Mach Learn (2010) 81: 333–357
DOI 10.1007/s10994-010-5185-8

Learning to rank on graphs

Shivani Agarwal

Received: 31 July 2008 / Revised: 27 December 2009 / Accepted: 28 March 2010 /
Published online: 29 May 2010

Abstract Graph representations of data are increasingly common. Such representations arise in a variety of applications, including computational biology, social network analysis, web applications, and many others. There has been much work in recent years on developing learning algorithms for such graph data; in particular, graph learning algorithms have been developed for both classification and regression on graphs. Here we consider graph learning problems in which the goal is not to predict labels of objects in a graph, but rather to rank the objects relative to one another; for example, one may want to rank genes in a biological network by relevance to a disease, or customers in a social network by their likelihood of being interested in a certain product. We develop algorithms for such problems of learning to rank on graphs. Our algorithms build on the graph regularization ideas developed in the context of other graph learning problems, and learn a ranking function in a reproducing kernel Hilbert space (RKHS) derived from the graph. This allows us to show attractive stability and generalization properties. Experiments on several graph ranking tasks in computational biology and in cheminformatics demonstrate the benefits of our framework.

Keywords Graphs · Networks · Ranking · Transductive learning · Graph regularization

1 Introduction

In an increasing number of applications of machine learning, it is of interest to analyze data represented in the form of a graph or network structure over objects. In computational biology, it is of interest to analyze protein interaction data, generally represented as a graph of pair-wise interactions between proteins. In social network analysis, it is of interest to

Editor: Tong Zhang.

A preliminary version of this paper appeared in the Proceedings of the 23rd International Conference on Machine Learning (ICML) in 2006.

S. Agarwal (✉)
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
e-mail: shivani@mit.edu

analyze social interaction data, in this case represented as a graph of interactions between individuals or organizations. In web applications, it is of interest to analyze hyperlink data, which results in graphs over webpages. In addition, as is now well known, if the input data resides in a high-dimensional space but comes from an underlying low-dimensional manifold, it is often best represented as a graph that captures the local geometry of the input space (Tenenbaum et al. 2000; Roweis and Saul 2000; Belkin and Niyogi 2002).

There has been much work in developing learning algorithms for such graph or network data, ranging from designing kernels for graphs (Kondor and Lafferty 2002; Smola and Kondor 2003) to developing algorithms for classification and regression on graphs (Belkin and Niyogi 2004; Belkin et al. 2004; Zhou and Schölkopf 2004; Zhou et al. 2005; Herbster et al. 2005; Johnson and Zhang 2008). In particular, we now have a well-developed theory for such learning problems on graphs, including a mathematical theory of graph regularization (Smola and Kondor 2003; Belkin et al. 2004; Zhou and Schölkopf 2004).

In many graph learning problems, however, the goal is not to predict a class or real-valued label for each object in the graph, but rather, to rank or prioritize objects in the graph relative to one another. In computational biology, one often wants to rank or prioritize genes or proteins such that those that are more likely to be involved in a given disease or biological pathway appear at the top of the ranking (Morrison et al. 2005; Aerts et al. 2006; Ma et al. 2007; Agarwal and Sengupta 2009); the top few genes or proteins can then be subjected to biological tests to further elucidate their functional and structural properties. In social network analysis, one often wants to rank individuals, for example by their likelihood of being interested in a particular product; depending on the available resources, the top k individuals for some appropriate k can then be targeted for advertising. In web applications, a common goal is to rank webpages by relevance to a query.

In this paper, we investigate the problem of learning to rank on graphs. There has been much interest in ranking problems in machine learning in recent years, both due to the fact that they are distinct from the classical learning problems of classification and regression, and due to their widespread applications, ranging from information retrieval to collaborative filtering and from computational biology to drug discovery (Cohen et al. 1999; Freund et al. 2003; Herbrich et al. 2000; Joachims 2002; Crammer and Singer 2005; Agarwal et al. 2005; Burges et al. 2005; Rudin et al. 2005; Agarwal 2006; Cortes et al. 2007; Clemencon et al. 2008; Cossock and Zhang 2008; Agarwal and Niyogi 2009; Agarwal and Sengupta 2009). Ranking problems on graphs, however, have received limited attention in the community, some notable exceptions in addition to our own earlier work (Agarwal 2006) being that of Zhou et al. (2004) and, more recently, that of Agarwal and Chakrabarti (2007).

We note that our work differs from standard graph ranking algorithms such as PageRank (Brin and Page 1998) and HITS (Kleinberg 1999). In particular, these algorithms yield a fixed ranking for any graph and do not involve any learning from examples. In contrast, we are interested in situations where one is given examples of preferences among objects in the graph (such as examples of one document being more relevant to a topic than another), and the goal is to learn a ranking over the remaining objects in the graph that takes into account these preferences; in this case, different sets of preferences will yield different rankings on the same graph. (We note that there are some 'personalized' variations of PageRank that can be biased towards certain preferred objects; however this is different from learning from general pair-wise preferences as we do here.) Our work also differs from the above-mentioned works of Zhou et al. (2004) and Agarwal and Chakrabarti (2007). In particular, Zhou et al. (2004) consider problems in which one is given only examples of 'positive' or 'query' objects (such as documents related to a specific topic), and the goal is to rank the remaining objects in the graph by relevance to these query objects. Agarwal and Chakrabarti

(2007) incorporate pair-wise preferences; however they do so using an algorithm that is derived from PageRank-based considerations and that models rankings on graphs via edge probability matrices.

Here we build on the graph regularization ideas of Smola and Kondor (2003), Belkin et al. (2004), Zhou and Schölkopf (2004), Zhou et al. (2005) to develop algorithms for learning ranking functions on graphs. The graph regularizers used in the algorithms induce a graph kernel; these algorithms can therefore be viewed as learning a ranking function in a reproducing kernel Hilbert space (RKHS) derived from the graph, which allows us to show attractive stability and generalization properties for these algorithms using recent results of Agarwal and Niyogi (2009). Experiments on several graph ranking tasks in computational biology and in cheminformatics show that our graph ranking algorithms yield better performance than the corresponding classification or regression algorithms on the same graphs. The graphs used in our experiments are shown in Fig. 1.

The paper is organized as follows. Section 2 describes formally the problem of learning to rank on graphs, and gives several examples of specific settings of such problems. Next we describe our graph ranking algorithms, first for undirected graphs in Sect. 3, then for directed graphs in Sect. 4. Section 5 discusses stability and generalization properties of our algorithms. Section 6 provides experimental results; we conclude with a discussion in Sect. 7.

2 Problem formulation

Consider the following problem setting: we are given a weighted *data graph* $G = (V, E, w)$, where $V = \{v_1, \dots, v_n\}$ is a finite set of vertices corresponding to objects or data points, $E \subseteq V \times V$ is a set of edges, and $w : E \to \mathbb{R}^+$ is a weight function; and in addition, we are given a small number of examples of preferences or order relationships among objects in V. The set of preferences or order relationships among elements of V can be represented as a (directed) weighted graph of its own, which we shall call the *preference 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 (is preferred over) v_j, and the penalty for mis-ordering such a pair is given by $\tau(v_i, v_j)$. The goal is to learn from $\langle G, \Gamma \rangle$ a ranking function $f : V \to \mathbb{R}$ that ranks accurately the remaining objects in V; here f is considered to rank objects $v_i \in V$ in descending order of the scores $f(v_i)$.

The edges E and weights w in the data graph G will typically denote similarities or interactions between objects in V, for example interactions between proteins in a protein interaction graph or similarities between webpages in a web similarity graph. These are often symmetric, resulting in an undirected graph; but as we will see, in some applications these similarities or interactions can be asymmetric, which requires us to consider directed graphs as well. The preference graph Γ can come from a variety of considerations. For example, in web applications, one may observe a user's clicking behavior and infer preferences among webpages from these clicks; in an online shopping application, one may record a user's viewing times for different products and use these to infer preferences among products. Often, the preferences in Γ are derived from relevance labels associated with individual objects in V; we give some common examples of this situation below.

Example 1 (Ranking with Binary Labels) Here the objects $v_i \in V$ are associated with binary labels $y_i = +1$ (denoting a 'positive' object, such as a relevant gene) or $y_i = -1$ (denoting a 'negative' object, such as an irrelevant gene); the learner is given examples of a few positive

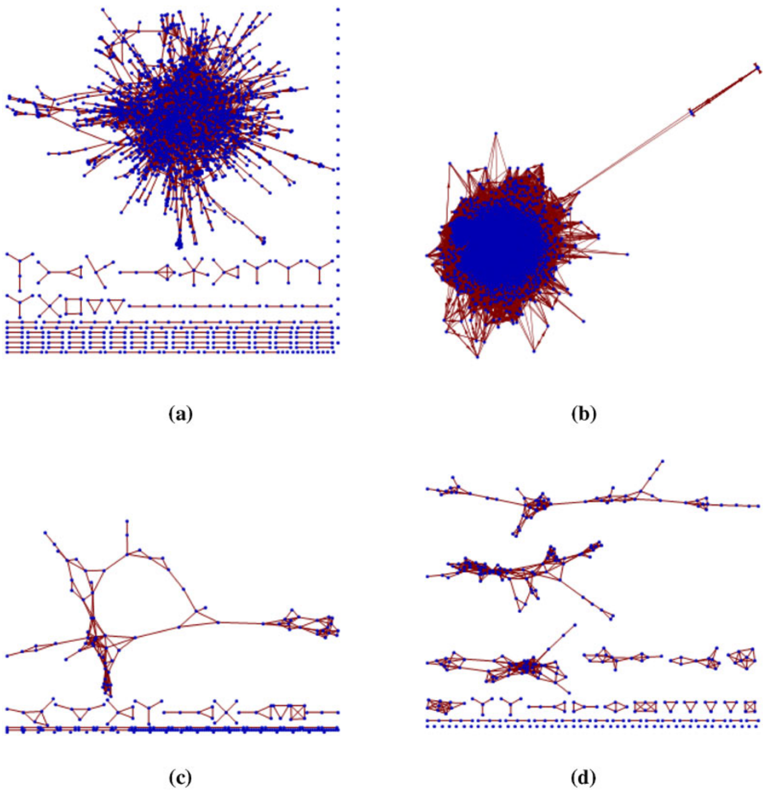

Fig. 1 Diagrams depicting the graphs used in our graph ranking experiments. (**a**) An *S. cerevisiae* (yeast) protein interaction graph. This is an unweighted, undirected graph, with the presence of an edge indicating an interaction between two proteins. The isolated vertices correspond to proteins that are given to interact only with themselves. The ranking task here was to rank kinases above non-kinases. (**b**) A protein similarity graph. This is a weighted, directed graph, with weights representing (approximate) similarities, with each weight taking a value in [0, 1]. The actual graph used in our experiments is a complete graph; the edges shown here correspond to weights greater than $1/2$. The ranking task here was to obtain a hierarchical ranking of proteins in the graph with respect to a specific protein family. (**c**) and (**d**) Chemical similarity graphs over inhibitors of (**c**) cyclooxygenase-2 (COX2) and (**d**) dihydrofolate reductase (DHFR). These are weighted, undirected graphs, again with each weight taking a value in [0, 1]. Again, the actual graphs used in our experiments are complete graphs; the edges shown here correspond to weights greater than $1/2$. The ranking task here was to rank chemical compounds in the graphs according to their activity with respect to the corresponding target. Further details of these graphs and our experiments are given in Sect. 6

objects $\{v_i : i \in S_+\}$ and a few negative objects $\{v_j : j \in S_-\}$ for some $S_+, S_- \subset \{1, \ldots, n\}$, and the goal is to learn a ranking function $f : V \to \mathbb{R}$ that ranks positive objects in V higher than negative objects. In such ranking problems, also known as *bipartite* ranking problems (Freund et al. 2003; Agarwal et al. 2005), the training examples can be viewed as expressing a preference for each object in S_+ over each object in S_-, with a constant mis-ranking penalty for each such positive-negative pair. The preference graph in this case is given by $\Gamma = (V, \Sigma, \tau)$, where $\Sigma = \{(v_i, v_j) : i \in S_+, j \in S_-\}$, and $\tau(v_i, v_j) = 1$ for all $(v_i, v_j) \in \Sigma$.

