therefore be expressed as

$$\hat{R}(f; \Gamma) \;=\; \frac{1}{|\Sigma|} \sum_{(v_i, v_j) \in \Sigma} \tau(v_i, v_j) \cdot \ell(f; v_i, v_j)\,. \qquad (2)$$

In the following two sections, we develop a regularization-based algorithmic framework for learning a ranking function $f_{\langle G, \Gamma \rangle}$ from $\langle G, \Gamma \rangle$. Our algorithms minimize regularized versions of a convex upper bound on the above quantity, *i.e.*, the ranking error w.r.t. the order graph Γ; the regularizers we use encourage smoothness of the learned function w.r.t. the data graph G.

[1]It is important to note that the form of ranking problem considered in (Zhou et al., 2004) is very different from that considered in this paper; in particular, the ranking problem considered in (Zhou et al., 2004), in which the goal is to learn to rank 'positive' objects higher than 'negative' objects from examples of positive objects only, *cannot* be formulated within our order graph setting.

3. Ranking on Undirected Graphs

In this section we treat the case when the data graph $G = (V, E, w)$ is undirected, *i.e.*, when E consists of *unordered* pairs of vertices in V. The case of directed graphs is treated in Section 4.

Our goal is to find a function $f : V \to \mathbb{R}$ that minimizes a suitably regularized version of the empirical ranking error $\hat{R}(f; \Gamma)$ w.r.t. the order graph Γ, *i.e.*, that minimizes a suitable combination of the empirical error and a regularization term that penalizes complex functions. However, minimizing an objective function that involves $\hat{R}(f; \Gamma)$ is an NP-hard problem, since $\hat{R}(f; \Gamma)$ is simply a weighted sum of terms comprising the step-function ranking loss $\ell(f; v_i, v_j)$ for various pairs $\{v_i, v_j\}$. Therefore, we use instead a convex upper bound on the empirical error $\hat{R}(f; \Gamma)$. In particular, we use the following convex loss, which upper bounds $\ell(f; v_i, v_j)$:

$$\ell_h(f; v_i, v_j) \;=\; \big(1 - (f(v_i) - f(v_j))\big)_+\,, \qquad (3)$$

where $a_+ = a$ if $a > 0$ and $a_+ = 0$ otherwise. We refer to ℓ_h as the *hinge* ranking loss due to its similarity to the hinge loss used in classification. We can now define the following empirical ℓ_h-error, which is convex in f and upper bounds $\hat{R}(f; \Gamma)$:

$$\hat{R}_{\ell_h}(f; \Gamma) \;=\; \frac{1}{|\Sigma|} \sum_{(v_i, v_j) \in \Sigma} \tau(v_i, v_j) \cdot \ell_h(f; v_i, v_j)\,. \qquad (4)$$

Our goal, then, is to minimize a regularized version of $\hat{R}_{\ell_h}(f; \Gamma)$; in other words, we want to find a function $f_{\langle G, \Gamma \rangle} : V \to \mathbb{R}$ that solves the following optimization problem for some suitable regularizer $\mathcal{S}(f)$ (and an appropriate regularization parameter $\lambda > 0$):

$$\min_{f : V \to \mathbb{R}} \Big\{ \hat{R}_{\ell_h}(f; \Gamma) + \lambda \mathcal{S}(f) \Big\}\,. \qquad (5)$$

What would make a good regularizer for real-valued functions defined on the vertices of an undirected graph? It turns out this question has been studied in considerable depth in recent years, and some answers are readily available (Belkin & Niyogi, 2004; Belkin et al., 2004; Zhou & Schölkopf, 2004).

A suitable measure of regularization on functions $f : V \to \mathbb{R}$ would be a measure of smoothness w.r.t. the graph G; in other words, a good function f would be one whose value does not vary rapidly across vertices that are closely related. Let $d : V \to \mathbb{R}^+ \cup \{0\}$ be the degree function of G:

$$d(v_i) \;=\; \sum_{j : \{v_i, v_j\} \in E} w(v_i, v_j)\,. \qquad (6)$$

We shall assume that $d(v_i) > 0$ for all i. As discussed in (Zhou & Schölkopf, 2004), if one defines the *edge derivative* of a function f along an edge $\{v_i, v_j\} \in E$ at the vertex v_i as

$$\left.\frac{\partial f}{\partial \{v_i, v_j\}}\right|_{v_i} = \sqrt{\frac{w(v_i,v_j)}{d(v_i)}}f(v_i) - \sqrt{\frac{w(v_i,v_j)}{d(v_j)}}f(v_j)\,, \tag{7}$$

then the *local variation* of f at each vertex v_i can be defined to be

$$\|\nabla_{v_i} f\| = \sqrt{\sum_{j:\{v_i,v_j\}\in E}\left(\left.\frac{\partial f}{\partial\{v_i,v_j\}}\right|_{v_i}\right)^2}\,. \tag{8}$$

