The Infinite Push: A New Support Vector Ranking Algorithm that Directly Optimizes Accuracy at the Absolute Top of the List

Shivani Agarwal*

Abstract

Ranking problems have become increasingly important in machine learning and data mining in recent years, with applications ranging from information retrieval and recommender systems to computational biology and drug discovery. In this paper, we describe a new ranking algorithm that directly maximizes the number of relevant objects retrieved at the absolute top of the list. The algorithm is a support vector style algorithm, but due to the different objective, it no longer leads to a quadratic programming problem. Instead, the dual optimization problem involves $l_{1,\infty}$ constraints; we solve this dual problem using the recent $l_{1,\infty}$ projection method of Quattoni et al (2009). Our algorithm can be viewed as an l_∞-norm extreme of the l_p-norm based algorithm of Rudin (2009) (albeit in a support vector setting rather than a boosting setting); thus we refer to the algorithm as the 'Infinite Push'. Experiments on real-world data sets confirm the algorithm's focus on accuracy at the absolute top of the list.

Keywords: Ranking; support vector machines (SVMs); area under ROC curve (AUC); $l_{1,\infty}$ projections.

1 Introduction

The problem of ranking objects has received increasing attention in machine learning and data mining in recent years. Ranking problems arise in a wide variety of domains: in information retrieval, one wants to rank documents according to relevance to a query; in recommendation systems, one wants to rank products according to a user's likes and dislikes; in drug discovery, one wants to rank chemical compounds according to their activity against a therapeutic target.

Following the development of classical ranking algorithms such as RankSVM [22, 23, 28], RankBoost [19], and RankNet [9], there has been much interest recently in ranking algorithms that optimize ranking performance near the top of the ranked list. Indeed, while early ranking algorithms such as RankSVM, RankBoost, and RankNet all focused on the pairwise ranking accuracy (fraction of correctly ranked pairs of objects), newer ranking algorithms have focused on optimizing ranking performance measures such as the average precision or the discounted cumulative gain (DCG), both of which emphasize ranking accuracy at the top of the list [8, 34, 12, 33, 16, 30, 10, 25, 13].

In this paper, we describe a new ranking algorithm that directly optimizes accuracy at the absolute top of the list. For simplicity, we consider the bipartite setting [19, 3] in which objects are either relevant (positive) or irrelevant (negative); in this case, our algorithm maximizes (an approximation to) the number of relevant objects retrieved at the absolute top of the list. The algorithm is a support vector style algorithm, but due to the different objective, it no longer leads to a quadratic programming problem. Instead, the dual optimization problem involves $l_{1,\infty}$ constraints; we solve this dual problem using the recent $l_{1,\infty}$ projection method of Quattoni et al [27]. Our algorithm can be viewed as an l_∞-norm extreme of the l_p-norm based algorithm of Rudin [29] (albeit in a support vector setting rather than a boosting setting); thus we refer to the algorithm as the 'Infinite Push'. Experiments on real-world data sets confirm the algorithm's focus on accuracy at the absolute top of the list.

Our main contributions in this paper can be summarized as follows:

- We give a new support vector ranking algorithm called the Infinite Push that directly optimizes ranking performance at the absolute top of the list.
- We give a fast training algorithm for standard RankSVM using the same gradient projection framework that is used for implementing the Infinite Push.
- We provide a generalization bound for the new Infinite Push algorithm, thereby answering in the affirmative an open question about generalization properties for the l_∞-norm posed in [29].
- We provide an empirical comparison of the Infinite Push ranking algorithm with RankSVM, SVMMAP [34], RankBoost, and the P-Norm Push [29], confirming that the Infinite Push indeed optimizes accuracy at the absolute top of the list.

*Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India.

The rest of the paper is organized as follows. After some preliminaries in Section 2, we describe the Infinite Push algorithm in detail in Section 3. Section 4 outlines a fast training algorithm for standard RankSVM that makes use of the same gradient projection framework that we use in implementing the Infinite Push. Section 5 provides a generalization bound for the new Infinite Push algorithm. Section 6 contains our experimental results, followed by a brief summary in Section 7.

2 Preliminaries

We consider the bipartite ranking setting [19, 3] in which objects are either relevant (positive) or irrelevant (negative), and the goal is to rank relevant objects above irrelevant ones. Specifically, let X be an instance space. Given a training sample $S = (S_+, S_-) \in X^m \times X^n$, where $S_+ = (x_1^+, \ldots, x_m^+)$ are positive examples in X and $S_- = (x_1^-, \ldots, x_n^-)$ are negative examples, the goal is to learn a ranking function $f : X \to \mathbb{R}$ that ranks positive instances higher than negative ones (where f is considered to rank an instance x higher than an instance x' if $f(x) > f(x')$, and vice-versa).[1] Such a setting arises in a variety of applications: in information retrieval, one wants to rank relevant documents higher than irrelevant ones; in drug discovery, one wants to rank active compounds higher than inactive ones.

As noted above, classical ranking algorithms such as RankSVM, RankBoost and RankNet focus on optimizing the pairwise ranking accuracy; in the bipartite setting, this corresponds to maximizing the area under the receiver operating characteristic (ROC) curve (AUC) [19, 15, 3]. For example, consider a training sample S consisting of $m = 4$ positive and $n = 6$ negative instances. Table 2 shows the scores assigned to these instances by two different ranking functions f_1 and f_2; Figure 2 shows plots of (scaled versions of) the corresponding ROC curves.[2] As can be seen from Figure 2, both functions have the same AUC of 19/24. However, if we look at the ranking accuracy at the top, f_2 clearly performs better than f_1: it retrieves 3 positives at the top, compared to just 1 for f_1.

Algorithms such as RankSVM, RankBoost or RankNet that optimize the AUC would not be able to distinguish between f_1 and f_2. Our goal is to design an algorithm that would pick f_2 over f_1.

[1] It is important to note that bipartite ranking is different from binary classification; see for example [15, 3].

[2] The ROC curve of a ranking function $f : X \to \mathbb{R}$ with respect to a sample $S = (S_+, S_-) \in X^m \times X^n$ plots the true positive rate (fraction of positives in S_+ correctly classified as positive) versus the false positive rate (fraction of negatives in S_- misclassified as positive) for classifiers of the form $\text{sign}(f(x) - \theta)$, as the threshold θ ranges from ∞ to $-\infty$.

Table 1: Scores assigned by two ranking functions f_1 and f_2 to a sample containing $m = 4$ positive and $n = 6$ negative instances.

	x_1^+	x_2^+	x_3^+	x_4^+	x_1^-	x_2^-	x_3^-	x_4^-	x_5^-	x_6^-
f_1	9.7	7.3	5.2	4.0	8.7	6.3	3.9	2.7	1.1	0.8
f_2	9.5	8.1	7.2	1.5	6.3	5.1	4.4	3.1	2.7	0.9

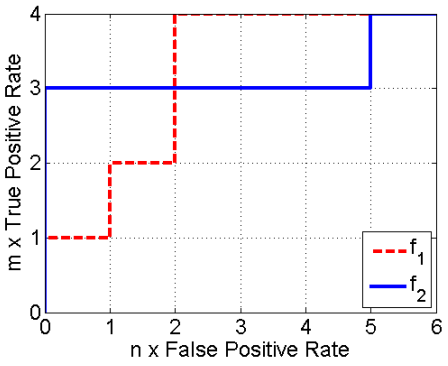

Figure 1: ROC curves for the two ranking functions shown in Table 2.

