Ranking with a P-Norm Push

Cynthia Rudin

Center for Neural Science and Courant Institute of Mathematical Sciences, New York University
4 Washington Place, Room 809, New York, NY 10003-6603
rudin@nyu.edu

New York University Tech Report: TR2005-874

Abstract. We are interested in supervised ranking with the following twist: our goal is to design algorithms that perform especially well near the top of the ranked list, and are only required to perform sufficiently well on the rest of the list. Towards this goal, we provide a general form of convex objective that gives high-scoring examples more importance. This "push" near the top of the list can be chosen arbitrarily large or small. We choose ℓ_p-norms to provide a specific type of push; as p becomes large, the algorithm concentrates harder near the top of the list.
We derive a generalization bound based on the p-norm objective. We then derive a corresponding boosting-style algorithm, and illustrate the usefulness of the algorithm through experiments on UCI data.

1 Introduction

The problem of supervised ranking is useful in many application domains, e.g., document processing, customer service routing, and drug discovery. Many of these domains require the construction of a ranked list, yet often, only the top portion of the list is used in practice. For instance, in the setting of supervised movie ranking, the learning algorithm provides the user (an avid movie-goer) with a ranked list of movies based on preference data. We expect the user to examine the top portion of the list as a recommendation. It is possible that she never looks at the rest of the list, or examines it only briefly. Thus, we wish to make sure that the top portion of the list is correctly constructed. This is the problem on which we concentrate.

Naturally, the design of these rankings requires a tradeoff. Given the option, we would correct a misrank towards the top of the list at the expense of possibly making a new misrank towards the bottom. This type of sacrifice will have to be made; assuming a learning machine with finite capacity, the best total ranking will not often correspond to the best ranking near the top of the list. The trick is to design an algorithm that knows when a misrank occurs at the top and forces us to pay a high price for it, relative to other misranks.

We have developed a somewhat general and fairly flexible technique for solving these types of problems. In our framework, a specific price is assigned for each misrank; the misranks at the top are given higher prices, and the ones towards the bottom are less expensive. Thus, the choice of these prices determines how much emphasis (or "push") is placed closer to the top. We may only desire to incorporate a small push; it is possible, for example, that our movie-goer has seen all movies near the top of the list and needs to look farther down in order to find a movie she has not seen. Or, perhaps she may want to view the rank of a particularly bad movie. Thus, it is important that the rest of the list be sufficiently well-constructed in this case. The desired size of the push might be anywhere between very large and very small, depending on the application. There is simply a tradeoff between the size of the push, i.e., the emphasis placed near the top, and the sacrifice made farther down the list. As mentioned, some sacrifice must always be made since, as usual, we take our algorithm to have limited capacity in order to enable generalization ability.

The choice of price we pay for misranked examples corresponds to the choice of objective function used for the learning. Using the form of ranking objective introduced in Section 2, one can make the prices quite high for misranking towards the top (a big push), or one can make them moderately high (a little push), or somewhere in between. The case with equal prices (no push) corresponds to a standard objective for supervised bipartite ranking.

The algorithms we develop are motivated in the usual setting of supervised bipartite ranking. In the supervised bipartite ranking problem, each training instance has a label of +1 or -1, i.e., each movie is either a good movie or a bad movie. In this case, we want to push the bad movies away from the top of the list where the good movies are desired. The quality of the ranking can be determined by examining the Receiver Operator Characteristic (ROC) curve. In the setting where all misranks are equally priced (no push) the AUC (Area Under the ROC Curve) is precisely a constant times one minus the total standard misranking error (see [4]). However, the quantity we measure in our problem is different. We care mostly about the leftmost portion of the ROC curve for this problem, corresponding to the top of the ranked list. This is precisely the sacrifice we must make; in order make the leftmost portion of the curve higher, we must sacrifice on the total area underneath the curve.