The smoothness of f can then naturally be measured by the sum of local variations at each of the vertices:

$$\mathcal{S}(f) = \frac{1}{2}\sum_{i=1}^{n}\|\nabla_{v_i} f\|^2\,. \tag{9}$$

Smooth functions f that have small local variations would receive lower values of $\mathcal{S}(f)$, and would thus be preferred by an algorithm using this quantity as a regularizer.

It turns out that the above regularizer can be re-written in terms of the Laplacian matrix $\mathbf{L}$ of the graph G. The Laplacian is defined as follows: if $\mathbf{W}$ is defined to be the $n \times n$ matrix with $W_{ij} = w(v_i, v_j)$ for $\{v_i, v_j\} \in E$ and $W_{ij} = 0$ otherwise, and $\mathbf{D}$ is a diagonal matrix with $D_{ii} = d(v_i)$, then

$$\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-1/2}\,. \tag{10}$$

As shown in (Zhou & Schölkopf, 2004), the smoothness regularizer $\mathcal{S}(f)$ defined in Eq. (9) has the following equivalent form (recall from Section 2 that we also represent $f : V\to\mathbb{R}$ as $\mathbf{f} \in \mathbb{R}^n$):

$$\mathcal{S}(f) = \mathbf{f}^T\mathbf{L}\mathbf{f}\,. \tag{11}$$

The regularizer used by Belkin et al. (2004) is very closely related: it reduces to the same form as in Eq. (11), but uses an unnormalized version of the Laplacian.

Putting everything together, our algorithm for learning from $\langle G, \Gamma\rangle$ a ranking function $f_{\langle G,\Gamma\rangle} : V\to\mathbb{R}$ thus consists of solving the following optimization problem:

$$\min_{f:V\to\mathbb{R}}\left\{\hat{R}_{\ell_h}(f;\Gamma) + \lambda\,\mathbf{f}^T\mathbf{L}\mathbf{f}\right\}\,. \tag{12}$$

In practice, the above optimization problem can be solved by reduction to a convex quadratic program, much as is done in support vector machines (SVMs). In particular, introducing a slack variable ξ_{ij} for each ordered pair $(v_i, v_j) \in \Sigma$, we can re-write the above optimization problem as follows:

$$\min_{\mathbf{f}\in\mathbb{R}^n}\left\{\frac{1}{2}\mathbf{f}^T\mathbf{L}\mathbf{f} + C\sum_{(v_i,v_j)\in\Sigma}\tau(v_i,v_j)\cdot\xi_{ij}\right\}$$

subject to

$$\begin{aligned} f_i - f_j &\geq 1 - \xi_{ij} && \forall (v_i,v_j)\in\Sigma \\ \xi_{ij} &\geq 0 && \forall (v_i,v_j)\in\Sigma\,, \end{aligned} \tag{13}$$

where $C = 1/2\lambda|\Sigma|$. On introducing Lagrange multipliers α_{ij} and β_{ij} for the above inequalities and formulating the Lagrangian dual (see, *e.g.*, (Boyd & Vandenberghe, 2004), or (Burges, 1998) for a detailed description of the use of this standard technique in SVMs), the above problem further reduces to the following (convex) quadratic program in the $|\Sigma|$ variables $\{\alpha_{ij}\}$:

$$\min_{\mathbf{f}\in\mathbb{R}^n}\left\{\frac{1}{2}\sum_{(v_i,v_j)\in\Sigma}\sum_{(v_k,v_l)\in\Sigma}\alpha_{ij}\alpha_{kl}\phi_{ijkl} - \sum_{(v_i,v_j)\in\Sigma}\alpha_{ij}\right\}$$

subject to

$$0 \leq \alpha_{ij} \leq C\,\tau(v_i,v_j) \qquad \forall (v_i,v_j)\in\Sigma\,, \tag{14}$$

where

$$\phi_{ijkl} = L^+_{ik} - L^+_{jk} - L^+_{il} + L^+_{jl}\,. \tag{15}$$

Here L^+_{ij} denotes the (i, j)th element of $\mathbf{L}^+$, the pseudo-inverse of $\mathbf{L}$. Note that the Laplacian $\mathbf{L}$ is known to be positive semi-definite, and to not be positive definite (Chung, 1997); this means it has a zero eigenvalue, and is therefore singular (Strang, 1988) (hence the need for the pseudo-inverse). It is also easy to verify from the definition that $\mathbf{D} - \mathbf{W}$ (and therefore $\mathbf{L}$) has rank smaller than n.