Example 2 (Ranking with Ordinal Labels) In some cases, the objects $v_i \in V$ are associated with ordinal labels or ratings $y_i \in \{1, \ldots, k\}$ for some $k > 2$ (where a larger value of y_i denotes a higher rating, such as with ratings of books using 1 to k stars); for each $1 \le r \le k$, the learner is given examples of a few objects of rating r, say $\{v_i : i \in S_r\}$ for some $S_r \subset \{1, \ldots, n\}$, and the goal is to learn a ranking function $f : V \rightarrow \mathbb{R}$ that ranks highly-rated objects in V higher than lower-rated objects. In such ranking problems, also referred to as *ordinal* or *k-partite* ranking problems (Herbrich et al. 2000; Rajaram and Agarwal 2005), the training examples can be viewed as expressing a preference for each higher-rated object over each lower-rated one, the penalty for mis-ranking such a pair being proportional to the difference in their ratings. The preference graph in this case is given by $\Gamma = (V, \Sigma, \tau)$, where $\Sigma = \{(v_i, v_j) : i \in S_r, j \in S_s, 1 \le s < r \le k\}$, and $\tau(v_i, v_j) = y_i - y_j$ for all $(v_i, v_j) \in \Sigma$.

Example 3 (Ranking with Real-Valued Labels) Here the objects $v_i \in V$ are associated with real-valued relevance labels $y_i \in \mathbb{R}$ (such as biological activities of chemical compounds with respect to some therapeutic target); the learner is given examples of a few objects together with their relevance labels, say $\{(v_i, y_i) : i \in S\}$ for some $S \subset \{1, \ldots, n\}$, and the goal is to learn a ranking function $f : V \rightarrow \mathbb{R}$ that ranks higher-relevance objects in V higher than lower-relevance objects. In such ranking problems, studied recently by Cortes et al. (2007) and Agarwal and Niyogi (2009), the training examples can be viewed as expressing a preference for each higher-relevance object over each lower-relevance one, the penalty for mis-ranking such a pair being proportional to the difference in their relevance values. The preference graph in this case is given by $\Gamma = (V, \Sigma, \tau)$, where $\Sigma = \{(v_i, v_j) : i, j \in S, y_i > y_j\}$, and $\tau(v_i, v_j) = y_i - y_j$ for all $(v_i, v_j) \in \Sigma$.

As discussed above, given a data graph G and a preference graph Γ, our goal is to learn from $\langle G, \Gamma \rangle$ a 'good' ranking function $f : V \rightarrow \mathbb{R}$. 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 and Niyogi 2004; Zhou et al. 2004). We also note that since $|V| = n$, we can represent any function $f : V \rightarrow \mathbb{R}$ 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 uniformly at random, the expected ranking loss or penalty incurred by a function $f : V \rightarrow \mathbb{R}$ on a pair $(v_i, v_j) \in \Sigma$ is given by

$$\ell(f, v_i, v_j) = \tau(v_i, v_j) \cdot \left[\mathbf{I}_{\{f_i < f_j\}} + \frac{1}{2} \mathbf{I}_{\{f_i = f_j\}} \right], \tag{1}$$

where $\mathbf{I}_{\{\phi\}}$ is an indicator variable that takes the value 1 if the predicate ϕ is true and 0 otherwise. The (empirical) ranking error of f with respect to the preference graph Γ can then be defined as

$$\widehat{\mathrm{er}}_\Gamma(f) = \frac{1}{|\Sigma|} \sum_{(v_i, v_j) \in \Sigma} \ell(f, v_i, v_j). \tag{2}$$

In the following two sections, we develop regularization-based algorithms for learning from $\langle G, \Gamma \rangle$ a ranking function $f_{\langle G, \Gamma \rangle}$ that minimizes an approximate version of the above ranking error. In particular, our algorithms minimize regularized versions of a convex upper bound on the above ranking error with respect to the preference graph Γ; the regularizers we use encourage smoothness of the learned function with respect to the data graph G.

3 Ranking on undirected graphs

In this section we treat the case when the data graph $G=(V,E,w)$ is undirected, so that $(v_i,v_j)\in E \Rightarrow (v_j,v_i)\in E$ and $w(v_i,v_j)=w(v_j,v_i)$ for all $(v_i,v_j)\in E$. The case of directed graphs is treated in Sect. 4.

Our goal is to find a ranking function $f:V\to\mathbb{R}$ that minimizes a suitably regularized version of the ranking error $\widehat{\mathrm{er}}_\Gamma(f)$ with respect to the preference graph Γ; in particular, we would like to find a function that minimizes a suitable combination of the ranking error and a regularization term that penalizes complex functions. Due to its discrete nature, minimizing directly an objective function involving the ranking error $\widehat{\mathrm{er}}_\Gamma(f)$ is NP-hard; we shall use instead a convex upper bound on the ranking error. In particular, consider the following ranking loss, which we refer to as the *hinge* ranking loss due to its similarity to the hinge loss used in classification:

$$\ell_{\mathsf{h}}(f,v_i,v_j)=\big(\tau(v_i,v_j)-(f_i-f_j)\big)_+, \tag{3}$$

where for $a\in\mathbb{R}$, we have

$$a_+=\begin{cases} a, & \text{if } a>0,\\ 0, & \text{otherwise.}\end{cases} \tag{4}$$

The hinge ranking loss $\ell_{\mathsf{h}}(f,v_i,v_j)$ is clearly convex in f and upper bounds the ranking loss $\ell(f,v_i,v_j)$. Our algorithms will minimize a regularized version of the following ℓ_{h}-error:

$$\widehat{\mathrm{er}}^{\mathsf{h}}_\Gamma(f)=\frac{1}{|\Sigma|}\sum_{(v_i,v_j)\in\Sigma}\ell_{\mathsf{h}}(f,v_i,v_j). \tag{5}$$

In particular, we would like to use a regularizer $\mathcal{S}_G(f)$ that encourages smoothness of the learned function with respect to the underlying data graph G; this will encourage generalization to other objects in the graph that are not included in the preference examples Σ.

Several such graph regularizers have been proposed in recent years (Smola and Kondor 2003; Belkin et al. 2004; Zhou and Schölkopf 2004). These regularizers take the form

$$\mathcal{S}_G(f)=\mathbf{f}^T\mathbf{S}\mathbf{f} \tag{6}$$

for an appropriate symmetric, positive semi-definite matrix $\mathbf{S}\in\mathbb{R}^{n\times n}$ derived from G. While we could use any such regularizer in our algorithms, for most of this paper, we shall focus on the Laplacian regularizer, which has the above form with $\mathbf{S}=\mathbf{L}$ or $\mathbf{S}=\widetilde{\mathbf{L}}$, where $\mathbf{L}$ and $\widetilde{\mathbf{L}}$ are the unnormalized and normalized versions of the Laplacian matrix of the graph G, respectively. We do so for two reasons: first, the Laplacian matrix has a nice extension to directed graphs, which we shall discuss in Sect. 4; and second, as discussed by Smola and Kondor (2003), any regularizer that is invariant to permutations of vertices of the graph (in a certain well-defined sense) is necessarily a function of the Laplacian.

The unnormalized Laplacian matrix of the graph G is defined as

$$\mathbf{L}=\mathbf{D}-\mathbf{W}, \tag{7}$$

where $\mathbf{W}$ is the weight matrix given by

$$W_{ij}=\begin{cases} w(v_i,v_j), & \text{if } (v_i,v_j)\in E,\\ 0, & \text{otherwise,}\end{cases} \tag{8}$$

and $\mathbf{D}$ is the diagonal degree matrix $\mathbf{D} = \operatorname{diag}(d_i)$, with diagonal entries d_i given by

$$d_i = \sum_{j=1}^{n} W_{ij}. \tag{9}$$

It is easy to show that for any $f : V \to \mathbb{R}$, the Laplacian regularizer using the above matrix $\mathbf{L}$ can be written as

$$\mathcal{S}_G(f) = \mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{(v_i, v_j) \in E} w(v_i, v_j)(f_i - f_j)^2. \tag{10}$$

This quantity is large for functions f that assign very different scores to objects in the graph that have high similarity or interaction weights, and is small for functions that assign similar scores to such objects. Therefore, selecting a function $f : V \to \mathbb{R}$ with a small value of the above regularizer ensures that f does not vary rapidly across similar objects in the graph; in other words, that the function is smooth with respect to the graph G.

As in Smola and Kondor (2003), Zhou and Schölkopf (2004), we use here the normalized version of the Laplacian matrix, which is given by

$$\widetilde{\mathbf{L}} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2} = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}, \tag{11}$$

where $\mathbf{I}_n$ denotes the $n \times n$ identity matrix and where we have assumed $d_i > 0$ for all i. The normalized Laplacian is often preferred over the unnormalized version due to its attractive spectral properties (Chung 1997; Smola and Kondor 2003); it also forms the basis for the extension to directed graphs, discussed in Sect. 4.

Thus, given the data graph G and preference graph Γ, our algorithm for learning a ranking function $f_{\langle G, \Gamma \rangle} : V \to \mathbb{R}$ minimizes a combination of the empirical ℓ_{h}-error and the normalized Laplacian regularizer as follows:

$$\min_{f : V \to \mathbb{R}} \left[\widehat{\mathrm{er}}^{\mathsf{h}}_{\Gamma}(f) + \lambda \mathbf{f}^T \widetilde{\mathbf{L}} \mathbf{f} \right], \tag{12}$$

for some suitable regularization parameter $\lambda > 0$.