3 The Infinite Push Algorithm

Given a training sample $S = (S_+, S_-) \in X^m \times X^n$, what we would like is to find a ranking function $f : X \to \mathbb{R}$ that maximizes the number of positives retrieved at the top. In other words, we would like to maximize the number of positives ranked above the highest-ranking negative. Equivalently, we would like to *minimize* the number of positives ranked *below* the highest-ranking negative.

Now, the number of positives ranked below the highest-ranking negative is actually the largest number of positives ranked below *any* negative, which as a fraction of the total number of positives m, can be written as[3]

$$R_\infty(f; S) = \max_{1 \le j \le n} \left(\frac{1}{m} \sum_{i=1}^{m} \mathbf{1}_{(f(x_i^+) < f(x_j^-))} \right), \tag{3.1}$$

where $\mathbf{1}_{(\cdot)}$ is 1 if the argument is true and 0 otherwise. The ∞ subscript here refers to the maximum over j, which corresponds to taking the l_∞-norm of the vector containing the n terms in the parentheses. Thus, in order to maximize the number of positives retrieved at

[3] In our implementation, we use
$$\mathbf{1}_{(f(x_i^+) < f(x_j^-))} + \tfrac{1}{2}\mathbf{1}_{(f(x_i^+) = f(x_j^-))}$$
rather than $\mathbf{1}_{(f(x_i^+) < f(x_j^-))}$ to account for ties.

the absolute top of the list, we are interested in finding a ranking function that minimizes the above quantity.

While minimizing the above quantity directly is hard due to its discrete nature, we can minimize instead a convex upper bound on it. In particular, consider the hinge ranking loss, which for f, x^+, x^- is defined as

$$\ell_{\mathrm{H}}(f, x^+, x^-) = \big(1 - (f(x^+) - f(x^-))\big)_+ , \tag{3.2}$$

where $u_+ = \max(u, 0)$. The above loss is clearly a convex upper bound on $\mathbf{1}_{(f(x^+)<f(x^-))}$. We shall minimize a regularized form of

$$R^{\mathrm{H}}_{\infty}(f; S) = \max_{1\le j\le n} \left(\frac{1}{m} \sum_{i=1}^{m} \ell_{\mathrm{H}}(f, x_i^+, x_j^-) \right). \tag{3.3}$$

Specifically, the algorithm we describe will solve the following optimization problem over an appropriate reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ of real-valued functions on X:

$$\min_{f\in\mathcal{F}} \left[R^{\mathrm{H}}_{\infty}(f; S) + \frac{\lambda}{2}\|f\|^2_{\mathcal{F}} \right], \tag{3.4}$$

where $\|f\|_{\mathcal{F}}$ denotes the RKHS norm of f in $\mathcal{F}$ and where $\lambda > 0$ is an appropriate regularization parameter.

Before describing the algorithm for the case of a general RKHS $\mathcal{F}$, consider first the linear case, where $X = \mathbb{R}^d$ for some d and $\mathcal{F}$ is the class of ranking functions $f : \mathbb{R}^d \to \mathbb{R}$ given by $f(x) = w \cdot x$ for some $w \in \mathbb{R}^d$. In this case Eq. (3.4) becomes

$$\min_{w\in\mathbb{R}^d} \left[R^{\mathrm{H}}_{\infty}(w; S) + \frac{\lambda}{2}\|w\|^2 \right], \tag{3.5}$$

where $\|w\|$ denotes the Euclidean (l_2) norm of w and where we have used w in place of f in $R^{\mathrm{H}}_{\infty}(w; S)$. Expanding the above equation using the definition of $R^{\mathrm{H}}_{\infty}(w; S)$ (see Eq. (3.3)), we have

$$\min_{w\in\mathbb{R}^d} \left[\frac{\lambda}{2}\|w\|^2 + \max_{1\le j\le n} \left(\frac{1}{m} \sum_{i=1}^{m} \big(1 - w \cdot (x_i^+ - x_j^-)\big)_+ \right) \right]. \tag{3.6}$$

Introducing slack variables ξ_{ij} ($1 \le i \le m, 1 \le j \le n$) corresponding to the max in the hinge loss terms $\big(1 - w \cdot (x_i^+ - x_j^-)\big)_+ = \max\big(1 - w \cdot (x_i^+ - x_j^-), 0\big)$ and writing $C = 1/\lambda$, we can write the above optimization problem as

$$\min_{w,\xi_{ij}} \left[\frac{1}{2}\|w\|^2 + C \max_{1\le j\le n} \left(\frac{1}{m} \sum_{i=1}^{m} \xi_{ij} \right) \right]$$

subject to

$$\begin{aligned} \xi_{ij} &\ge 1 - w \cdot (x_i^+ - x_j^-) \qquad \forall i, j \\ \xi_{ij} &\ge 0 \qquad \forall i, j. \end{aligned} \tag{3.7}$$

This is where the technical difference between the proposed algorithm and the standard RankSVM algorithm [22, 23] (see also Section 4) becomes clear: instead of an average over the slack variables ξ_{ij} over both i, j (as is the case in RankSVM), the objective in the above optimization problem now contains a maximum over j. Introducing a further slack variable ξ corresponding to this max, the above optimization problem becomes

$$\min_{w,\xi,\xi_{ij}} \left[\frac{1}{2}\|w\|^2 + \frac{C}{m}\xi \right]$$

subject to

$$\begin{aligned} \xi &\ge \sum_{i=1}^{m} \xi_{ij} \qquad \text{for } j = 1, \ldots, n \\ \xi_{ij} &\ge 1 - w \cdot (x_i^+ - x_j^-) \qquad \forall i, j \\ \xi_{ij} &\ge 0 \qquad \forall i, j. \end{aligned} \tag{3.8}$$

Next, introducing Lagrange multipliers and taking the dual then results in the following optimization problem in the mn variables $\{\alpha_{ij} : 1 \le i \le m, 1 \le j \le n\}$ (the Lagrange multipliers corresponding to the first set of inequalities in Eq. (3.8) disappear in the dual):

$$\min_{\alpha_{ij}} \left[\frac{1}{2} \sum_{i,j} \sum_{k,l} \alpha_{ij}\alpha_{kl}(x_i^+ - x_j^-) \cdot (x_k^+ - x_l^-) - \sum_{i,j} \alpha_{ij} \right]$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} \Big(\max_{1\le i\le m} \alpha_{ij} \Big) &\le \frac{C}{m} \\ \alpha_{ij} &\ge 0 \quad \forall i, j. \end{aligned} \tag{3.9}$$

Unlike standard RankSVM, this is not a quadratic program due to the $l_{1,\infty}$ constraints on the α_{ij}. Nevertheless, it is a convex optimization problem and can be solved efficiently using a gradient projection algorithm. Specifically, let $Q(\alpha)$ denote the (quadratic) objective function in Eq. (3.9), and let $\Omega \subset \mathbb{R}^{mn}$ denote the feasible set, *i.e.*, the set of $\alpha \in \mathbb{R}^{mn}$ satisfying the constraints in Eq. (3.9). Then the gradient projection algorithm starts with some initial value $\alpha^{(1)}$ for α, and on each iteration t, updates $\alpha^{(t)}$ using a gradient and projection step:

$$\alpha^{(t+1)} \longleftarrow \mathcal{P}_{\Omega}\Big(\alpha^{(t)} - \eta_t \nabla Q(\alpha^{(t)})\Big), \tag{3.10}$$

where $\eta_t > 0$ is a learning rate; ∇Q is the gradient of Q; and $\mathcal{P}_{\Omega}$ denotes Euclidean projection onto Ω. The gradient computation for the linear case can be done in $O(mnd)$ time. For the projection step, we use a recent method for $l_{1,\infty}$ projections due to [27], followed by a simple projection onto the non-negativity constraints (achieved by setting negative values of α_{ij} to 0); as described in [27], this takes $O(mn \log(mn))$ time.