It can be shown that, on solving the above quadratic program for $\{\alpha_{ij}\}$, the solution $\mathbf{f}_{\langle G,\Gamma\rangle} \in \mathbb{R}^n$ to the original problem is found as

$$\mathbf{f}_{\langle G,\Gamma\rangle} = \mathbf{L}^+\big(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\big)\,, \tag{16}$$

where $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \in \mathbb{R}^n$ are given by

$$\alpha_i^+ = \sum_{j:(v_i,v_j)\in\Sigma}\alpha_{ij}\,, \qquad \alpha_i^- = \sum_{j:(v_j,v_i)\in\Sigma}\alpha_{ji}\,. \tag{17}$$

4. Ranking on Directed Graphs

The case when the data graph $G = (V, E, w)$ is directed, *i.e.*, when E consists of *ordered* pairs of vertices in V, can be treated similarly to the undirected case. In particular, the goal is the same: to find a function $f : V\to\mathbb{R}$ that minimizes a suitably regularized convex upper bound on the empirical ranking error $\hat{R}(f; \Gamma)$ w.r.t. the order graph Γ.

The convex upper bound on $\hat{R}(f; \Gamma)$ can be chosen to be the same as before, *i.e.*, to be the ℓ_h-error $\hat{R}_{\ell_h}(f; \Gamma)$. The goal is then again to solve the optimization problem given in Eq. (5), for some suitable regularizer $\mathcal{S}(f)$. This is where the technical difference lies: in the form described so far, the regularizer used in the previous section applies only to undirected graphs. Indeed, until very recently, the notion of a Laplacian matrix has been associated only with undirected graphs .

Recently, however, an analogue of the Laplacian has been proposed for directed graphs (Chung, 2005). This shares many nice properties with the Laplacian for undirected graphs, and in fact can also be derived via discrete analysis on directed graphs (Zhou et al., 2005). It is defined in terms of a random walk on the given directed graph.

Given the (weighted) directed graph G, let $d^+ : V \rightarrow \mathbb{R}^+ \cup \{0\}$ be the out-degree function of G, defined as follows:

$$d^+(v_i) = \sum_{j:(v_i,v_j)\in E} w(v_i, v_j)\,. \tag{18}$$

If the directed graph G is strongly connected and aperiodic, one can consider the random walk over G with transition probability matrix $\mathbf{P}$ defined as follows:

$$P_{ij} \;=\; \begin{cases} w(v_i,v_j)/d^+(v_i) & \text{if } (v_i,v_j) \in \mathbf{E} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

In this case, the above random walk has a unique stationary distribution $\pi : V \rightarrow (0,1]$, and the Laplacian $\mathbf{L}$ of G is defined as

$$\mathbf{L} \;=\; \mathbf{I} - \frac{\mathbf{\Pi}^{1/2}\mathbf{P}\mathbf{\Pi}^{-1/2} + \mathbf{\Pi}^{-1/2}\mathbf{P}^T\mathbf{\Pi}^{1/2}}{2}\,, \tag{20}$$

where $\mathbf{\Pi}$ is a diagonal matrix with $\Pi_{ii} = \pi(v_i)$. In the case when G is not strongly connected and aperiodic, one can use what is termed a *teleporting* random walk, which effectively allows one to jump uniformly to a random vertex with some small probability η (Zhou et al., 2005); the probability transition matrix $\mathbf{P}^{(\eta)}$ for such a walk is given by

$$P^{(\eta)}_{ij} \;=\; \begin{cases} (1-\eta)P_{ij} + \eta\frac{1}{n-1}(\mathbf{1}-\mathbf{I}) & \text{if } d^+(v_i) > 0 \\ \frac{1}{n-1}(\mathbf{1}-\mathbf{I}) & \text{otherwise,} \end{cases} \tag{21}$$

where $\mathbf{1}$ is an $n \times n$ matrix of all ones and $\mathbf{I}$ is the $n \times n$ identity matrix. Such a teleporting random walk always converges to a unique and positive stationary distribution, and therefore the Laplacian $\mathbf{L}$ for a general directed graph can be defined as in Eq. (20), using $\mathbf{P}^{(\eta)}$ and the corresponding stationary distribution in place of $\mathbf{P}$ and $\mathbf{\Pi}$.