In practice, the above optimization problem can be solved by reduction to a convex quadratic program (QP), 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:

$$\begin{aligned} &\min_{\mathbf{f} \in \mathbb{R}^n} \left[\frac{1}{2} \mathbf{f}^T \widetilde{\mathbf{L}} \mathbf{f} + \frac{C}{|\Sigma|} \sum_{(v_i, v_j) \in \Sigma} \xi_{ij} \right] \\ &\text{subject to} \\ &\quad \xi_{ij} \geq \tau(v_i, v_j) - (f_i - f_j) \quad \forall (v_i, v_j) \in \Sigma \\ &\quad \xi_{ij} \geq 0 \quad \forall (v_i, v_j) \in \Sigma, \end{aligned} \tag{13}$$

where $C = 1/(2\lambda)$. Introducing Lagrange multipliers and taking the Lagrangian dual then results in the following (convex) QP in $|\Sigma|$ variables $\{\alpha_{ij} : (v_i, v_j) \in \Sigma\}$:

$$\min_{\{\alpha_{ij}\}} \left[\frac{1}{2} \sum_{(v_i,v_j)\in\Sigma} \sum_{(v_k,v_l)\in\Sigma} \alpha_{ij}\alpha_{kl} \left(\widetilde{L}^+_{ik} - \widetilde{L}^+_{jk} - \widetilde{L}^+_{il} + \widetilde{L}^+_{jl} \right) - \sum_{(v_i,v_j)\in\Sigma} \alpha_{ij} \tau(v_i, v_j) \right]$$
$$\text{subject to} \qquad (14)$$
$$0 \le \alpha_{ij} \le \frac{C}{|\Sigma|} \quad \forall (v_i, v_j) \in \Sigma,$$

where $\widetilde{L}^+_{ij}$ denotes the (i, j)th element of $\widetilde{\mathbf{L}}^+$, the pseudo-inverse of $\widetilde{\mathbf{L}}$.[1] It can be shown that, on solving the above QP for $\{\alpha_{ij}\}$, the solution $\mathbf{f}_{\langle G,\Gamma\rangle} \in \mathbb{R}^n$ to the original problem is found as

$$\mathbf{f}_{\langle G,\Gamma\rangle} = \widetilde{\mathbf{L}}^+ \big(\mathbf{a}^+ - \mathbf{a}^- \big), \qquad (15)$$

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

$$a^+_i = \sum_{j:(v_i,v_j)\in\Sigma} \alpha_{ij}, \qquad a^-_j = \sum_{i:(v_i,v_j)\in\Sigma} \alpha_{ij}. \qquad (16)$$

Thus, given $\langle G, \Gamma\rangle$, our algorithm for learning a ranking function $\mathbf{f}_{\langle G,\Gamma\rangle}$ consists of solving the QP in (14) for $\{\alpha_{ij}\}$, and then using these to obtain $\mathbf{f}_{\langle G,\Gamma\rangle}$ using (15)–(16).

There are several observations to be made. First, note from (15) that the ranking function $\mathbf{f}_{\langle G,\Gamma\rangle}$ returned by the above algorithm lies in the column-space of $\widetilde{\mathbf{L}}^+$, which is defined as $\{\mathbf{f} \in \mathbb{R}^n : \mathbf{f} = \widetilde{\mathbf{L}}^+\mathbf{u} \text{ for some } \mathbf{u} \in \mathbb{R}\}$. This column-space, together with the inner product defined by

$$\langle \mathbf{f}, \mathbf{g} \rangle = \mathbf{f}^T \widetilde{\mathbf{L}} \mathbf{g}, \qquad (17)$$

forms an RKHS with kernel $\widetilde{\mathbf{L}}^+$. Indeed, that it is a Hilbert space follows from $\widetilde{\mathbf{L}}$ being positive semi-definite. The reproducing property is also easily verified: let $\widetilde{\mathbf{L}}^+_i$ denote the ith column vector of $\widetilde{\mathbf{L}}^+$ and $\mathbf{e}_i \in \mathbb{R}^n$ denote a column vector with 1 in the ith position and 0 elsewhere; then for any $\mathbf{f} = \widetilde{\mathbf{L}}^+\mathbf{u}$, we have:

$$\langle \mathbf{f}, \widetilde{\mathbf{L}}^+_i \rangle = \mathbf{f}^T \widetilde{\mathbf{L}} \widetilde{\mathbf{L}}^+_i = (\mathbf{u}^T \widetilde{\mathbf{L}}^+) \widetilde{\mathbf{L}} (\widetilde{\mathbf{L}}^+ \mathbf{e}_i) = \mathbf{u}^T (\widetilde{\mathbf{L}}^+ \widetilde{\mathbf{L}} \widetilde{\mathbf{L}}^+) \mathbf{e}_i = \mathbf{u}^T \widetilde{\mathbf{L}}^+ \mathbf{e}_i = \mathbf{u}^T \widetilde{\mathbf{L}}^+_i = f_i. \qquad (18)$$

Thus, the algorithm above can be viewed as learning a ranking function in the above RKHS with kernel $\widetilde{\mathbf{L}}^+$; indeed, the regularizer $\mathbf{f}^T \widetilde{\mathbf{L}} \mathbf{f}$ used in the algorithm then corresponds to the (squared) norm of $\mathbf{f}$ in this RKHS.

Second, as noted above, one could in principle use other graph regularizers of the form $\mathbf{f}^T \mathbf{S} \mathbf{f}$, with $\mathbf{S}$ an appropriate positive semi-definite matrix derived from G; the algorithm would take a similar form as above with $\widetilde{\mathbf{L}}^+$ being replaced by $\mathbf{S}^{-1}$, the inverse of $\mathbf{S}$ if $\mathbf{S}$ is invertible and its pseudo-inverse otherwise. This would induce a different RKHS, given by the column space of $\mathbf{S}^{-1}$ together with the inner product $\langle \mathbf{f}, \mathbf{g} \rangle = \mathbf{f}^T \mathbf{S} \mathbf{g}$, and would correspond to learning a ranking function using a different graph kernel, given by $\mathbf{S}^{-1}$.

Third, we note that the optimization problems derived above resemble the RankSVM formulations of Herbrich et al. (2000) and Joachims (2002), although their formulations did

[1] The Laplacian matrix $\widetilde{\mathbf{L}}$ is known to be singular: indeed, it is easy to see that the unnormalized Laplacian $\mathbf{L}$ is singular, since all its rows sum to zero; that $\widetilde{\mathbf{L}}$ is singular then follows from its definition in terms of $\mathbf{L}$.

not involve learning with graphs and did not involve preference weights τ. Indeed, in view of the above observations, the graph ranking algorithm above can be viewed as an extension of RankSVM to graphs using a graph kernel derived from G.

Finally, we note that solving the QP in (14) using a standard QP solver can take $O(|\Sigma|^3)$ time. This can be prohibitive; for example, in ranking with real-valued labels, if we are given labels for m objects in the graph (see Sect. 2, Example 3), all of which have distinct labels, then we have $|\Sigma| = \binom{m}{2}$, giving $O(m^6)$ time. In our experiments, we used a gradient projection algorithm for solving the above QP that is considerably more efficient; this algorithm can be useful for regular RankSVM implementations as well, and is outlined in Appendix A.

4 Ranking on directed graphs

The case when the data graph $G = (V, E, w)$ is directed, so that E and w are asymmetric, can be treated similarly to the undirected case. In particular, the goal is the same: to find a ranking function $f : V \to \mathbb{R}$ that minimizes a suitably regularized convex upper bound on the empirical ranking error $\widehat{\mathrm{er}}_\Gamma(f)$ with respect to the preference graph Γ.

The convex upper bound on $\widehat{\mathrm{er}}_\Gamma(f)$ can be chosen to be the same as before, namely, the ℓ_{h}-error, which was denoted by $\widehat{\mathrm{er}}^{\mathsf{h}}_\Gamma(f)$. The goal is then again to solve an optimization problem of the form

$$\min_{f:V\to\mathbb{R}} \left[\widehat{\mathrm{er}}^{\mathsf{h}}_\Gamma(f) + \lambda \mathcal{S}_G(f)\right] \tag{19}$$

for some suitable regularizer $\mathcal{S}_G(f)$ (and regularization parameter $\lambda > 0$). This is where the technical difference lies: there has been limited work on regularization for directed graphs, and indeed, in the form described so far, the Laplacian regularizer used in the previous section applies only to undirected graphs.

Recently, however, an analogue of the Laplacian was 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. In particular, given the (weighted) directed graph G, let d_i^+ denote the out-degree of v_i:

$$d_i^+ = \sum_{j:(v_i,v_j)\in E} w(v_i, v_j). \tag{20}$$

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:

$$P_{ij} = \begin{cases} \frac{w(v_i,v_j)}{d_i^+}, & \text{if } (v_i, v_j) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

In this case, the above random walk has a unique stationary distribution $\pi = (\pi_1, \ldots, \pi_n)^T$ with $\pi_i > 0$ for all i. The Laplacian of G is then defined as[2]

$$\widetilde{\mathbf{L}} = \mathbf{I}_n - \frac{\Pi^{1/2}\mathbf{P}\Pi^{-1/2} + \Pi^{-1/2}\mathbf{P}^T\Pi^{1/2}}{2}, \tag{22}$$

[2]We continue to use the notation $\widetilde{\mathbf{L}}$ in this section as the Laplacian defined here for directed graphs extends the normalized Laplacian for undirected graphs.

where Π is the diagonal matrix $\Pi = \operatorname{diag}(\pi_i)$. In the case when G is not strongly connected or is periodic, one can use what is termed a *teleporting* random walk, which effectively allows one to jump to a random vertex (chosen uniformly from the vertices distinct from the current vertex) with some small probability η (Zhou et al. 2005); the probability transition matrix $\mathbf{P}^{(\eta)}$ for such a walk is given by

$$P_{ij}^{(\eta)} = \begin{cases} (1-\eta)P_{ij} + \eta(\frac{1}{n-1}), & \text{if } i \neq j, \\ (1-\eta)P_{ij}, & \text{if } i = j. \end{cases} \tag{23}$$

Such a teleporting random walk always converges to a unique and positive stationary distribution $\pi^{(\eta)}$, and therefore the Laplacian $\widetilde{\mathbf{L}}$ for a general directed graph (and appropriate η) can be defined as in (22), using $\mathbf{P}^{(\eta)}$ and the corresponding $\pi^{(\eta)}$ in place of $\mathbf{P}$ and π.

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

5 Stability and generalization properties

The RKHS view of our graph ranking algorithms allows us to show attractive stability and generalization properties of these algorithms using results of Agarwal and Niyogi (2009). In particular, we will consider the setting of ranking with real-valued labels (see Sect. 2, Example 3); similar results can be shown for other settings such as ranking with binary or ordinal labels (Sect. 2, Examples 1 and 2).