Algorithm Infinite Push

Inputs:

Training sample $S = (S_+, S_-) \in X^m \times X^n$

Kernel function $K : X \times X \to \mathbb{R}$

Parameters $C, t_{\max}, \eta_0$

Initialize:

$$\alpha_{ij}^{(1)} \leftarrow \frac{C}{1000mn} \quad \forall 1 \le i \le m, 1 \le j \le n \qquad \text{(initialize to some small values)}$$

For $t = 1$ to $t_{\max}$ do:

- [Gradient step] $\alpha^{(t+1/2)} \leftarrow \alpha^{(t)} - \frac{\eta_0}{\sqrt{t}} \nabla Q(\alpha^{(t)})$
- [Projection step] $\alpha^{(t+1)} \leftarrow \mathcal{P}_\Omega(\alpha^{(t+1/2)})$ (using method of [27])

Output:

$$f(x) = \sum_{i,j} \alpha_{ij}^{t^*} \left(K(x_i^+, x) - K(x_j^-, x) \right), \qquad \text{where } t^* = \arg\min_{1 \le t \le (t_{\max}+1)} Q(\alpha^{(t)})$$

Figure 2: The Infinite Push algorithm. Here Q and Ω refer to the objective function and constraint set in Eq. (3.12), respectively.

Moreover, it is well known from standard results in the optimization literature [6] that if $\eta_t = \eta_0/\sqrt{t}$ for some constant $\eta_0 > 0$, then gradient projection converges to an optimal solution, and in particular, reaches a solution whose objective value is within ϵ of the optimal in $O(1/\epsilon^2)$ iterations. Thus, our algorithm for solving Eq. (3.9) takes a total of $O((mnd + mn\log(mn))/\epsilon^2)$ time to reach an ϵ-optimal solution. On solving for α, the solution to the original problem in Eq. (3.5) can be recovered as

$$w = \sum_{i,j} \alpha_{ij} (x_i^+ - x_j^-) \,. \tag{3.11}$$

Let us now return to the general case, when X is any instance space and $\mathcal{F}$ is an RKHS corresponding to a kernel function $K : X \times X \to \mathbb{R}$. In this case, a similar derivation as above results in a dual optimization problem corresponding to Eq. (3.4) that takes a similar form as in the linear case, but with dot products replaced with appropriate kernel evaluations:

$$\min_{\alpha_{ij}} \left[\frac{1}{2} \sum_{i,j} \sum_{k,l} \alpha_{ij} \alpha_{kl} \, \phi_K(x_i^+, x_j^-, x_k^+, x_l^-) - \sum_{i,j} \alpha_{ij} \right]$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} \left(\max_{1 \le i \le m} \alpha_{ij} \right) &\le \frac{C}{m} \\ \alpha_{ij} &\ge 0 \quad \forall i, j, \end{aligned} \tag{3.12}$$

where

$$\begin{aligned} \phi_K(x, y, z, u) &= K(x, z) - K(x, u) \\ &\quad - K(y, z) + K(y, u) \,. \end{aligned} \tag{3.13}$$

For K symmetric and positive semi-definite, this is again a convex optimization problem, and the same gradient projection algorithm as above can be used here as well. The only difference is that in this case, gradient computations take $O((m+n)^2)$ time, leading to a total of $O(((m+n)^2 + mn\log(mn))/\epsilon^2)$ time for the algorithm to reach an ϵ-optimal solution to Eq. (3.12). On solving for α, the solution to the original problem in Eq. (3.4) is then given by

$$f(x) = \sum_{i,j} \alpha_{ij} \left(K(x_i^+, x) - K(x_j^-, x) \right) \,. \tag{3.14}$$

The general algorithm is outlined in Figure 2.

It is worth pointing out that the P-Norm Push algorithm of Rudin [29] minimizes a convex upper bound on the l_p-norm analogue of the quantity in Eq. (3.1) for finite p:

$$R_p(f; S) = \left(\frac{1}{n} \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} \mathbf{1}_{(f(x_i^+) < f(x_j^-))} \right)^p \right)^{1/p} . \tag{3.15}$$

The algorithm of [29], which makes use of the exponential loss as an upper bound resulting in a boosting style algorithm, applies for finite p; while this does not directly maximize the number of positives at the absolute top, values of $p > 1$ do emphasize the lower left portion of the ROC curve, with larger values of p causing a greater emphasis on the top portion of the ranked list. As described above, the use of the hinge loss in an RKHS setting allows for direct minimization of the l_∞-norm based objective, corresponding to directly maximizing accuracy at the absolute top of the list.

Algorithm RankSVM (Fast training algorithm via gradient projection method)

Inputs:

Training sample $S = (S_+, S_-) \in X^m \times X^n$

Kernel function $K : X \times X \to \mathbb{R}$

Parameters $C, t_{\max}, \eta_0$

Initialize:

$$\alpha_{ij}^{(1)} \leftarrow \frac{C}{1000mn} \quad \forall 1 \le i \le m, 1 \le j \le n \qquad \text{(initialize to some small values)}$$

For $t = 1$ to $t_{\max}$ do:

- [Gradient step] $\quad \alpha^{(t+1/2)} \leftarrow \alpha^{(t)} - \frac{\eta_0}{\sqrt{t}} \nabla Q(\alpha^{(t)})$
- [Projection step]

  For $1 \le i \le m, 1 \le j \le n$:

  If $\left(\alpha_{ij}^{(t+1/2)} < 0\right)$ **Then** $\alpha_{ij}^{(t+1)} \leftarrow 0$

  Else If $\left(\alpha_{ij}^{(t+1/2)} > \frac{C}{mn}\right)$ **Then** $\alpha_{ij}^{(t+1)} \leftarrow \frac{C}{mn}$

  Else $\alpha_{ij}^{(t+1)} \leftarrow \alpha_{ij}^{(t+1/2)}$

Output:

$$f(x) = \sum_{i,j} \alpha_{ij}^{t^*} \left(K(x_i^+, x) - K(x_j^-, x)\right), \qquad \text{where } t^* = \arg\min_{1 \le t \le (t_{\max}+1)} Q(\alpha^{(t)})$$

Figure 3: Fast training algorithm for RankSVM. Here Q refers to the objective function in Eq. (4.20).

4 Fast Algorithm for RankSVM Training

In this section we describe a fast training algorithm for standard RankSVM that makes use of the same gradient projection framework used above for implementing the Infinite Push algorithm. The training algorithm we give is also much simpler to implement than standard RankSVM implementations, which typically need to employ specialized quadratic program (QP) solvers.