As discussed by Zhou et al. (2005), the Laplacian matrix constructed as above can be used in exactly the same way as in the undirected case to define a smoothness regularizer $\mathcal{S}(f) = \mathbf{f}^T\mathbf{L}\mathbf{f}$ appropriate for functions defined on the vertices of a directed graph. Thus, the algorithmic framework developed for the undirected case applies in exactly the same manner to the directed case, except for the replacement with the appropriate Laplacian matrix.

5. Some Properties of Our Algorithms

In this section we study various properties of the algorithms derived in the previous two sections. In particular, we provide an alternate, RKHS-based view of the algorithms, and use this to study their stability and generalization properties. The results in this section apply to the algorithms for both undirected and directed cases, with the Laplacian $\mathbf{L}$ referring to the appropriate matrix in each case.

5.1. RKHS View

Let $\mathcal{F}$ be the column-space of $\mathbf{L}^+$, *i.e.*, $\mathcal{F}$ is the set of all vectors in $\mathbb{R}^n$ that can be expressed as a linear combination of the columns of $\mathbf{L}^+$. Recall that the column-space of any symmetric positive semi-definite (p.s.d.) matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ is an RKHS with $\mathbf{S}$ as its kernel. Since the Laplacian $\mathbf{L}$ is symmetric p.s.d. (Chung, 1997; Chung, 2005), and since the pseudo-inverse of a symmetric p.s.d. matrix is also symmetric p.s.d. (Strang, 1988), we have that $\mathbf{L}^+$ is symmetric p.s.d. Consequently, $\mathcal{F}$ is an RKHS with $\mathbf{L}^+$ as its kernel.

We shall show now a somewhat surprising and rather pleasant result: that the algorithms of Sections 3 and 4 can be viewed as performing regularization within the RKHS $\mathcal{F}$. In order to establish this, we need to show two things: first, that the algorithms always return a function in $\mathcal{F}$, and second, that the regularizer $\mathcal{S}(f) = \mathbf{f}^T\mathbf{L}\mathbf{f}$ used by the algorithms is equivalent to the (squared) norm of $\mathbf{f}$ in the RKHS $\mathcal{F}$. The first of these follows simply from the form of the solution to the optimization problem in Eq. (13); in particular, it is clear from Eq. (16) that the solution always belongs to the column-space of $\mathbf{L}^+$. To see the second of these, *i.e.*, the equivalence of the algorithmic regularizer and the RKHS norm, let $f \in \mathcal{F}$; by definition, this means there exists a coefficient vector $\mathbf{a} \in \mathbb{R}^n$ such that $\mathbf{f} = \sum_{i=1}^n a_i\mathbf{L}^+_i$, where $\mathbf{L}^+_i$ denotes the ith column of $\mathbf{L}^+$. Then we have

$$\begin{aligned} \|\mathbf{f}\|^2_{\mathcal{F}} &= \langle \mathbf{f},\mathbf{f}\rangle_{\mathcal{F}} \\ &= \sum_{i=1}^n a_i \langle \mathbf{f}, \mathbf{L}^+_i\rangle_{\mathcal{F}} \\ &= \sum_{i=1}^n a_i f_i \quad \text{(reproducing property)} \\ &= \mathbf{a}^T\mathbf{f}\,. \end{aligned} \tag{22}$$

Furthermore, we have

$$\begin{aligned} \mathcal{S}(f) &= \mathbf{f}^T\mathbf{L}\mathbf{f} \\ &= (\mathbf{a}^T\mathbf{L}^+)\mathbf{L}(\mathbf{L}^+\mathbf{a}) \\ &= \mathbf{a}^T\mathbf{L}^+\mathbf{a} \\ &= \mathbf{a}^T\mathbf{f}\,. \end{aligned} \tag{23}$$

Thus we see that $\mathcal{S}(f) = \|\mathbf{f}\|^2_{\mathcal{F}}$, and therefore our algorithms can be viewed as performing regularization within the RKHS $\mathcal{F}$.