Let $G = (V, E, w)$ be a data graph. In the setting of ranking with real-valued labels, the objects $v_i \in V$ are associated with real-valued relevance labels $y_i \in \mathbb{R}$; we shall assume that the labels are bounded and take without loss of generality $y_i \in [0, M]$ for some $M > 0$. The learner is given a training sample consisting of examples of objects in V together with their relevance labels, which we shall denote here as $R = \{(v_{i_1}, y_{i_1}), \ldots, (v_{i_m}, y_{i_m})\} \subset V \times [0, M]$; the goal is to learn from R a ranking function $f : V \to \mathbb{R}$ that ranks accurately the remaining objects in V. We shall assume that each $(v_{i_r}, y_{i_r}) \in R$ is drawn randomly and independently according to some underlying joint distribution $\mathcal{D}$ over $V \times [0, M]$. We shall further assume that a set $Y \subseteq [0, M]$ of positive measure is included in the support of the marginal of $\mathcal{D}$ over the label space $[0, M]$; this ensures that with probability 1, all labels in R are distinct, and therefore the resulting preference graph $\Gamma = (V, \Sigma, \tau)$ contains $\binom{m}{2}$ edges.

Overloading notation, let the loss or penalty incurred by a ranking function $f : V \to \mathbb{R}$ on any pair of labeled objects $(v, y), (v', y') \in V \times \mathbb{R}$ be

$$\ell(f, (v, y), (v', y')) = |y - y'| \cdot \left[\mathbf{I}_{\{(y-y')(f(v)-f(v'))<0\}} + \frac{1}{2}\mathbf{I}_{\{f(v)=f(v')\}}\right]. \tag{24}$$

Then for any ranking function $f : V \to \mathbb{R}$, we can write the empirical error of f with respect to (the preference graph resulting from) a training sample $R \in (V \times [0, M])^m$ as above as

$$\widehat{\mathrm{er}}_R(f) = \frac{1}{\binom{m}{2}} \sum_{r=1}^{m-1} \sum_{s=r+1}^{m} \ell\big(f, (v_{i_r}, y_{i_r}), (v_{i_s}, y_{i_s})\big). \tag{25}$$

Also, define the expected error of f with respect to $\mathcal{D}$ as

$$\mathrm{er}_{\mathcal{D}}(f) = \mathbf{E}_{((v,y),(v',y'))\sim\mathcal{D}\times\mathcal{D}}\big[\ell\left(f,(v,y),(v',y')\right)\big]. \tag{26}$$

Moreover, for any $R \in (V \times [0,M])^m$ as above, we will denote by $f_R : V \to \mathbb{R}$ the ranking function learned by our graph ranking algorithm (the algorithm of either Sect. 3 or Sect. 4, depending on whether G is undirected or directed) from the preference graph resulting from R. Then using results of Agarwal and Niyogi (2009), we can show that the expected error of the learned function, $\mathrm{er}_{\mathcal{D}}(f_R)$, can with high probability (over the draw of R) be upper bounded in terms of a quantity related to the empirical error, $\widehat{\mathrm{er}}_R(f_R)$.

We first have the following stability result for our algorithms, which shows that a small change in the training sample has limited effect on the learned ranking function:

Theorem 1 (Stability of Graph Ranking) *Let $R \in (V \times [0,M])^m$ and $R' \in (V \times [0,M])^m$ differ in only one example. Then the functions $f_R : V \to \mathbb{R}$ and $f_{R'} : V \to \mathbb{R}$ satisfy the following for all $v \in V$:*

$$|f_R(v) - f_{R'}(v)| \le \frac{8 \max_{1\le i\le n} \widetilde{L}^+_{ii}}{\lambda m}.$$

The above result follows from Agarwal and Niyogi (2009, Theorem 11), which establishes stability of all kernel-based ranking algorithms that minimize a suitable ranking error with regularization in an RKHS (recall that $\widetilde{\mathbf{L}}^+$ can be viewed as the kernel matrix in our case), and from the fact that the hinge ranking loss optimized by our algorithms, which in the above setting can be written as

$$\ell_{\mathsf{h}}(f,(v,y),(v',y')) = \big(|y-y'| - \mathrm{sign}(y-y')\cdot(f(v)-f(v'))\big)_+, \tag{27}$$

is 1-admissible, in the sense that for any $f_1, f_2 : V \to \mathbb{R}$ and any $(v,y),(v',y') \in V \times \mathbb{R}$,

$$\big|\ell_{\mathsf{h}}(f_1,(v,y),(v',y')) - \ell_{\mathsf{h}}(f_2,(v,y),(v',y'))\big| \le |f_1(v)-f_2(v)| + |f_1(v')-f_2(v')|. \tag{28}$$

See Agarwal and Niyogi (2009) for details.

Before giving our generalization result, we need to define one more ranking loss, which we will refer to as the γ ranking loss (where $\gamma > 0$):

$$\ell_\gamma(f,(v,y),(v',y')) = \begin{cases} |y-y'|, \\ \quad \text{if } \mathrm{sign}(y-y')\cdot(f(v)-f(v')) < 0, \\ |y-y'| - \frac{1}{\gamma}\mathrm{sign}(y-y')\cdot(f(v)-f(v')), \\ \quad \text{if } 0 \le \mathrm{sign}(y-y')\cdot(f(v)-f(v')) \le \gamma|y-y'|, \\ 0, \quad \text{otherwise.} \end{cases} \tag{29}$$

For any $\gamma > 0$, the corresponding empirical ℓ_γ-error of a ranking function $f : V \to \mathbb{R}$ with respect to a training sample $R \in (V \times [0,M])^m$ as above can be written as

$$\widehat{\mathrm{er}}^\gamma_R(f) = \frac{1}{\binom{m}{2}} \sum_{r=1}^{m-1} \sum_{s=r+1}^{m} \ell_\gamma\big(f,(v_{i_r},y_{i_r}),(v_{i_s},y_{i_s})\big). \tag{30}$$

Then we have the following generalization result for our algorithms:

Theorem 2 (Generalization Bound for Graph Ranking) *Let* $\gamma > 0$. *Then for any* $0 < \delta < 1$, *with probability at least* $1 - \delta$ *over the draw of* $R \in (V \times [0, M])^m$ (*according to* $\mathcal{D}^m$),

$$\mathrm{er}_{\mathcal{D}}(f_R) < \widehat{\mathrm{er}}_R^{\gamma}(f_R) + \frac{32 \max_{1\le i\le n} \widetilde{L}_{ii}^+}{\gamma\lambda m} + \left(\frac{16 \max_{1\le i\le n} \widetilde{L}_{ii}^+}{\gamma\lambda} + M\right)\sqrt{\frac{2\ln(1/\delta)}{m}}.$$

The above result follows from Agarwal and Niyogi (2009, Theorem 8) and from the stability result in Theorem 1 above.

Theorems 1 and 2 above apply to our algorithms for both directed and undirected graphs, with $\widetilde{\mathbf{L}}^+$ referring to the pseudo-inverse of the appropriate Laplacian matrix in each case. For the case of connected, undirected graphs, the diagonal elements of this pseudo-inverse, $\widetilde{L}_{ii}^+$, can be further bounded in terms of the diameter of the graph:

Theorem 3 (Diameter Bound for Connected, Undirected Graphs) *Let* $G = (V, E, w)$ *be a connected, weighted, undirected graph, and let* $\widetilde{\mathbf{L}}$ *be the* (*normalized*) *Laplacian matrix of* G. *Let* $d = \max_{1\le i\le n} d_i$ *and* $w_{\min} = \min_{(i,j)\in E} w(i,j)$, *and let* ρ *be the unweighted diameter of* G, *i.e., the length* (*number of edges*) *of the longest path between any two vertices* i *and* j *in* V. *Then for all* $1 \le i \le n$,

$$\widetilde{L}_{ii}^+ \le \frac{\rho d}{w_{\min}}.$$

The proof of the above result is based on the proof of a similar result of Herbster et al. (2005), which was given for the unnormalized Laplacian of an unweighted graph; for completeness, details are included in Appendix B. In particular, this leads to the following:

Corollary 1 (Generalization Bound for Ranking on Connected, Undirected Graphs) *Let* G, ρ, d, *and* $w_{\min}$ *be as in Theorem* 3, *and let* $\gamma > 0$. *Then for any* $0 < \delta < 1$, *with probability at least* $1 - \delta$ *over the draw of* $R \in (V \times [0, M])^m$ (*according to* $\mathcal{D}^m$),

$$\mathrm{er}_{\mathcal{D}}(f_R) < \widehat{\mathrm{er}}_R^{\gamma}(f_R) + \frac{32\rho d}{\gamma\lambda m w_{\min}} + \left(\frac{16\rho d}{\gamma\lambda w_{\min}} + M\right)\sqrt{\frac{2\ln(1/\delta)}{m}}.$$

6 Experiments

We describe below experiments on three different graph ranking tasks in computational biology and in cheminformatics. These tasks were chosen for their differing characteristics, both in terms of the type of ranking problem and in terms of the structure of the graph.

The first task involves an *S. cerevisiae* (yeast) protein interaction network, in which some of the proteins are kinases; the learner is given examples of both kinases and non-kinases, and the goal is to identify kinases in the remaining proteins, i.e., to rank the remaining proteins such that kinases are ranked higher than non-kinases. This corresponds to ranking with binary labels (see Sect. 2, Example 1) on an undirected graph. We will compare our graph ranking algorithm on this task with a corresponding graph classification algorithm.

The second task involves a directed similarity graph; this is also a graph over proteins, drawn from a different database, with the asymmetry arising from an approximation in the sequence alignment tool used to find the similarity of one protein to another. The proteins in this graph are associated with a hierarchical organization; we consider a ranking problem in which the learner is given examples of proteins at different levels of the hierarchy with respect to a specific protein family, and the goal is to rank other proteins in the graph according to this hierarchy. This corresponds to ranking with ordinal labels (see Sect. 2, Example 2) on a directed graph. We will compare our graph ranking algorithm on this task with a corresponding algorithm for ordinal regression on graphs.

The third task involves chemical similarity graphs that describe similarities between chemical compounds. We consider two such graphs; in each case, the learner is given (real-valued) biological activities for some of the compounds, which measure the extent to which the compounds inhibit the activity of a particular drug target, and the goal is to rank the remaining compounds such that those that are most active against the target are ranked higher. This corresponds to ranking with real-valued labels (see Sect. 2, Example 3) on an undirected graph; in this case the weight matrix itself is positive semi-definite, and so we do not need to use the Laplacian. We will compare our graph ranking algorithm on this task with a corresponding graph regression algorithm.