As discussed above, the (bipartite) RankSVM algorithm finds a ranking function that maximizes (approximately) the AUC. In particular, it can be verified [15, 3, 29] that the AUC of a ranking function f with respect to a sample $S = (S_+, S_-) \in X^m \times X^n$ can be written as

$$\text{AUC}(f; S) = 1 - R_1(f; S), \tag{4.16}$$

where $R_1(f; S)$ is as defined in Eq. (3.15) with $p = 1$. The (bipartite) RankSVM algorithm [22, 23, 28] maximizes a regularized bound on the AUC, or equivalently, minimizes a regularized bound on $R_1(f; S)$, over an RKHS $\mathcal{F}$ as follows:

$$\min_{f \in \mathcal{F}} \left[R_1^{\text{H}}(f; S) + \frac{\lambda}{2} \|f\|_{\mathcal{F}}^2 \right], \tag{4.17}$$

where $R_1^{\text{H}}(f; S)$ is defined analogously to $R_1(f; S)$ but with the hinge loss $\ell_{\text{H}}(f, x_i^+, x_j^-)$ replacing the $\mathbf{1}_{(f(x_i^+) < f(x_j^-))}$ terms. Expanding Eq. (4.17) using the definition of $R_1^{\text{H}}(f; S)$, we then have

$$\min_{f \in \mathcal{F}} \left[\frac{\lambda}{2} \|f\|_{\mathcal{F}}^2 + \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(1 - (f(x_i^+) - f(x_j^-))\right)_+ \right]. \tag{4.18}$$

Introducing slack variables ξ_{ij} corresponding to the max in the hinge loss terms $\left(1 - (f(x_i^+) - f(x_j^-))\right)_+ = \max\left(1 - (f(x_i^+) - f(x_j^-)), 0\right)$ and writing $C = 1/\lambda$, we have

$$\min_{f, \xi_{ij}} \left[\frac{1}{2} \|f\|_{\mathcal{F}}^2 + \frac{C}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \xi_{ij} \right]$$

subject to

$$\begin{aligned} \xi_{ij} &\ge 1 - (f(x_i^+) - f(x_j^-)) \qquad \forall i, j \\ \xi_{ij} &\ge 0 \qquad\qquad\qquad\qquad\quad \forall i, j. \end{aligned} \tag{4.19}$$

Introducing Lagrange multipliers and taking the dual then leads to the following optimization problem in the mn variables $\{\alpha_{ij} : 1 \le i \le m, 1 \le j \le n\}$:

$$\min_{\alpha_{ij}} \left[\frac{1}{2} \sum_{i,j} \sum_{k,l} \alpha_{ij} \alpha_{kl} \, \phi_K(x_i^+, x_j^-, x_k^+, x_l^-) - \sum_{i,j} \alpha_{ij} \right]$$

subject to

$$0 \le \alpha_{ij} \le \frac{C}{mn} \qquad \forall i, j, \tag{4.20}$$

where K is the kernel function associated with $\mathcal{F}$ and ϕ_K is as defined in Eq. (3.13); on solving for α, the solution to the original problem in Eq. (4.17) is recovered as in Eq. (3.14). This is a convex quadratic program (QP), and the usual approach is to solve this using a standard QP solver, which can take $O((mn)^3)$ time. However, comparing Eq. (4.20) with Eq. (3.12), it should be clear that a similar gradient projection algorithm as outlined for the Infinite Push algorithm in Section 3 can be used in this case as well. Indeed, the only difference between the two dual problems is in the constraint set, and in fact projection onto the box constraints in Eq. (4.20) is especially simple: values of each α_{ij} outside the interval $[0, C/mn]$ are simply clipped to the interval. Thus the projection step in this case takes only $O(mn)$ time; the gradient step takes $O((m+n)^2)$ time as before, leading to a total of $O((m+n)^2/\epsilon^2)$ time to reach an ϵ-optimal solution. For completeness, the algorithm is outlined in Figure 3. We used this algorithm to implement RankSVM in our experiments; the algorithm was also used recently in [2].

We note that alternative algorithms for fast RankSVM training that operate on the primal problem have been proposed recently in [11]. We also note that the gradient projection algorithm we describe here is not restricted to the bipartite setting, but can be used to efficiently solve the dual RankSVM problem in more general settings as well.

5 Generalization Bound for the Infinite Push Algorithm

In this section we provide a generalization bound for the Infinite Push algorithm; this is essentially a uniform convergence bound for R_∞. The bound makes use of covering numbers and is similar in broad outline to a bound presented in [29] for R_p for finite p, but with a crucial difference to accommodate the l_∞-norm (thereby also answering in the affirmative an open question in [29] regarding the existence of such bounds for the l_∞ case).

As is common in the bipartite ranking setting [19, 3, 29], assume positive instances are drawn randomly and independently according to a distribution $\mathcal{D}_+$ on X, and negative instances according to a distribution $\mathcal{D}_-$ on X. Let $\mathcal{D} = (\mathcal{D}_+, \mathcal{D}_-)$, and define

$$R_\infty(f;\mathcal{D}) = \sup_{x^- \in \mathrm{supp}(\mathcal{D}_-)} \left(\mathbf{E}_{x^+ \sim \mathcal{D}_+} \left[\mathbf{1}_{(f(x_i^+) < f(x_j^-))} \right] \right), \tag{5.21}$$

where $\mathrm{supp}(\mathcal{D}_-)$ denotes the support of $\mathcal{D}_-$.

Next, for any $\gamma > 0$, define the γ-margin ranking loss as

$$\ell_\gamma(f, x^+, x^-) = \mathbf{1}_{(f(x^+) - f(x^-) < \gamma)}, \tag{5.22}$$

and let $R^\gamma_\infty(f;S)$ and $R^\gamma_\infty(f;\mathcal{D})$ be defined as follows:

$$R^\gamma_\infty(f;S) = \max_{1 \le j \le n} \left(\frac{1}{m} \sum_{i=1}^{m} \ell_\gamma(f, x_i^+, x_j^-) \right), \tag{5.23}$$

$$R^\gamma_\infty(f;\mathcal{D}) = \sup_{x^- \in \mathrm{supp}(\mathcal{D}_-)} \left(\mathbf{E}_{x^+ \sim \mathcal{D}_+} \left[\ell_\gamma(f, x^+, x^-) \right] \right). \tag{5.24}$$

Also define

$$\ell_\gamma(f, \mathcal{D}_+, x^-) = \mathbf{E}_{x^+ \sim \mathcal{D}_+} \left[\ell_\gamma(f, x^+, x^-) \right]. \tag{5.25}$$

Then we have the following:

THEOREM 5.1. (GENERALIZATION BOUND) *Let $\mathcal{F}$ be a class of real-valued functions on X. Let $\epsilon, \gamma > 0$. Then*

$$\mathbf{P}_{S \sim \mathcal{D}_+^m \times \mathcal{D}_-^n} \left[\exists f \in \mathcal{F} : R_\infty(f;\mathcal{D}) > R^\gamma_\infty(f;S) + \epsilon \right] \le \mathcal{N}\left(\mathcal{F}, \frac{\epsilon\gamma}{8}\right) \left(2n\, e^{-m\epsilon^2/8} + \left(\sup_{f \in \mathcal{F}} \delta_{\mathcal{D}}(f, \gamma, \epsilon/4) \right)^n \right),$$

where

$$\delta_{\mathcal{D}}(f, \gamma, \epsilon) = \mathbf{P}_{x^- \sim \mathcal{D}_-} \left[\ell_\gamma(f, \mathcal{D}_+, x^-) < R^\gamma_\infty(f;\mathcal{D}) - \epsilon \right].$$