5.2. Generalization Properties

Using the notion of algorithmic stability (Bousquet & Elisseeff, 2002), it was shown recently that, in a particular setting of the ranking problem known as the *bipartite* ranking problem, ranking algorithms that perform regularization in an RKHS (subject to some conditions) have good generalization properties (Agarwal & Niyogi, 2005). Based on the RKHS view discussed above, our algorithms benefit directly from these results in the bipartite setting.

In the bipartite setting of the ranking problem (Freund et al., 2003; Agarwal et al., 2005), the order graph Γ consists of a complete (directed) bipartite graph on some subset of the vertices in V, with a weight of 1 on all the edges involved; in other words, there exist $m_1, m_2 \in \mathbb{N}$ and $S_+ = (v_1^+, \ldots, v_{m_1}^+) \in V^{m_1}$, $S_- = (v_1^-, \ldots, v_{m_2}^-) \in V^{m_2}$ such that $\Sigma = \{(v_i^+, v_j^-)\}$ and $\tau(v_i^+, v_j^-) = 1$. The preference labels for ranking in the bipartite setting can therefore be represented simply by a sequence of 'positive' training examples $S_+ = (v_1^+, \ldots, v_{m_1}^+) \in V^{m_1}$ and a sequence of 'negative' training examples $S_- = (v_1^-, \ldots, v_{m_2}^-) \in V^{m_2}$, in place of the order graph Γ.

In studying generalization properties of algorithms for bipartite ranking, one generally assumes that 'positive' instances are drawn i.i.d. from some (unknown) distribution $\mathcal{D}_+$ over the instance space (here V) and that 'negative' instances are drawn i.i.d. from some (unknown) distribution $\mathcal{D}_-$ over the instance space (here V). The true (expected) error of a ranking function $f : V \rightarrow \mathbb{R}$ (w.r.t. the distributions $\mathcal{D}_+, \mathcal{D}_-$) is then defined as

$$R(f) \;=\; \mathbf{E}_{u \sim \mathcal{D}_+, v \sim \mathcal{D}_-} \ell(f; u, v) \,. \qquad (24)$$

Based on the results of (Agarwal & Niyogi, 2005), we can bound the expected error of the ranking function $f_{\langle G, S_+, S_- \rangle}$ learned by either of our algorithms as follows:

Theorem 1. *Let $G = (V, E, w)$ be a (directed or undirected) data graph with $|V| = n$, let $\mathbf{L}$ denote the Laplacian of G, and let $\mathbf{L}^+$ denote the pseudo-inverse of $\mathbf{L}$. Let $\kappa^2 = \max_{i \in \{1, \ldots, n\}} L_{ii}^+$, and let $m_1, m_2 \in \mathbb{N}$. Then for any $0 < \delta < 1$, with probability at least $1 - \delta$ over the draw of $(S_+, S_-) \in V^{m_1} \times V^{m_2}$, the expected ranking error of the ranking function $f_{\langle G, S_+, S_- \rangle}$ learned by the appropriate one of our algorithms (Section 3 or 4, depending on whether G is undirected or directed) is bounded by*

$$\begin{aligned} R(f_{\langle G,S_+,S_-\rangle}) \;<\; & \hat{R}_{\ell_1}(f_{\langle G,S_+,S_-\rangle}; S_+, S_-) \\ & + \frac{8\kappa^2}{\lambda}\Big(\frac{m_1+m_2}{m_1 m_2}\Big) \\ & + \Big(1 + \frac{16\kappa^2}{\lambda}\Big)\sqrt{\frac{(m_1+m_2)\ln(1/\delta)}{2 m_1 m_2}} \,, \end{aligned}$$

where

$$\hat{R}_{\ell_1}(f; S_+, S_-) \;=\; \frac{1}{m_1 m_2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \ell_1(f, v_i^+, v_j^-)$$

with

$$\ell_1(f, v_i, v_j) = \begin{cases} \big(1 - (f(v_i) - f(v_j))\big)_+ & \text{if } f(v_i) > f(v_j) \\ 1 & \text{otherwise.} \end{cases}$$

The proof follows directly from Theorems 3 and 4 of (Agarwal & Niyogi, 2005).

6. Experiments

We evaluated our graph-based ranking algorithms on ranking problems in two different application domains: a protein ranking problem in computational biology, and a document ranking problem in information retrieval.