The following sections describe in more detail the experimental setup in each case, together with our results.

6.1 Identifying kinases in a protein interaction network

Our first graph ranking task involves an *S. cerevisiae* (yeast) protein interaction network downloaded from the Database of Interacting Proteins (DIP).[3] We considered only those proteins for which UniProt Knowledgebase (UniProtKB) identifiers were provided;[4] this resulted in an interaction graph $G = (V, E, w)$ over 2418 proteins, containing 4538 interactions (edges) in all. This is an unweighted, undirected graph, in which the presence of an edge between two proteins indicates an interaction between them (see Fig. 1(a)); for pairs of proteins v_i, v_j that share an interaction, we took $w(v_i, v_j) = w(v_i, v_j) = 1$.

The ranking task we considered was to identify kinases in this network. A kinase is a type of protein (specifically, enzyme) that transfers phosphate groups from a donor molecule to another recipient molecule. Given their central role in many diseases including cancer, kinases form prominent targets for drug development, thus making their identification an important task in biology. In order to apply machine learning methods for this task, we obtained a list of *S. cerevisiae* proteins that are known to belong to the kinase family by searching the UniProtKB database. This gave an initial list of 183 kinases; of these, 101 were present in the protein interaction network constructed above. Thus, of the 2418 proteins in the above graph, we had 101 'positive' proteins corresponding to the kinases, and 2317 'negative' proteins corresponding to the non-kinases. This gave rise to a ranking task with binary labels (see Sect. 2, Example 1); in particular, given a small number of examples of both kinases and non-kinases, our goal was to rank the remaining proteins such that kinases are ranked higher than non-kinases.

We considered a number of different splits of the 2418 proteins in the graph into training and test sets. In particular, we considered training sets of sizes $m \in \{120, 240, 360, 480, 600\}$; for each size m, we used 10 random splits of the proteins V into a training set $\{v_i : i \in S\}$ of size $|S| = m$ and a test set $V \setminus \{v_i : i \in S\}$ of size $(2418 - m)$, subject to a constant proportion of positives and negatives in each set. Specifically, for $m = 120$, we included 5 positive examples and 115 negative examples; for $m = 240$, we included 10 positive examples and

[3]The DIP database catalogs experimentally determined interactions between proteins, and is available at http://dip.doe-mbi.ucla.edu. We used the core *S. cerevisiae* data set dated October 14, 2008.

[4]The UniProtKB database is a central repository for functional information on proteins, and is available at http://www.uniprot.org/help/uniprotkb.

230 negative examples; and so on. As described in Example 1, this results in a preference graph $\Gamma = (V, \Sigma, \tau)$ with $|\Sigma| = |S_+||S_-|$ preferences, where S_+, S_- denote the positive and negative training examples in S, respectively. For each such training set S (equivalently, preference graph Γ), we used the graph ranking algorithm described in Sect. 3 to learn a ranking function $f : V \rightarrow \mathbb{R}$. For comparison, we also used the support vector machine (SVM) algorithm using the same Laplacian graph kernel to learn a binary classifier $h : V \rightarrow \{-1, +1\}$: this has the form $h(v_i) = \text{sign}(f(v_i) + b)$ for some $f : V \rightarrow \mathbb{R}$ and $b \in \mathbb{R}$; we used the underlying real-valued function f to rank the remaining proteins in the graph. The trade-off parameter C for both algorithms was selected from the range $\{0.1, 1, 10, 100, 1000\}$ using 5-fold cross-validation; in each case, the parameter value that gave the lowest average ranking error across the five folds was selected for training. The number of iterations $t_{\max}$ in the gradient projection algorithm for ranking (see Appendix A) was fixed to 1000; the learning rate η was selected from the range $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ using 5-fold cross-validation as above.

The results are shown in Fig. 2. Figure 2(a) shows the results in terms of the ranking error, which is the quantity that is approximately minimized by the graph ranking algorithm; each value shown is the average across the 10 random splits for the corresponding training size m. We note that the ranking error in this case is simply the fraction of mis-ranked positive-negative pairs. We also note that the area under the ROC curve (AUC), a popular ranking performance measure in the binary setting, is simply one minus this ranking error (Agarwal et al. 2005). As can be seen, the graph ranking algorithm yields better ranking performance than the corresponding classification algorithm, which uses the same graph kernel but optimizes classification accuracy.

Often in ranking, the quality of the results returned at the top of a ranked list is especially important. One quantity used to capture this in the binary setting is the *average precision*. In particular, let T_+, T_- denote the (indices of) the positive and negative test objects in V, respectively; then the *precision at k* of a ranking function $f : V \rightarrow \mathbb{R}$ is defined as the pro-

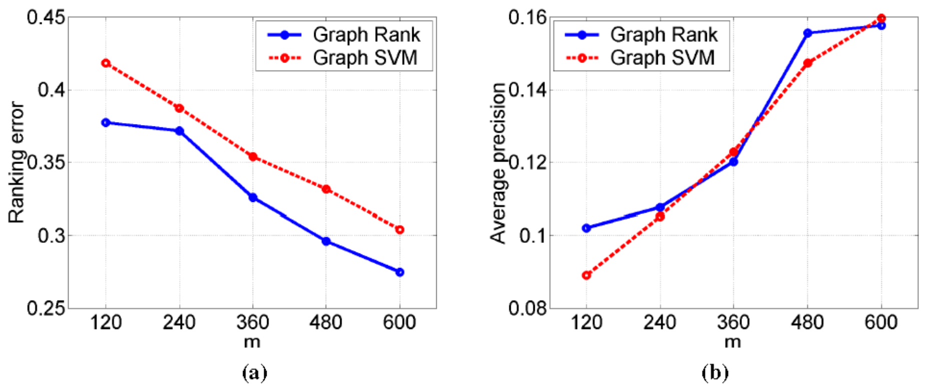

Fig. 2 Results on the task of identifying kinases in an *S. cerevisiae* protein interaction network. This is a bipartite ranking task on an undirected graph, the goal being to rank kinases higher than non-kinases. The plots compare the performance of the graph ranking algorithm given in Sect. 3 with the binary classification SVM using the same Laplacian graph kernel; the two algorithms therefore learn a function from the same function class. Results are shown for increasing values of the training size m, in terms of **(a)** ranking error (one minus AUC); and **(b)** average precision. Each value shown in the plots is an average over 10 random trials. For the ranking error, lower values correspond to better ranking performance; for the average precision, higher values correspond to better ranking performance. See text for details

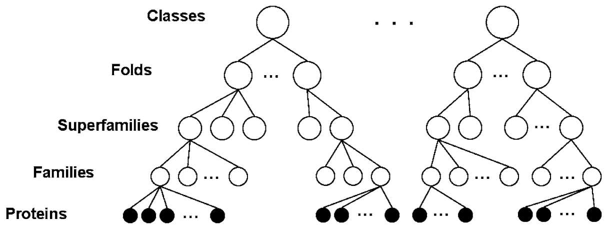

Fig. 3 Proteins in the Structural Classification of Proteins (SCOP) database are organized hierarchically into classes, folds, superfamilies, and families. We considered a hierarchical ranking task on a protein similarity graph derived from this database. See text for details

portion of positives in the top k objects returned by f:

$$\text{prec}_k(f) = \frac{1}{k} \sum_{i \in T_+} \mathbf{I}_{\{\pi(i) \leq k\}}, \tag{31}$$

where $\pi(i)$ is the position of v_i in the ranking of test objects returned by f. The average precision of f is simply the average of these precision values taken at the positions of positive objects in the returned ranking:

$$\text{AP}(f) = \frac{1}{|T_+|} \sum_{j \in T_+} \text{prec}_{\pi(j)}(f). \tag{32}$$

Clearly, higher values of the average precision correspond to better ranking performance. Figure 2(b) shows the results in terms of the average precision. In this case, the difference in performance of the two algorithms is smaller, although again, the graph ranking algorithm gives slightly better overall performance than the classification algorithm. Note however that overall, neither algorithm has good performance in terms of the average precision (the ideal ranking would have an average precision of 1). We discuss this further in Sect. 7.

6.2 Hierarchical ranking of proteins in an asymmetric protein similarity graph

The second graph ranking task we consider involves a protein similarity graph derived from a different protein database, specifically, the Structural Classification of Proteins (SCOP) database.[5] The SCOP database consists of a hierarchical organization of proteins, based on their 3D structure, into classes, folds, superfamilies, and families: closely related proteins are grouped to form a protein family; related protein families form a superfamily; related superfamilies form a fold; and finally, related folds form a protein structure class (Fig. 3).

The protein similarity graph we used was drawn from a graph used by Weston et al. (2004),[6] who considered a different kind of protein ranking task[7] on the same database. In our experiments, we considered proteins from two classes, namely the 'all α' and 'all β' classes, which resulted in a similarity graph $G = (V, E, w)$ over 3314 proteins in all. This is a complete, weighted, directed graph (see Fig. 1(b)), in which the similarity $w(v_i, v_j)$ of

[5]The SCOP database describes structural relationships between proteins, and is available at http://scop.mrc-lmb.cam.ac.uk/scop/.

[6]The graph used by Weston et al. (2004) is based on SCOP version 1.59 and is available at http://www.kyb.tuebingen.mpg.de/bs/people/weston/rankprot/supplement.html.

[7]The protein ranking task considered by Weston et al. (2004) involves ranking proteins in the graph by relevance to a single 'query' protein. The approach used is based on that of Zhou et al. (2004); see also the discussion in Sect. 1.

each protein v_i to each other protein v_j is based on an 'E-value' $\mathbf{E}_{ij}$ computed using the popular PSI-BLAST sequence search and alignment tool (Altschul et al. 1997). In particular, given a protein v_i, PSI-BLAST attempts to find for each other protein v_j in the database the highest-scoring alignment between pairs of segments in the amino acid sequences for v_i and v_j (under some appropriate alignment scoring scheme), and then returns an E-value $\mathbf{E}_{ij}$ that denotes the expected number of segment pairs that would result in a similarly high-scoring alignment if the two sequences had been drawn randomly (under an appropriate random sequence model). Thus, the E-value can be viewed as a measure of statistical significance of the (local) similarity between the two sequences: if the sequences are similar (contain segments with a high-scoring alignment), the E-value is low; and vice-versa. Following Weston et al. (2004), we convert these to similarity scores by taking $w(v_i, v_j) = e^{-\mathbf{E}_{ij}/100}$. Under an exact alignment search algorithm, the E-values would satisfy $\mathbf{E}_{ij} = \mathbf{E}_{ji}$; however, due to efficiency considerations, PSI-BLAST employs an approximation that results in asymmetric E-values, thus giving asymmetric similarity weights $w(v_i, v_j)$.