Here $\mathcal{N}(\mathcal{F}, \epsilon)$ is the l_∞ covering number of $\mathcal{F}$ for radius ϵ, *i.e.*, the number of l_∞ balls of radius ϵ needed to cover $\mathcal{F}$. As noted above, this bound is similar to the bound presented in [29]; the only difference is in the second term in the parentheses, namely $\left(\sup_{f \in \mathcal{F}} \delta_{\mathcal{D}}(f, \gamma, \epsilon/4)\right)^n$, which arises due to the l_∞-norm and replaces a corresponding term in the bound of [29] that also involves the supremum of a quantity over $\mathcal{F}$, but has a different form since it is derived for the l_p-norm for finite p. To get an intuitive feel for the quantity $\delta_{\mathcal{D}}(f, \gamma, \epsilon)$, note that it represents the probability that the random variable $\ell_\gamma(f, \mathcal{D}_+, x^-)$ (as a function of x^-, drawn randomly from $\mathcal{D}_-$) is more than ϵ away from its supremum value $R^\gamma_\infty(f;\mathcal{D})$. For example, if $\ell_\gamma(f, \mathcal{D}_+, x^-)$ is distributed uniformly over $[0, 1]$, then $\delta_{\mathcal{D}}(f, \gamma, \epsilon) = 1 - \epsilon$. In general, the quantity $\delta_{\mathcal{D}}(f, \gamma, \epsilon)$ is bounded away from 1, and therefore the term in the bound, $\left(\sup_{f \in \mathcal{F}} \delta_{\mathcal{D}}(f, \gamma, \epsilon/4)\right)^n$, decreases exponentially as $n \to \infty$.

We briefly sketch the proof of the bound here; details are provided in Appendix A. The main step in the proof involves showing that the true γ-error $R^\gamma_\infty(f;\mathcal{D})$ is within $\epsilon/2$ of the empirical γ-error $R^\gamma_\infty(f;S)$ with high probability (over the draw of S) for any *fixed* $f \in \mathcal{F}$; the remainder of the proof then follows via more or less standard arguments for extending this to a uniform convergence result using the covering numbers of $\mathcal{F}$ (and the

observation that $R_\infty(f;\mathcal{D}) \leq R^\gamma_\infty(f;\mathcal{D})$). To bound the probability of the difference $R^\gamma_\infty(f;\mathcal{D}) - R^\gamma_\infty(f;S)$ being larger than ϵ for fixed f, one re-writes this difference so as to split the probability into two terms: one involving the difference between $R^\gamma_\infty(f;\mathcal{D})$ and $R^\gamma_\infty(f;\mathcal{D}_+,S_-) = \max_{1\leq j\leq n} \ell_\gamma(f,\mathcal{D}_+,x_j^-)$, and the second involving the difference between $R^\gamma_\infty(f;\mathcal{D}_+,S_-)$ and $R^\gamma_\infty(f;S)$. The second term can be bounded as in [29]; the first term can be bounded using a standard analysis of the convergence of the maximum of a sequence of iid random variables to the supremum value. Details of the proof are provided in Appendix A.

6 Experiments

This section provides an empirical comparison of the performance of the Infinite Push algorithm proposed in this paper with that of RankSVM [22, 23, 28], SVMMAP [34], RankBoost [19], and the P-Norm Push algorithm [29]. We conducted experiments to compare the performance of these algorithms on three different data sets: the Ionosphere data set drawn from the UCI Machine Learning Repository [18]; the Spambase data set, also drawn from the UCI Repository; and a cheminformatics data set that has been used to test virtual screening algorithms for drug discovery [24, 2].

The Infinite Push and RankSVM algorithms were implemented as described in Sections 3 and 4; SVMMAP, which is a support vector algorithm for optimizing the average precision, was implemented using publicly available code [34]; RankBoost and the P-Norm Push were implemented as described in [29] (RankBoost is simply a special case of the P-Norm Push with $p = 1$). To facilitate comparison, all algorithms were implemented to learn a linear ranking function (for the support vector algorithms, a linear kernel was used; for the P-Norm Push, individual features were used as weak rankers, leading to a linear function as in [29]).

In each case, we compared the performance of the ranking functions learned by the different algorithms using four different performance measures: the area under the ROC curve (AUC); the number of positives retrieved at the absolute top; the average precision; and the discounted cumulative gain (DCG). The average precision and DCG for a ranking function $f : X \to \mathbb{R}$ and sample $S = (S_+, S_-) \in X^m \times X^n$ are defined as

$$\mathrm{AP}(f;S) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{r_f(i)} \sum_{k=1}^{m} \mathbf{1}_{(f(x_i^+)<f(x_k^+))}\,, \tag{6.26}$$

$$\mathrm{DCG}(f;S) = \sum_{i=1}^{m} \frac{1}{\log_2(r_f(i)+1)}\,, \tag{6.27}$$

where

$$r_f(i) = 1 + \sum_{k=1}^{m} \mathbf{1}_{(f(x_i^+)<f(x_k^+))} + \sum_{j=1}^{n} \mathbf{1}_{(f(x_i^+)<f(x_j^-))} \tag{6.28}$$

is the overall rank of x_i^+ in the ranking of S returned by f. As noted in Section 1, the average precision and DCG also emphasize ranking accuracy at the top. For all four performance measures, a larger value corresponds to better ranking performance.

6.1 Ionosphere Data Set The first data set we used is the Ionosphere data set, drawn from the UCI Machine Learning Repository [18]; this data set was also used in the experiments reported in [29]. The data set contains 351 instances representing radar signals collected from a phased array of antennas; of these, 225 are 'good' (positive), representing signals that reflect back toward the antennas and indicating structure in the ionosphere, and 126 are 'bad' (negative), representing signals that pass through the ionosphere. Each instance in the data set is described by 33 features (a 34th feature present in the data set was zero for all instances and was ignored); in our experiments, each feature was scaled to $[0, 1]$.

To train the algorithms, we randomly divided the data set into two-thirds for training and one-third for testing (subject to both having the same proportion of positives); this was repeated 10 times. In each run, the number of iterations $t_{\max}$ in the gradient projection algorithms for RankSVM and the Infinite Push was fixed to 1000; the parameters C and η_0 were selected from the ranges $\{0.1, 1, 10, 100, 1000\}$ and $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, respectively, using 5-fold cross validation on the training set (in each case, the parameters giving the highest average precision across the 5 folds were selected). The parameter C for SVMMAP was selected similarly. The number of iterations $t_{\max}$ in the P-Norm Push was fixed to 100 as in [29].

The results are shown in Table 2; each value is an average over 10 random trials. For each performance measure, the two algorithms giving the best performance in terms of that measure are highlighted in bold typeface. As expected, the RankSVM algorithm performs well in terms of the AUC. On this data set, the SVMMAP algorithm gives the highest number of positives at the top, but the Infinite Push algorithm follows closely behind. In particular, the Infinite Push gives more positives at the top than both RankSVM and the P-Norm Push for various values of p. ROC curves for the four algorithms on a sample run (using the best value of p, in terms of number of positives at the top, for the P-Norm Push) are shown in Figure 4.