6.1. Protein Ranking – SCOP/PSI-BLAST Data

The Structural Classification of Proteins (SCOP) database (Murzin et al., 1995) consists of a hierarchical classification of proteins, based on their 3D structure, into classes, folds, superfamilies, and families (Figure 1). Given a target protein family, one can define a natural ranking task associated with this family: proteins within the family are to be ranked highest (level 1); proteins in the same superfamily (but not the same family) are to be ranked next (level 2), followed by proteins in the same fold (but not same family or superfamily) (level 3), then proteins in the same class (level 4) and, finally, proteins outside the class (level 5). Given an order graph Γ that contains examples of such ranking preferences, the goal is to learn a ranking over all the proteins. This can be viewed as a *5-partite* ranking problem, in which the order graph Γ is 5-partite, *i.e.*, the vertices of the graph (proteins) can be divided into 5 disjoint subsets such that there are no edges (ranking preferences) within the subsets.

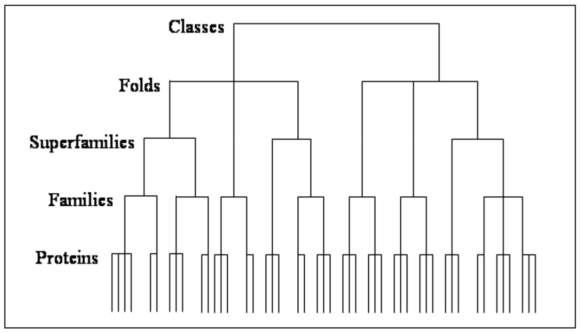

Figure 1. Proteins in the SCOP database are classified hierarchically into clases, folds, superfamilies, and families. Given a target protein family, there is a natural 5-partite ranking task associated with the family. (See text for details.)

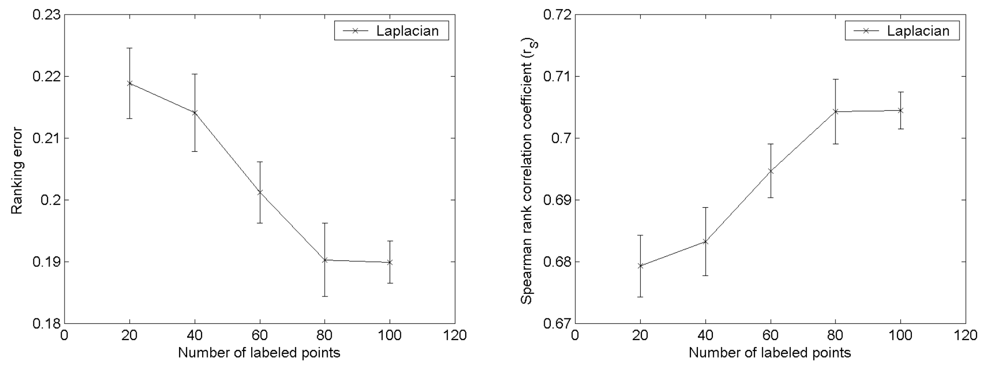

Figure 2. Performance of our algorithm on the 5-partite *b.1.1.1* protein ranking task. Left: Ranking error (test set). Right: Spearman rank-order correlation (test set). Each point is an average over 10 random splits; error bars show standard error. (See text for details.)

The SCOP/PSI-BLAST data set[2] consists of (dis)similarity scores between pairs of SCOP proteins, computed using the popular PSI-BLAST sequence search and alignment tool (Altschul et al., 1997). This data set has been used in another kind of protein ranking task in (Weston et al., 2004)[3]. In our experiments, we used a subset of the SCOP data consisting of 3314 proteins from two classes (*all alpha* and *all beta*). Following (Weston et al., 2004), we converted the dissimilarity scores (E-values returned by PSI-BLAST) to similarity scores by taking $w(v_i, v_j) = \exp(-E_{ij}/100)$, where E_{ij} denotes the E-value assigned by PSI-BLAST to protein v_j given query v_i. The similarity scores were then used to define a weighted data graph over the 3314 proteins. The scores are actually asymmetric; consequently, the data graph G in this case was a directed graph.

We took the largest protein family data set (an *all beta* family of *V set domains (antibody variable like)*, denoted as *b.1.1.1* in the data set, consisting of 403 proteins) as our target, and evaluated our algorithm on the ranking task associated with this family. The ranking task was defined as described above, with $\tau(v_i, v_j) = \big(\text{level}(j) - \text{level}(i)\big)_+$. We used our Laplacian-based ranking algorithm from Section 4; in constructing the graph Laplacian, we used a teleporting random walk with $\eta = 0.01$ (see Section 4).