The ranking task we considered was to rank proteins in the above graph according to the structural hierarchy of SCOP with respect to a specific protein family. In particular, we took the largest protein family, an 'all β' family of 'V set domains (antibody variable like)', denoted as 'b.1.1.1' in the SCOP database and containing 403 proteins, as our target; given a small number of examples of proteins at different levels of the hierarchy with respect to this target family, our goal was to rank the remaining proteins such that those in the target family would be ranked highest, those in other families but belonging to the same superfamily would be ranked next, and so on. This can be viewed as a ranking task with ordinal labels (see Sect. 2, Example 2), with $k = 5$ possible ratings: a protein v_i in the target family has rating $y_i = 5$; a protein in a different family but in the same superfamily has rating $y_i = 4$; a protein in a different superfamily but in the same fold has rating $y_i = 3$; a protein in a different fold but in the same class has rating $y_i = 2$; and a protein in a different class has rating $y_i = 1$. Given a small number of examples of proteins of each rating, the goal is to rank the remaining proteins in the graph such that higher-rated proteins are ranked higher.

We considered a number of different splits of the 3314 proteins in the graph into training and test sets. In particular, we considered training sets of sizes $m \in \{100, 200, 300, 400, 500\}$; for each size m, we used 10 random splits of the proteins V into a training set $\{v_i : i \in S\}$ of size $|S| = m$ and a test set $V \setminus \{v_i : i \in S\}$ of size $(3314 - m)$, subject to each having the same proportion of proteins from the five rating classes as in the complete set V. As described in Example 2, this results in a preference graph $\Gamma = (V, \Sigma, \tau)$ with $|\Sigma| = \sum_{1 \le s < r \le 5} |S_r||S_s|$ preferences, where for each $1 \le r \le 5$, S_r denotes the training examples in S with rating r. For each such training set S (equivalently, preference graph Γ), we used the graph ranking algorithm described in Sect. 4 to learn a ranking function $f : V \to \mathbb{R}$; in constructing the graph Laplacian, we used a teleporting random walk with $\eta = 0.01$ (see Sect. 4). For comparison, we also used the state-of-the-art support vector ordinal regression (SVOR) algorithm of Chu and Keerthi (2007) using the same Laplacian graph kernel to learn an ordinal regression prediction function $g : V \to \{1, \dots, 5\}$: this consists of a real-valued function $f : V \to \mathbb{R}$ together with a set of thresholds $b_1, \dots, b_4 \in \mathbb{R}$ which are used to predict a rating class via $g(v_i) = \min\{1 \le i \le 5 : f(v_i) \le b_i\}$, where $b_5 = \infty$; we used the underlying real-valued function f to rank the remaining proteins in the graph. We used gradient projection algorithms for both ranking (see Appendix A) and ordinal regression; for both algorithms, the number of iterations $t_{\max}$ was fixed to 1000, while the trade-off parameter C and the learning rate η were selected using 5-fold cross-validation from the ranges $\{0.1, 1, 10, 100, 1000\}$ and $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, respectively (in each case, the

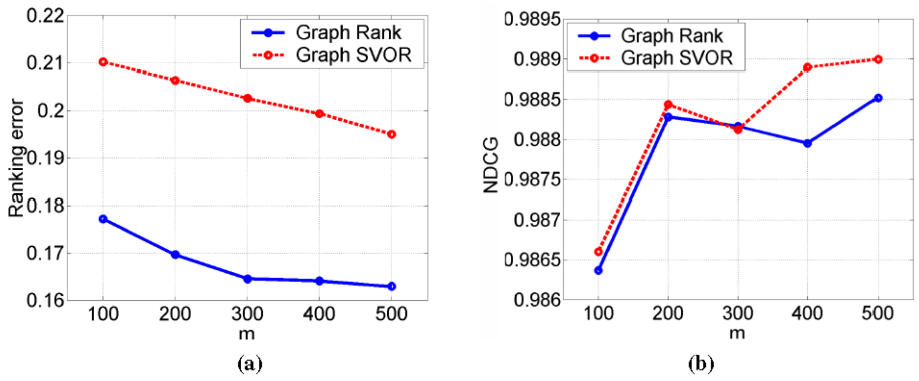

Fig. 4 Results on the task of hierarchical ranking of proteins in an asymmetric protein similarity graph derived from the Structural Classification of Proteins (SCOP) database. This is an ordinal ranking task on a directed graph, the goal being to rank proteins according to their position in the SCOP hierarchy with respect to the 'b.1.1.1' protein family. The plots compare the performance of the graph ranking algorithm given in Sect. 4 with a support vector ordinal regression (SVOR) algorithm using the same Laplacian graph kernel; the two algorithms therefore learn a function from the same function class. Results are shown for increasing values of the training size m, in terms of **(a)** ranking error; and **(b)** NDCG. Each value shown in the plots is an average over 10 random trials. For the ranking error, lower values correspond to better ranking performance; for the NDCG, higher values correspond to better ranking performance. See text for details

parameter values that gave the lowest average ranking error across the five folds were selected for training).

The results are shown in Fig. 4. Figure 4(a) shows the results in terms of the ranking error, which is the quantity that is approximately minimized by the graph ranking algorithm; each value shown is the average across the 10 random splits for the corresponding training size m. As can be seen, the graph ranking algorithm yields better ranking performance than the corresponding ordinal regression algorithm, which uses the same graph kernel but optimizes ordinal regression prediction accuracy.

We also evaluated both algorithms in terms of the *normalized discounted cumulative gain* (NDCG), a quantity often used in information retrieval to measure ranking quality at the top of a ranked list when there are more than two relevance levels (Järvelin and Kekäläinen 2002). Specifically, in our setting, let T denote the (indices of) the test objects in V; then the NDCG of a ranking function $f : V \to \mathbb{R}$ with respect to T is defined as

$$\mathrm{NDCG}(f) = \frac{1}{Z_T} \sum_{i \in T} \frac{2^{y_i} - 1}{\log_2(\pi(i) + 1)}, \tag{33}$$

where $\pi(i)$ is the position of v_i in the ranking of test objects returned by f, y_i is the relevance (rating) of v_i, and Z_T is a normalizing constant which ensures that the maximum NDCG with respect to T over all functions f is 1. As can be seen, the NDCG accumulates a gain value $2^{y_i} - 1$ for each object $i \in T$, discounted using a logarithmic discounting factor $\log_2(\pi(i) + 1)$ that depends on the position of the object in the ranked list. Clearly, higher values of the NDCG correspond to better ranking performance. Figure 4(b) shows the results in terms of the NDCG. In this case, the performance of the two algorithms is broadly similar; in particular, note that both algorithms give NDCG values close to 0.99.

6.3 Ranking chemical compounds by activity in chemical similarity graphs

Our third graph ranking task is from the cheminformatics domain and involves two chemical similarity graphs. These were derived from cheminformatics data sets used by Sutherland et al. (2004). Specifically, the first graph was derived from a data set containing inhibitors of dihydrofolate reductase (DHFR). DHFR is an enzyme that is involved in the production of folate in the human body. Folate is important for healthy cell and tissue growth; however, it is also needed by rapidly dividing cancer cells. Therefore drugs that inhibit DHFR, thereby interfering with folate metabolism, can be useful in treating cancer; indeed, one of the earliest anti-cancer drugs, methotrexate, works by inhibiting DHFR. The DHFR data set used by Sutherland et al. (2004) consists of 361 compounds that inhibit DHFR to varying levels, together with their biological activities with respect to DHFR. Each compound in the data set is represented as a vector of 70 chemical descriptors; we scaled each descriptor value to lie between a minimum of 0 and a maximum of 1, and constructed a chemical similarity graph $G = (V, E, w)$ over the 361 compounds by taking the similarity between two compounds v_i, v_j to be $w(v_i, v_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}$, where $\mathbf{x}_i, \mathbf{x}_j \in [0, 1]^{70}$ denote the corresponding scaled descriptor vectors. This is a complete, weighted, undirected graph (see Fig. 1(c)).

The second graph was derived from a data set containing inhibitors of cyclooxygenase-2 (COX2). COX2 is an enzyme that is implicated in the generation of prostaglandins, which are responsible for inflammation and pain; drugs that inhibit COX2 can therefore be useful in providing relief from these symptoms. Indeed, some of the classical non-steroidal anti-inflammatory drugs (NSAIDs), such as aspirin and ibuprofen, inhibit COX2 along with COX1, another COX isoenzyme; some newer NSAIDs, such as celecoxib and etoricoxib, are selective inhibitors of COX2. The COX2 data set used by Sutherland et al. (2004) consists of 282 compounds that inhibit COX2 to varying levels, together with their biological activities with respect to COX2. Each compound in this data set is represented as a vector of 74 chemical descriptors; again, we scaled each descriptor value to lie between 0 and 1, and constructed a chemical similarity graph $G = (V, E, w)$ over the 282 compounds as was done in the case of the DHFR data set above. Again, this is a complete, weighted, undirected graph (see Fig. 1(d)).

For each of the above graphs, the ranking task we considered was to rank compounds according to their biological activities with respect to the target of interest (DHFR or COX2). In particular, the data sets used by Sutherland et al. (2004) include the biological activities of the corresponding compounds represented as pIC_{50} values. The pIC_{50} value of a compound measures its effectiveness in inhibiting a particular biological or biochemical function, and is defined as

$$\mathrm{pIC}_{50} = -\log_{10} \mathrm{IC}_{50}, \tag{34}$$

where IC_{50} denotes the 50% inhibitory concentration of the compound, i.e., the concentration of the compound (generally measured in mol/L) required to inhibit a given biological function (in this case, the function of DHFR or COX2) by half. Thus the higher the pIC_{50} value of a compound, the greater its inhibition effect (and therefore biological activity). The pIC_{50} values in the DHFR data set range from 3.3 to 9.8; those in the COX2 data set range from 4.0 to 9.0. In each case, we had a ranking task with real-valued labels (see Sect. 2, Example 3): given the pIC_{50} values for a small number of compounds, our goal was to rank the remaining compounds in the graph such that compounds with greater activity against the target would be ranked higher.