Table 2: Results on the Ionosphere data set.

Training method	AUC	Positives at Top	Average Precision	DCG
RankSVM	**0.9271**	12.1	**0.9330**	16.6200
SVMMAP	0.9207	**16.5**	**0.9357**	**16.7940**
P-Norm Push ($p = 1$; RankBoost)	0.9052	10.7	0.9086	16.4462
P-Norm Push ($p = 4$)	0.8789	6.7	0.8849	16.2743
P-Norm Push ($p = 16$)	0.9070	13.9	0.9218	**16.6527**
P-Norm Push ($p = 64$)	0.8917	9.9	0.9064	16.4970
Infinite Push	**0.9237**	**14.7**	0.9328	16.6336

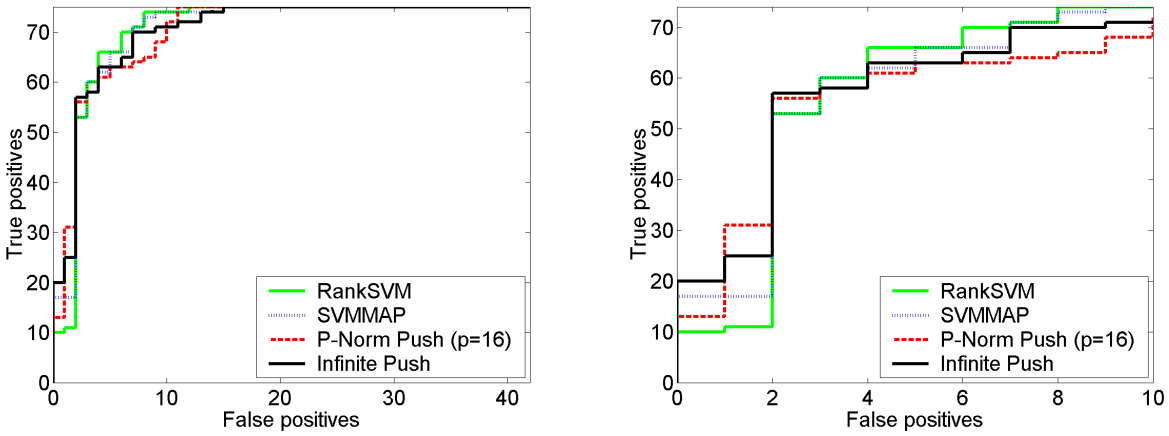

Figure 4: ROC curves (full and zoomed-in versions) on a sample run on the Ionosphere data set.

6.2 Spambase Data Set The second data set we used is the Spambase data set, also drawn from the UCI Repository [18]. This contains 4601 email messages, of which 1813 are spam. The task here is to prioritize emails such that as much of the spam as possible goes to the bottom; equivalently, if we treat the spam messages as positives, the goal is to learn a ranking that maximizes the number of positives at the top (the ranking can later be inverted to place spam at the bottom). Each email message is represented as a 57-dimensional feature vector representing various word frequencies and other attributes; each feature was scaled to lie in $[0, 1]$. In this case we used a small fraction (5%) of the data set for training and the rest for testing, again randomly divided. The parameters for the various algorithms were selected as before.

The results are shown in Table 3; again, each value is an average over 10 random trials. As before, for each performance measure, the two algorithms giving the best performance in terms of that measure are highlighted in bold typeface. Again, RankSVM gives the best performance in terms of AUC. In this case, the Infinite Push algorithm gives the highest number of positives at the top; it also gives the best performance in terms of the other measures emphasizing accuracy at the top, namely the average precision and DCG. ROC curves for the four algorithms on a sample run (using the best p for the P-Norm Push) are shown in Figure 5.

6.3 Cheminformatics Data Set The third data set we used in our experiments is a cheminformatics data set that has been used to test virtual screening algorithms in drug discovery [24, 2]. The data set contains five sets of chemical compounds that each target a different protein, and a background set of compounds that are assumed to be inactive. The five active sets contain 50 compounds each in the following classes: reversible inhibitors of cyclin-dependent kinase 2 (CDK2), cyclooxygenase-2 (COX2), factor Xa (FXa), and phosphodiesterase-5 (PDE5), and reversible antag-

Table 3: Results on the Spambase data set.

Training method	AUC	Positives at Top	Average Precision	DCG
RankSVM	**0.9418**	22.2	**0.9010**	**189.6650**
SVMMAP	0.9319	**48.6**	0.8957	189.6074
P-Norm Push ($p = 1$; RankBoost)	0.9194	12.5	0.8714	188.2330
P-Norm Push ($p = 4$)	0.8024	5.0	0.6905	180.4931
P-Norm Push ($p = 16$)	0.8293	24.1	0.7707	185.4531
P-Norm Push ($p = 64$)	0.8490	31.4	0.8143	187.3132
Infinite Push	**0.9388**	**49.9**	**0.9028**	**189.8070**

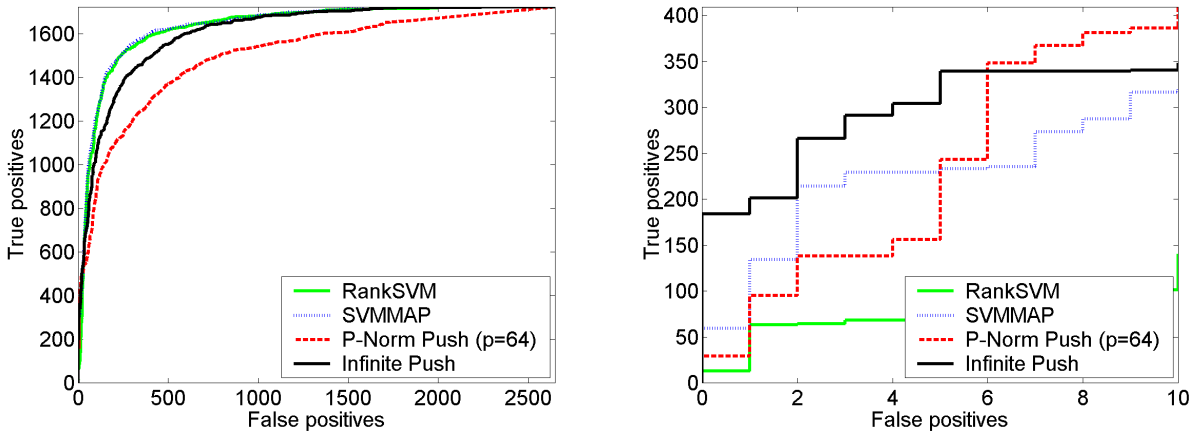

Figure 5: ROC curves (full and zoomed-in versions) on a sample run on the Spambase data set.

onists of the α_{1A} adrenoceptor (α_{1A}AR). The inactive set contains 1892 compounds drawn from the National Cancer Institute (NCI) diversity set. Details of these data sets can be found in [24].