The results are shown in Figure 2. Experiments were conducted with varying numbers of labeled examples; the results for each number are averaged over 10 trials (random train/test splits, subject to a fixed proportion of proteins from the 5 rank levels). Error bars show standard error. For each train/test split, the order graph consisted of the appropriate τ values for edges within the training set; a similar order graph over the test set was used to compute the ranking error plotted in the first graph. We also computed in each case the Spearman rank-order correlation between the "true" ranking of the test proteins (ranks between 1 and 5) and the learned ranking; this is plotted in the second graph. The value of the parameter C in the algorithm was selected from the set $\{0.01, 0.1, 1, 10, 100\}$ using 5-fold cross validation in each trial; to compensate for the small training set sizes, the cross-validation was done by dividing the training *edges* into 5 groups, rather than the vertices.

The above protein ranking task is an example of a ranking problem which, due to the graph-based input data representation, could not be tackled using previous approaches. In order to compare our approach to others, we used next a data set in which the data could be represented both as vector data and as graph data.

6.2. Document Ranking – 20 Newsgroups Data

The 20 newsgroups data set[4] consists of documents comprised of newsgroup messages, classified according to newsgroup. We used the "mini" version of the data set in our experiments, which contains a total of 2000 messages, 100 each from 20 different newsgroups. These newsgroups can be grouped together into categories based on subject matter, allowing for a hierarchical classification. This again leads to a natural ranking task associated with any target newsgroup: messages from the given newsgroup are to be ranked highest (level 1), followed by messages from other newsgroups in the same category (level 2), followed finally by messages in other categories (level 3). This can be viewed as a *3-partite* ranking task.

We categorized the 20 newsgroups according to the recommendation given by Jason Rennie on his webpage,

[2]Available at *www.kyb.tuebingen.mpg.de/bs/people/weston/rankprot/supplement.html*

[3]Again, it is important to note that the ranking problem in (Weston et al., 2004), as in (Zhou et al., 2004), is very different from those considered in this paper; in particular, the ranking tasks there are defined by a single protein as a query, and cannot be formulated within our order graph setting.

[4]Available at *www.ics.uci.edu/ kdd/databases/20newsgroups/20newsgroups.html*

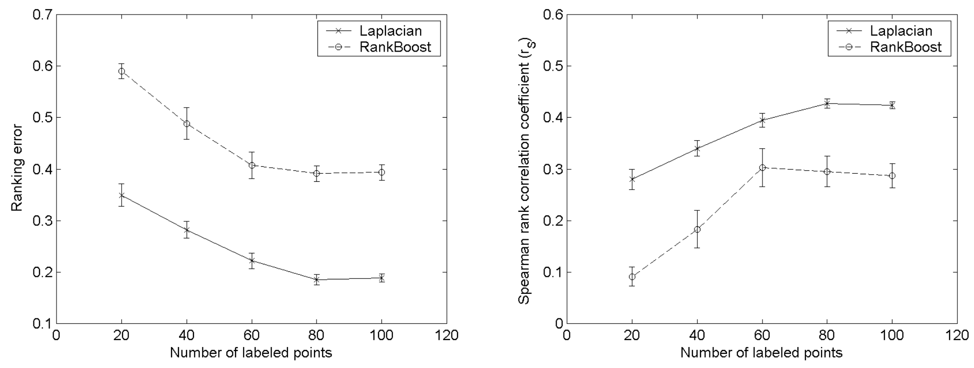

Figure 3. Comparison of our algorithm with RankBoost on the 3-partite *alt.atheism* newsgroup document ranking task. Left: Ranking error (test set). Right: Spearman rank-order correlation (test set). Each point is an average over 10 random splits; error bars show standard error. (See text for details.)

http://people.csail.mit.edu/jrennie/20Newsgroups/, and chose the *alt.atheism* newsgroup as our target. There were two other newsgroups in the same category as the target, namely *soc.religion.christian* and *talk.religion.misc*.

Following (Belkin & Niyogi, 2004), we tokenized the documents using the Rainbow software package written by Andrew McCallum, using a stop list of approximately 500 common words and removing message headers. The vector representation of each message then consisted of the counts of the most frequent 6000 words, normalized so as to sum to 1. The graph representation of the data was derived from the vector values; in particular, we constructed an undirected graph over the 2000 documents using Gaussian similarity weights given by $w(v_i, v_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2)$, where $\mathbf{x}_i$ denotes the vector representation of document v_i. We applied our Laplacian-based ranking algorithm from Section 3, and compared this to the RankBoost algorithm of (Freund et al., 2003), using threshold rankers with range $\{0, 1\}$ (similar to boosted stumps) as weak rankings.