In each case, we considered a number of different splits of the compounds in the graph into training and test sets. In particular, for both graphs, we considered training sets of sizes

$m \in \{20, 40, 60, 80, 100\}$; for each size m, we used 10 random splits of the compounds V into a training set $\{v_i : i \in S\}$ of size $|S| = m$ and a test set $V \setminus \{v_i : i \in S\}$ of size $(n - m)$, where $n = 361$ for the graph over DHFR inhibitors and $n = 282$ for the graph over COX2 inhibitors. As described in Example 3, this results in a preference graph $\Gamma = (V, \Sigma, \tau)$ with $|\Sigma| = |\{(i, j) : i, j \in S, y_i > y_j\}|$ preferences, where $y_i \in \mathbb{R}$ denotes the biological activity of compound v_i. For each such training set S (equivalently, preference graph Γ), we used the graph ranking algorithm described in Sect. 3 to learn a ranking function $f : V \to \mathbb{R}$; in this case, by construction, the weight matrix $\mathbf{W}$ is already symmetric and positive semi-definite, so we used this as the kernel matrix (which amounts to using $\mathbf{f}^T \mathbf{W}^{-1} \mathbf{f}$ as the regularizer). For comparison, we also used the support vector regression (SVR) algorithm using the same kernel to learn a prediction function $f : V \to \mathbb{R}$, and used this to rank the remaining compounds in the graph. The trade-off parameter C for both algorithms was selected from the range $\{0.1, 1, 10, 100, 1000\}$ using 5-fold cross-validation; the insensitivity parameter ϵ in the SVR algorithm was selected similarly from the range $\{0.01, 0.05, 0.1, 0.5, 1\}$. In each case, the parameter values that gave the lowest average ranking error across the five folds were selected for training. The number of iterations $t_{\max}$ in the gradient projection algorithm for ranking (see Appendix A) was fixed to 1000; the learning rate η was selected from the range $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ using 5- fold cross-validation as above.

The results are shown in Fig. 5 (DHFR) and Fig. 6 (COX2). Figures 5(a) and 6(a) show the results in terms of the ranking error, which is the quantity that is approximately minimized by the graph ranking algorithm; each value shown is the average across the 10 random splits for the corresponding training size m. As can be seen, in both cases, the graph ranking algorithm yields better ranking performance than the corresponding regression algorithm, which uses the same graph kernel but optimizes prediction accuracy.

We also evaluated both algorithms in terms of the NDCG (see Sect. 6.2); these results are shown in Figs. 5(b) and 6(b). Again, the graph ranking algorithm gives slightly better performance than the regression algorithm. However the performance in terms of the NDCG does not always show the same monotonic behavior that is observed with the ranking error;

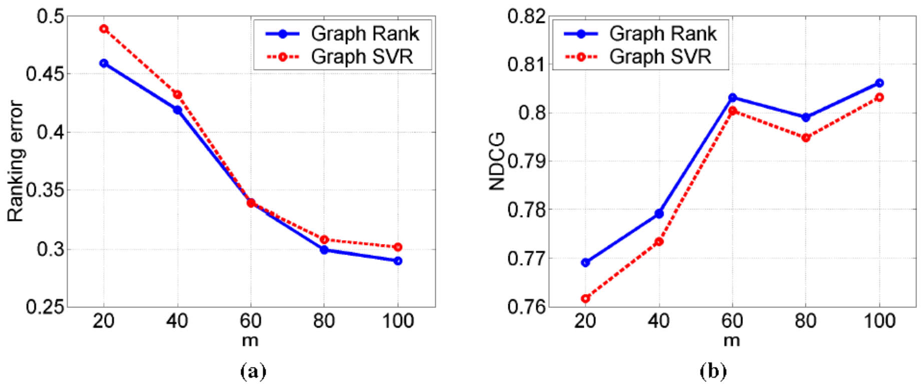

Fig. 5 Results on the task of ranking compounds in a chemical similarity graph by their inhibition activity against dihydrofolate reductase (DHFR). The plots compare the performance of the graph ranking algorithm given in Sect. 3 with SVR using the same kernel; the two algorithms therefore learn a function from the same function class. Results are shown for increasing values of the training size m, in terms of (**a**) ranking error; and (**b**) NDCG. Each value shown in the plots is an average over 10 random trials. For the ranking error, lower values correspond to better ranking performance; for the NDCG, higher values correspond to better ranking performance. See text for details

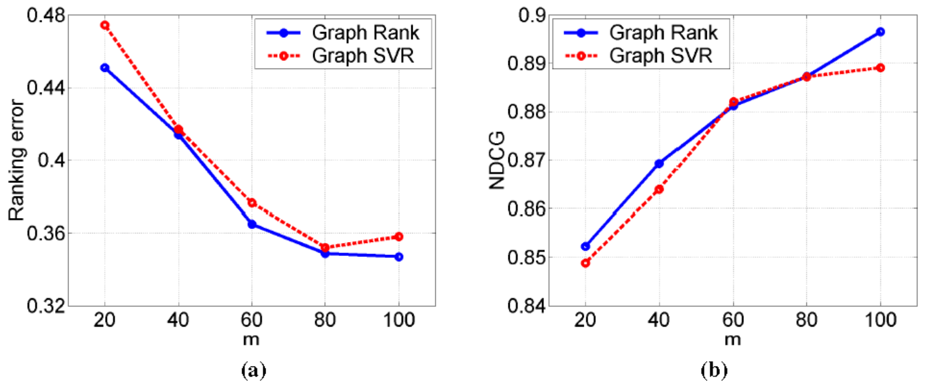

Fig. 6 Results on the task of ranking compounds in a chemical similarity graph by their inhibition activity against cyclooxygenase-2 (COX2). The plots compare the performance of the graph ranking algorithm given in Sect. 3 with SVR using the same kernel; the two algorithms therefore learn a function from the same function class. Results are shown for increasing values of the training size m, in terms of (**a**) ranking error; and (**b**) NDCG. Each value shown in the plots is an average over 10 random trials. For the ranking error, lower values correspond to better ranking performance; for the NDCG, higher values correspond to better ranking performance. See text for details

for example, in the case of DHFR (Fig. 5(b)), we see that an increase in the number of training examples sometimes results in a drop in NDCG. This is discussed further below.

7 Conclusion

Our goal in this paper has been to develop an algorithmic framework for learning ranking functions on graphs. Problems of ranking on graphs are increasingly common, and in many applications, stand to benefit from a machine learning approach that can take into account general preference examples. Our work builds on recent work on graph learning and regularization (Smola and Kondor 2003; Belkin and Niyogi 2004; Belkin et al. 2004; Zhou and Schölkopf 2004; Zhou et al. 2005; Herbster et al. 2005; Johnson and Zhang 2008). Our algorithms can be viewed as learning a ranking function in a reproducing kernel Hilbert space (RKHS) derived from the underlying graph; using results of Agarwal and Niyogi (2009), we have shown that this leads to attractive stability and generalization properties. Furthermore, we have illustrated our approach on three different graph ranking tasks in computational biology and in cheminformatics; in each case, our graph ranking algorithms outperform other graph learning approaches that learn a function from the same RKHS.

There are several questions to be explored. First, we have used the hinge ranking loss in our formulations, which results in RankSVM-type formulations (Herbrich et al. 2000; Joachims 2002) (see also Sect. 3). It may be possible to use other ranking losses that also serve as convex upper bounds on the standard pair-wise ranking loss, and that lead to alternative algorithmic formulations.

Second, for the hinge ranking loss based formulation, we have described an efficient gradient projection algorithm for solving the dual quadratic program (QP) resulting from the graph ranking optimization problem; for a preference graph $\Gamma = (V, \Sigma, \tau)$, this takes $O(n_\Sigma^2/\varepsilon^2)$ time to converge to an ε-accurate solution, where n_Σ is the number of objects in V included in Σ (see Appendix A). It may be possible to use for example ideas of Chapelle and Keerthi (2010) to obtain yet more efficient algorithms that directly solve the primal problem.

Finally, while our algorithms showed clear performance gains over other approaches in terms of the pair-wise ranking error, they did not exhibit as good performance in terms of the average precision or NDCG. This is not surprising since the pair-wise ranking error, which is the quantity optimized by our algorithms, measures the global ranking accuracy, whereas the average precision and the NDCG measure local accuracy at the top of a ranking. Nevertheless, in many applications, ranking accuracy at the top is of considerable importance; indeed, there has been much interest recently in algorithms for optimizing ranking performance measures that focus on accuracy at the top of a returned ranking (Burges et al. 2007; Yue et al. 2007; Xu and Li 2007; Chapelle et al. 2007; Cossock and Zhang 2008; Taylor et al. 2008; Chakrabarti et al. 2008; Rudin 2009; Chapelle and Wu 2010). We are not aware of any such algorithms in the context of graph ranking problems.

Acknowledgements The author would like to thank Partha Niyogi for discussions related to regularization and algorithmic stability, Michael Collins for discussions related to gradient-based optimization algorithms, and Shiladitya Sengupta for suggesting the kinase ranking task. Thanks also to Archana Kamal for help with plotting some of the graphs in Fig. 1. This work was supported in part by the National Science Foundation (NSF) under Grant No. DMS-0732334. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the author and do not necessarily reflect the views of the NSF.

Appendix A: Gradient projection algorithm for solving graph ranking QP of (14)

Consider the QP in (14) that results from taking the Lagrangian dual of the optimization problem in our graph ranking framework. This is a QP over $|\Sigma|$ variables $\{\alpha_{ij} : (v_i, v_j) \in \Sigma\}$. Let $Q(\alpha)$ denote the objective:

$$Q(\alpha) = \frac{1}{2} \sum_{(v_i,v_j)\in\Sigma} \sum_{(v_k,v_l)\in\Sigma} \alpha_{ij}\alpha_{kl} \left(\widetilde{L}^+_{ik} - \widetilde{L}^+_{jk} - \widetilde{L}^+_{il} + \widetilde{L}^+_{jl}\right) - \sum_{(v_i,v_j)\in\Sigma} \alpha_{ij}\tau(v_i, v_j). \qquad (35)$$

The constraint set Ω can be described as

$$\Omega = \left\{\alpha : 0 \le \alpha_{ij} \le \frac{C}{|\Sigma|}\ \forall (v_i, v_j) \in \Sigma\right\}. \qquad (36)$$

These are simple box constraints; therefore, we can solve the QP using a gradient projection algorithm that starts at some initial values $\alpha^{(1)}$, and on each iteration t, updates $\alpha^{(t)}$ using a gradient and projection step:

$$\alpha^{(t+1)} \leftarrow \mathcal{P}_\Omega\left(\alpha^{(t)} - \eta_t \nabla^{(t)}\right), \qquad (37)$$

where $\eta_t > 0$ is a learning rate; $\nabla^{(t)}$ is the gradient of the objective $Q(\alpha)$ evaluated at $\alpha^{(t)}$; and $\mathcal{P}_\Omega$ denotes Euclidean projection to the above constraint set Ω. The projection to the box constraints above is straightforward: values of α_{ij} outside the interval $[0, C/|\Sigma|]$ are simply clipped to the interval. It is well known from standard results in optimization (Bertsekas 1999) that if $\eta_t = \eta/\sqrt{t}$ for some constant $\eta > 0$, then gradient projection converges to an ε-accurate solution (in our case, a solution α for which $|Q(\alpha) - Q^*| < \varepsilon$, where Q^* denotes the optimal value) in $O(1/\varepsilon^2)$ iterations. In our case, each iteration takes $O(n_\Sigma^2)$ time, where

$$n_\Sigma = \left|\left\{v_i \in V : (v_i, v_j) \in \Sigma \text{ or } (v_j, v_i) \in \Sigma \text{ for some } v_j \in V\right\}\right|, \qquad (38)$$

thus leading to $O(n_\Sigma^2/\varepsilon^2)$ time for convergence to an ε-accurate solution. See algorithm.