Each compound in the above data sets was represented using a molecular fingerprint representation. Such representations are popular in virtual screening applications [20, 21, 31, 32] and effectively represent each compound as a bit vector. In particular, we used the FP2 fingerprint available with the OpenBabel chemical informatics software package [26, 2], which indexes small molecular fragments (of up to 7 atoms) and then applies a hash code that leads to a 1021-bit vector for each compound. For each of the five targets, we had 50 active compounds, and a total of 2092 inactives (including the 1892 background compounds and the 200 compounds belonging to the other four active sets). To train the algorithms, for each target, we randomly divided the 50 actives and 2092 inactives into a small fraction (10%) for training and the rest (90%) for testing (subject to both having the same proportion of actives and inactives); this was repeated 10 times. The parameters for the various algorithms were selected as before. In each case, given the training set, the goal was to rank compounds in the test set such that active compounds would be retrieved at the top of the ranked list.

The results are shown in Table 4. In this case, there were 10 random trials for each of the five targets; therefore each value is first averaged over the 10 random trials corresponding to each target, and then averaged across the five targets. As before, for each performance measure, the two algorithms giving the best performance in terms of that measure are highlighted in bold typeface. In this case, the Infinite Push algorithm actually gives the best performance in terms of all four performance measures. As would be expected, RankSVM performs well in terms of the AUC; SVMMAP performs well in terms of the average precision. ROC curves for the four algorithms on a sample run (using the best p for the P-Norm Push) are shown in Figure 6.

Table 4: Results on the cheminformatics data set.

Training method	AUC	Positives at Top	Average Precision	DCG
RankSVM	**0.8881**	**6.00**	0.4708	6.4877
SVMMAP	0.8852	5.80	**0.4778**	**6.5580**
P-Norm Push ($p = 1$; RankBoost)	0.8753	3.31	0.3714	5.9695
P-Norm Push ($p = 4$)	0.8525	3.16	0.3178	5.5329
P-Norm Push ($p = 16$)	0.8786	3.89	0.4005	6.1465
P-Norm Push ($p = 64$)	0.8805	3.96	0.3997	6.1685
Infinite Push	**0.8991**	**6.34**	**0.4910**	**6.6052**

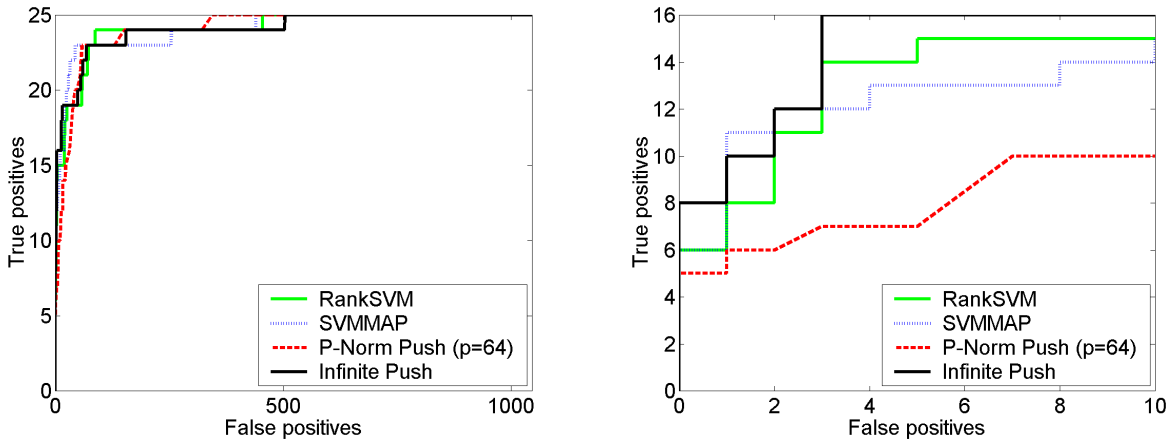

Figure 6: ROC curves (full and zoomed-in versions) on a sample run on the cheminformatics data set.

7 Conclusion and Open Questions

We have proposed a new ranking algorithm called the Infinite Push that directly maximizes the number of positives at the absolute top of the list. The algorithm can be viewed as optimizing an l_∞ extreme of the l_p-norm based objective considered in [29], albeit using a hinge ranking loss rather than the exponential loss, resulting in a support vector style algorithm. Unlike standard RankSVM, the dual of the optimization problem we consider is not a quadratic program; however it is a convex optimization problem which can be solved efficiently using a gradient projection algorithm that employs a recent $l_{1,\infty}$ projection method of [27]. Our experiments suggest that in the linear case, the proposed Infinite Push algorithm is superior in terms of accuracy at the top than RankSVM and the P-Norm Push for various values of p, and is comparable to SVMMAP, an adaptation of structural SVMs to optimizing the average precision. A benefit of the Infinite Push over SVMMAP is that it can be applied efficiently with nonlinear kernels as well; for SVMMAP, efficient algorithms are known primarily in the linear case.

There are several questions that remain open. First, we have derived a generalization bound for the proposed Infinite Push algorithm using uniform convergence arguments; it would be interesting to explore a stability-based analysis for this algorithm [7, 4]. Second, it would be of interest to understand consistency properties of the proposed algorithm; in general, with a few notable exceptions [14, 16, 17], there is limited understanding of consistency for ranking algorithms. Finally, for simplicity, we have focused in this paper on the bipartite ranking setting; however it should be possible to extend the framework proposed here to other settings. Some preliminary results in this direction are contained in [1]; we plan to investigate this further in future work.

Acknowledgments

We would like to thank Michael Collins for helpful discussions regarding gradient projection methods.

A Proof of Theorem 5.1

Proof. The crucial step in the proof is to bound the probability

$$\mathbf{P}_{S\sim\mathcal{D}_+^m\times\mathcal{D}_-^n}\Big[R^\gamma_\infty(f;\mathcal{D}) > R^\gamma_\infty(f;S)+\frac{\epsilon}{2}\Big] \tag{A.1}$$

for *fixed* $f\in\mathcal{F}$. To this end, define

$$R^\gamma_\infty(f;\mathcal{D}_+,S_-) = \max_{1\le j\le n}\ell_\gamma(f,\mathcal{D}_+,x_j^-)\,, \tag{A.2}$$

where $\ell_\gamma(f,\mathcal{D}_+,x_j^-)$ is as defined in Eq. (5.25). Then we can write

$$\begin{aligned}
&\mathbf{P}_{S\sim\mathcal{D}_+^m\times\mathcal{D}_-^n}\Big[R^\gamma_\infty(f;\mathcal{D}) - R^\gamma_\infty(f;S) > \frac{\epsilon}{2}\Big]\\
&\quad= \mathbf{P}_S\Big[R^\gamma_\infty(f;\mathcal{D}) - R^\gamma_\infty(f;\mathcal{D}_+,S-)\\
&\qquad\qquad + R^\gamma_\infty(f;\mathcal{D}_+,S-) - R^\gamma_\infty(f;S) > \frac{\epsilon}{2}\Big]\\
&\quad\le \mathbf{P}_{S_-}\Big[R^\gamma_\infty(f;\mathcal{D}) - R^\gamma_\infty(f;\mathcal{D}_+,S-) > \frac{\epsilon}{4}\Big]\\
&\qquad\qquad + \mathbf{P}_S\Big[R^\gamma_\infty(f;\mathcal{D}_+,S-) - R^\gamma_\infty(f;S) > \frac{\epsilon}{4}\Big]\\
&\quad\equiv P_1+P_2\,, \quad \text{say.}
\end{aligned} \tag{A.3}$$