The results are shown in Figure 3. The results for each number are averaged over 10 trials (random train/test splits, subject to equal numbers from all newsgroups). RankBoost was run for 100 rounds in each trial (increasing the number of rounds further did not yield any improvement in performance). Similarly to the protein ranking experiments, the value of the parameter C in each trial was selected from the set $\{0.01, 0.1, 1, 10, 100\}$ using 4-fold cross validation.

It should be noted that this is not truly a fair comparison, since RankBoost operates in a strictly supervised setting, *i.e.*, it sees only the training points, while the Laplacian algorithm operates in a transductive setting and sees the complete data graph. What it shows, however, is that for domains in which ranking labels are limited but similarity information about data points can be obtained, one can gain considerably by using an algorithm that can exploit this information.

7. Discussion

The goal of this paper has been to extend the growing repertoire of ranking algorithms to data represented in the form of a (similarity) graph. Building on recent developments in regularization theory for graphs and corresponding Laplacian-based methods for classification, we have developed an algorithmic framework for learning ranking functions on both undirected and directed graphs.

Our algorithms have an SVM-like flavour in their formulations; indeed, they can be viewed as minimizing a regularized ranking error within a reproducing kernel Hilbert space (RKHS). From a theoretical standpoint, this means that they benefit from theoretical results such as those establishing stability and generalization properties of algorithms that perform regularization within an RKHS. From a practical standpoint, it means that the implementation of these algorithms can benefit from the large variety of techniques that have been developed for scaling SVMs to large data sets (*e.g.*, (Joachims, 1999; Platt, 1999)).

The formulation of the graph ranking problem that we have focused on in this paper falls under the setting of transductive learning. It should be possible to use out-of-sample extension techniques, such as those of (Sindhwani et al., 2005), to extend our framework to a semi-supervised learning setting.

Acknowledgments

The author would like to thank Partha Niyogi for stimulating discussions on many topics related to this work, including manifold and graph Laplacians and regularization theory for continuous and discrete spaces; these discussions took place mostly at the Chicago Machine Learning Summer School in May 2005. The author also benefited from a lecture by Fan Chung in the MIT Applied Mathematics Colloquium series in October 2005, in which she talked about Laplacians and Cheeger inequalities for directed graphs. The author is supported by a postdoctoral lecturership in the Electrical Engineering & Computer Science Department at MIT.

References

Agarwal, S., Graepel, T., Herbrich, R., Har-Peled, S., & Roth, D. (2005). Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 393–425.

Agarwal, S., & Niyogi, P. (2005). Stability and generalization of bipartite ranking algorithms. *Proceedings of the 18th Annual Conference on Learning Theory*.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, *25*, 3389–3402.

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. *Proceedings of the 17th Annual Conference on Learning Theory*.

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Machine Learning*, *56*, 209–239.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, *2*, 499–526.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Chung, F. R. K. (1997). *Spectral graph theory*. American Mathematical Society.

Chung, F. R. K. (2005). Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, *9*, 1–19.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, *10*, 243–270.

Cortes, C., & Mohri, M. (2004). AUC optimization vs. error rate minimization. *Advances in Neural Information Processing Systems 16*.

Crammer, K., & Singer, Y. (2002). Pranking with ranking. *Advances in Neural Information Processing Systems 14*.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*, 933–969.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, 115–132.

Joachims, T. (1999). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *Proceedings of the 20th International Conference on Machine Learning*.

Murzin, A., Brenner, S. E., Hubbard, T., & Chothia, C. (1995). SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, *247*, 536–540.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. *Proceedings of the 22nd International Conference on Machine Learning*.

Strang, G. (1988). *Linear algebra and its applications*. Brooks Cole. 3rd edition.

Tenenbaum, J., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley

Weston, J., Eliseeff, A., Zhou, D., Leslie, C., & Noble, W. S. (2004). Protein ranking: from local to global structure in the protein similarity network. *Proceedings of the National Academy of Science*, *101*, 6559–6563.

Zhou, D., Huang, J., & Schölkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning*.

Zhou, D., & Schölkopf, B. (2004). A regularization framework for learning from graph data. *ICML Workshop on Statistical Relational Learning*.

Zhou, D., Weston, J., Gretton, A., Bousquet, O., & Schölkopf, B. (2004). Ranking on data manifolds. *Advances in Neural Information Processing Systems 16*.