Algorithm Gradient Projection for Graph Ranking QP

```
Inputs:
    Preference graph $\Gamma = (V, \Sigma, \tau)$
    Laplacian pseudo-inverse $\widetilde{\mathbf{L}}^+$ (or $\widetilde{L}^+_{ij}$ for $i, j$ such that $v_i$ and $v_j$ included
    in preference examples $\Sigma$)
    Parameters $C, t_{\max}, \eta$
Initialize:
    $\alpha^{(1)}_{ij} \leftarrow C/(1000|\Sigma|)\ \ \forall (v_i, v_j) \in \Sigma$ (initialize to some small values)
For $t = 1$ to $t_{\max}$ do:
  • $\alpha^{(t+1/2)} \leftarrow \alpha^{(t)} - \frac{\eta}{\sqrt{t}} \nabla^{(t)}$
  • For each $(v_i, v_j) \in \Sigma$:
        If $\alpha^{(t+1/2)}_{ij} < 0$ Then $\alpha^{(t+1)}_{ij} \leftarrow 0$
        Else If $\alpha^{(t+1/2)}_{ij} > C/|\Sigma|$ Then $\alpha^{(t+1)}_{ij} \leftarrow C/|\Sigma|$
        Else $\alpha^{(t+1)}_{ij} \leftarrow \alpha^{(t+1/2)}_{ij}$
Output:
    $\alpha^{(t^*)}$, where $t^* = \arg\min_{1 \le t \le t_{\max}+1} Q(\alpha^{(t)})$
```

Appendix B: Proof of Theorem 3

The proof is based on the proof of a similar result of Herbster et al. (2005), which was given for the unnormalized Laplacian of an unweighted graph.

Proof of Theorem 3 Since $\widetilde{\mathbf{L}}^+$ is positive semi-definite, we have $\widetilde{L}^+_{ii} \ge 0$. If $\widetilde{L}^+_{ii} = 0$, the result holds trivially. Therefore assume $\widetilde{L}^+_{ii} > 0$. Then $\exists j$ such that $\widetilde{L}^+_{ij} < 0$ (since for all i, $\sum_{j=1}^n \widetilde{L}^+_{ij}\sqrt{d_j} = 0$; this is due to the fact that the vector $(\sqrt{d_1}, \ldots, \sqrt{d_n})^T$ is an eigenvector of $\widetilde{\mathbf{L}}^+$ with eigenvalue 0). Let P_{ij} denote (the set of edges in) the shortest path in G from v_i to v_j (shortest in terms of number of edges; such a path exists since G is connected); let r be the number of edges in this path. Since $\|\mathbf{a}\|_1 \le \sqrt{r}\|\mathbf{a}\|_2$ for any $\mathbf{a} \in \mathbb{R}^r$, we have

$$\sum_{(v_k,v_l)\in P_{ij}} \left(\frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right)^2 \ge \frac{1}{r} \left(\sum_{(v_k,v_l)\in P_{ij}} \left| \frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right| \right)^2 . \tag{39}$$

Now, we have

$$\begin{aligned} \sum_{(v_k,v_l)\in P_{ij}} \left| \frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right| &\ge \sum_{(v_k,v_l)\in P_{ij}} \left(\frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right) \\ &= \frac{\widetilde{L}^+_{ii}}{\sqrt{d_i}} - \frac{\widetilde{L}^+_{ij}}{\sqrt{d_j}} \\ &> \frac{\widetilde{L}^+_{ii}}{\sqrt{d_i}}, \end{aligned} \tag{40}$$

where the equality follows since all other terms in the sum cancel out, and the last inequality follows since $\widetilde{L}^+_{ij} < 0$. Furthermore, it is easy to show that for any $\mathbf{f} \in \mathbb{R}^n$,

$$\mathbf{f}^T \widetilde{\mathbf{L}} \mathbf{f} = \frac{1}{2} \sum_{(v_k, v_l) \in E} w(v_k, v_l) \left(\frac{f_k}{\sqrt{d_k}} - \frac{f_l}{\sqrt{d_l}} \right)^2 . \tag{41}$$

Then we have

$$\begin{aligned} \widetilde{L}^+_{ii} &= (\widetilde{\mathbf{L}}^+_i)^T \widetilde{\mathbf{L}} (\widetilde{\mathbf{L}}^+_i) \\ &= \frac{1}{2} \sum_{(v_k, v_l) \in E} w(v_k, v_l) \left(\frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right)^2 \\ &\geq \frac{1}{2} \cdot 2 \sum_{(v_k, v_l) \in P_{ij}} w(v_k, v_l) \left(\frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right)^2 \\ &\geq w_{\min} \sum_{(v_k, v_l) \in P_{ij}} \left(\frac{\widetilde{L}^+_{ik}}{\sqrt{d_k}} - \frac{\widetilde{L}^+_{il}}{\sqrt{d_l}} \right)^2 , \end{aligned} \tag{42}$$

where the second equality follows from (41) (applied to $\mathbf{f} = \widetilde{\mathbf{L}}^+_i$, the ith column of $\widetilde{\mathbf{L}}^+$), and the first inequality follows since E contains both (v_k, v_l) and (v_l, v_k) for all edges $(v_k, v_l) \in P_{ij}$. Combining (39), (40), and (42), we thus get that

$$\widetilde{L}^+_{ii} \geq \frac{w_{\min}}{r} \frac{(\widetilde{L}^+_{ii})^2}{d_i},$$

which gives

$$\widetilde{L}^+_{ii} \leq \frac{r d_i}{w_{\min}}.$$

The result follows since $r \leq \rho$ and $d_i \leq d$. □

References

Aerts, S., Lambrechts, D., Maity, S., Van Loo, P., Coessens, B., De Smet, F., Tranchevent, L.C., De Moor, B., Marynen, P., Hassan, B., Carmeliet, P., & Moreau, Y. (2006). Gene prioritization through genomic data fusion. *Nature Biotechnology*, *24*(5), 537–544.

Agarwal, S. (2006). Ranking on graph data. In *Proceedings of the 23rd international conference on machine learning*.

Agarwal, A., & Chakrabarti, S. (2007). Learning random walks to rank nodes in graphs. In *Proceedings of the 24th international conference on machine learning*.

Agarwal, S., & Niyogi, P. (2009). Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research*, *10*, 441–474.

Agarwal, S., & Sengupta, S. (2009). Ranking genes by relevance to a disease. In *Proceedings of the 8th annual international conference on computational systems bioinformatics*.

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*, *6*, 393–425.

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*(17), 3389–3402.

Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems* (Vol. 14).

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

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. In *Proceedings of the 17th annual conference on learning theory*.

Bertsekas, D. (1999). *Nonlinear programming* (2nd ed.). Nashua: Athena Scientific.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th international world Wide Web conference*.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on machine learning*.

Burges, C. J. C., Ragno, R., & Le, Q. V. (2007). Learning to rank with non-smooth cost functions. In *Advances in neural information processing systems* (Vol. 19). Cambridge: MIT Press.

Chakrabarti, S., Khanna, R., Sawant, U., & Bhattacharyya, C. (2008). Structured learning for non-smooth ranking losses. In *Proceedings of the 14th ACM conference on knowledge discovery and data mining*.

Chapelle, O., & Keerthi, S. S. (2010, to appear). Efficient algorithms for ranking with SVMs. *Information Retrieval Journal*. doi:101007/s10791-009-9109-9

Chapelle, O., & Wu, M. (2010, to appear). Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval Journal*. doi:101007/s10791-009-9110-3

Chapelle, O., Le, Q., & Smola, A. (2007). Large margin optimization of ranking measures. In *Proceedings of the NIPS-2007 workshop on machine learning for Web search*.

Chu, W., & Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation*, *19*(3), 792–815.

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

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

Clemencon, S., Lugosi, G., & Vayatis, N. (2008). Ranking and empirical minimization of U-statistics. *Annals of Statistics*, *36*, 844–874.

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., & Rastogi, A. (2007). Magnitude-preserving ranking algorithms. In *Proceedings of 24th international conference on machine learning*.

Cossock, D., & Zhang, T. (2008). Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory*, *54*(11), 5140–5154.

Crammer, K., & Singer, Y. (2005). Online ranking by projecting. *Neural Computation*, *17*(1), 145–175.

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. In *Advances in large margin classifiers* (pp. 115–132).

Herbster, M., Pontil, M., & Wainer, L. (2005). Online learning over graphs. In *Proceedings of 22nd international conference on machine learning*.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the ACM conference on knowledge discovery and data mining*.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th international conference on machine learning*.

Johnson, R., & Zhang, T. (2008). Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, *54*(1), 275–288.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, *46*(5), 604–632.

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*.

Ma, X., Lee, H., Wang, L., & Sun, F. (2007). CGI: a new approach for prioritizing genes by combining gene expression and protein-protein interaction data. *Bioinformatics*, *23*(2), 215–221.

Morrison, J. L., Breitling, R., Higham, D. J., & Gilbert, D. R. (2005). GeneRank: using search engine technology for the analysis of microarray experiments. *BMC Bioinformatics*, *6*, 233.

Rajaram, S., & Agarwal, S. (2005). Generalization bounds for k-partite ranking. In *Proceedings of the NIPS-2005 workshop on learning to rank*.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Rudin, C. (2009). The p-norm push: a simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, *10*, 2233–2271.

Rudin, C., Cortes, C., Mohri, M., & Schapire, R. E. (2005). Margin-based ranking meets boosting in the middle. In *Proceedings of the 18th annual conference on learning theory*.

Smola, A. J., & Kondor, R. (2003). Kernels and regularization on graphs. In *Proceedings of the 16th annual conference on learning theory*.

Sutherland, J. J., O'Brien, L. A., & Weaver, D. F. (2004). A comparison of methods for modeling quantitative structure-activity relationships. *Journal of Medicinal Chemistry*, *47*(22), 5541–5554.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 1st international conference on Web search and data mining*.

Tenenbaum, J., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: 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*(17), 6559–6563.

Xu, J., & Li, H. (2007). AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the 30th ACM SIGIR conference on research and development in information retrieval*.

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. In *Proceedings of the 30th ACM SIGIR conference on research and development in information retrieval* (pp. 271–278).

Zhou, D., & Schölkopf, B. (2004). A regularization framework for learning from graph data. In *ICML workshop on statistical relational learning*.

Zhou, D., Weston, J., Gretton, A., Bousquet, O., & Schölkopf, B. (2004). Ranking on data manifolds. In *Advances in neural information processing systems* (Vol. 16).

Zhou, D., Huang, J., & Schölkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on machine learning*.