We will bound P_1 and P_2 in turn; let us start with P_2. We shall find it useful to define

$$\ell_\gamma(f,S_+,x^-) = \frac{1}{m}\sum_{i=1}^m \ell_\gamma(f,x_i^+,x^-)\,. \tag{A.4}$$

We have,

$$\begin{aligned}
P_2 &= \mathbf{P}_S\Big[\max_{1\le j\le n}\ell_\gamma(f,\mathcal{D}_+,x_j^-) - \max_{1\le j\le n}\ell_\gamma(f,S_+,x_j^-) > \frac{\epsilon}{4}\Big]\\
&\le \mathbf{P}_S\Big[\max_{1\le j\le n}\Big|\ell_\gamma(f,\mathcal{D}_+,x_j^-) - \ell_\gamma(f,S_+,x_j^-)\Big| > \frac{\epsilon}{4}\Big]\\
&\le \sum_{j=1}^n \mathbf{P}_S\Big[\Big|\ell_\gamma(f,\mathcal{D}_+,x_j^-) - \ell_\gamma(f,S_+,x_j^-)\Big| > \frac{\epsilon}{4}\Big], \quad \text{by the union bound}\\
&\le n\sup_{x^-}\mathbf{P}_{S_+}\Big[\Big|\ell_\gamma(f,\mathcal{D}_+,x^-) - \ell_\gamma(f,S_+,x^-)\Big| > \frac{\epsilon}{4}\Big]\\
&\le 2n\,e^{-m\epsilon^2/8}\,, \quad \text{by Hoeffding's inequality.}
\end{aligned} \tag{A.5}$$

Next, we bound P_1. We have,

$$\begin{aligned}
P_1 &= \mathbf{P}_{S_-}\Big[\sup_{x^-}\ell_\gamma(f,\mathcal{D}_+,x^-) - \max_{1\le j\le n}\ell_\gamma(f,\mathcal{D}_+,x_j^-) > \frac{\epsilon}{4}\Big]\\
&= \mathbf{P}_{S_-}\Big[\max_{1\le j\le n}\ell_\gamma(f,\mathcal{D}_+,x_j^-) < \sup_{x^-}\ell_\gamma(f,\mathcal{D}_+,x^-) - \frac{\epsilon}{4}\Big]\\
&= \prod_{j=1}^n \mathbf{P}_{x_j^-}\Big[\ell_\gamma(f,\mathcal{D}_+,x_j^-) < \sup_{x^-}\ell_\gamma(f,\mathcal{D}_+,x^-) - \frac{\epsilon}{4}\Big], \quad \text{by the independence of } x_1^-,\ldots,x_n^-\\
&= \Big(\delta_D(f,\gamma,\epsilon/4)\Big)^n\,,
\end{aligned} \tag{A.6}$$

where $\delta_D(f,\gamma,\epsilon/4)$ is as defined in Theorem 5.1.

Combining Eqs. (A.3), (A.5) and (A.6) above gives for any fixed $f\in\mathcal{F}$:

$$\begin{aligned}
&\mathbf{P}_{S\sim\mathcal{D}_+^m\times\mathcal{D}_-^n}\Big[R^\gamma_\infty(f;\mathcal{D}) - R^\gamma_\infty(f;S) > \frac{\epsilon}{2}\Big]\\
&\quad\le 2n\,e^{-m\epsilon^2/8} + \Big(\delta_D(f,\gamma,\epsilon/4)\Big)^n\,.
\end{aligned} \tag{A.7}$$

The remainder of the proof then follows from an application of standard techniques to convert the above bound to a uniform convergence result that holds uniformly over all $f\in\mathcal{F}$, via covering number arguments. Specifically, it can be shown that by considering the centers $f_1,\ldots,f_q$ of $q=\mathcal{N}\left(\mathcal{F},\frac{\epsilon\gamma}{8}\right)$ balls that cover $\mathcal{F}$ up to a radius of $\frac{\epsilon\gamma}{8}$, the above result applied to $f_1,\ldots,f_q$, together with the fact that $R_\infty(f;S)\le R^\gamma_\infty(f;S)$, implies the uniform convergence result stated in Theorem 5.1. We omit the details since the techniques are now considered standard; detailed descriptions can be found for example in [5] or [29]. □

References

[1] S. Agarwal and M. Collins. Maximum margin ranking algorithms for information retrieval. In *Proceedings of the 32nd European Conference on Information Retrieval*, 2010.

[2] S. Agarwal, D. Dugar, and S. Sengupta. Ranking chemical structures for drug discovery: A new machine learning approach. *Journal of Chemical Information and Modeling*, 50(5):716–731, 2010.

[3] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

[4] S. Agarwal and P. Niyogi. Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research*, 10:441–474, 2009.

[5] M. Anthony and P. L. Bartlett. *Learning in Neural Networks: Theoretical Foundations*. Cambridge University Press, 1999.

[6] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[7] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[8] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with non-smooth cost functions. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

[9] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[10] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceedings of the 14th ACM Conference on Knowledge Discovery and Data Mining*, 2008.

[11] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *Information Retrieval Journal*, 13(3):201–215, 2010.

[12] O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *Proceedings of the NIPS-2007 Workshop on Machine Learning for Web Search*, 2007.

[13] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval Journal*, 13(3):216–235, 2010.

[14] S. Clemencon and N. Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8:2671–2699, 2007.

[15] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[16] D. Cossock and T. Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

[17] J. Duchi, L. Mackey, and M. I. Jordan. On the consistency of ranking algorithms. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[18] A. Frank and A. Asuncion. UCI Machine Learning Repository, 2010. http://archive.ics.uci.edu/ml.

[19] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[20] H. Geppert, T. Horvath, T. Gärtner, S. Wrobel, and J. Bajorath. Support vector machine-based ranking significantly improves the effectiveness of similarity searching using 2D fingerprints and multiple reference compounds. *Journal of Chemical Information and Modeling*, 48:742–746, 2008.

[21] H. Geppert, J. Humrich, D. Stumpfe, T. Gärtner, and J. Bajorath. Ligand prediction from protein sequence and small molecule information using support vector machines and fingerprint descriptors. *Journal of Chemical Information and Modeling*, 49:767–779, 2009.

[22] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[23] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM Conference on Knowledge Discovery and Data Mining*, 2002.

[24] R. N. Jorissen and M. K. Gilson. Virtual screening of molecular databases using a support vector machine. *Journal of Chemical Information and Modeling*, 45:549–561, 2005.

[25] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[26] OpenBabel. http://openbabel.org.

[27] A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for $l_{1,\infty}$ regularization. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[28] A. Rakotomamonjy. Optimizing area under ROC curves with SVMs. In *Proceedings of the ECAI-2004 Workshop on ROC Analysis in AI*, 2004.

[29] C. Rudin. 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, 2009.

[30] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 1st ACM International Conference on Web Search and Data Mining*, 2008.

[31] A. M. Wassermann, H. Geppert, and J. Bajorath. Searching for target-selective compounds using different combinations of multi-class support vector machine ranking methods, kernel functions, and fingerprint descriptors. *Journal of Chemical Information and Modeling*, 49:582–592, 2009.

[32] D. Wilton and P. Willett. Comparison of ranking methods for virtual screening in lead-discovery programs. *Journal of Chemical Information and Computer Science*, 43:469–474, 2003.

[33] J. Xu and H. Li. AdaRank: A boosting algorithm for information retrieval. In *Proceedings of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.

[34] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.