Recently, there has been a large amount of interest in the supervised ranking problem, and especially in the bipartite problem. Freund et al. have developed the RankBoost algorithm for the general setting [8]. We inherit the setup of RankBoost, since our algorithms will also be boosting-style algorithms. Oddly, there is a recent theoretical proof that AdaBoost performs just as well for bipartite ranking as RankBoost, since both algorithms achieve the same AUC [11]. A recent algorithm of Crammer and Singer is named"PRank" [5]. Mozer et al. [10] aim to manipulate specific points of the ROC curve. The closest work to ours is that of Dekel et al. [7], who have used a similar form of objective with different specifics for the score. There is a lot of recent work on generalization bounds for supervised ranking, for instance, those of Freund et al. [8], Agarwal et al. [1], and Rudin et al. [11]. There is a body of work on reranking (see Collins [3]), and also work characterizing the difference between classification error and misranking error for the bipartite problem [4].

In Section 2, we present a general form of objective function, allowing us to incorporate a push near the top of the ranked list. In order to construct a specific case of this objective, one chooses both a loss function ℓ and a convex price function g. If the price function is very steep, it means that the push near the top is very strong. For instance, if we choose g to be a power law, $g(r) = r^p$, then a higher power p corresponds to a larger push near the top. With permissible choices of ℓ and g, the objective function will be convex, and can possibly be minimized by coordinate descent on the space of weak rankers used for the boosting. We show that our objective also provides a convex upper-bound on the "misranking error at the very top", which is directly related to the highest rank of a negative example (the most offending negative example).

In Section 3, we provide a generalization bound for a variant of our objective function, using the standard "0-1" loss function, and a p-norm price function. We expect that as p increases, more training examples are necessary to ensure the same generalization error; one cannot expect to concentrate on smaller regions of the input space without requiring more examples. Our bound does reflect this observation, as well as our experiments. We will further discuss this in Section 6.

In Section 4 we derive a coordinate descent algorithm based on the p-norm objective, with ℓ chosen as the exponential loss used for AdaBoost and RankBoost. As with AdaBoost and RankBoost, this algorithm requires normalization at each iteration in order to maintain numerical stability, and we assign this appropriately.

In Section 5, we demonstrate our p-norm ranking algorithms on UCI data in order to show their usefulness and flexibility. Namely, we illustrate that a larger push, corresponding to a higher power p, yields a very different result than a smaller push. For a larger push, the algorithm generally performs much better at the top of the list in both training and testing.

When choosing an algorithm for a particular application, one should always consider the domain of problems for which the algorithm succeeds. Of course no single algorithm is successful for every application. The final discussion of our paper is an attempt to understand, for our p-norm algorithm, precisely this domain. In order to do this, we show, in Section 6, where the algorithm "breaks", i.e., where the limit of its problem domain lies. We use the generalization bound of Section 3 to lead our discussion.

2 Motivation: A General Objective for Ranking with a Convex Push

First, some notation. The set of instances with positive labels is $\{\mathbf{x}_i\}_{i=1,\ldots,I}$, where $\mathbf{x}_i \in \mathcal{X}$. The negative instances are $\{\tilde{\mathbf{x}}_k\}_{k=1,\ldots,K}$, where $\tilde{\mathbf{x}}_k \in \mathcal{X}$. We always use i for the index over positive instances and k for the index over negative instances. In the case of the movie ranking problem, the $\mathbf{x}_i$'s are the good movies used for training, the $\tilde{\mathbf{x}}_k$'s are the bad movies, and $\mathcal{X}$ is a database of movies. Our goal is to construct a ranking function $f \in \mathcal{F}$ that gives a real valued score to each instance in $\mathcal{X}$, that is, $f : \mathcal{X} \to \mathcal{R}$. We do not care about the actual values of each instance, only the relative values; for positive-negative pair $\mathbf{x}_i, \tilde{\mathbf{x}}_k$, we do not care if $f(\mathbf{x}_i) = .4$ and $f(\tilde{\mathbf{x}}_k) = .1$, but we do care that $f(\mathbf{x}_i) > f(\tilde{\mathbf{x}}_k)$.

Let us now derive the general form of objective function promised in the introduction. For a particular negative example, we wish to reduce the number of positive examples that are ranked beneath it. In other words, for each k, we wish to make the following quantity small:

$$\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}. \tag{1}$$

Consider any two negative examples indexed by k and $\bar{k}$, such that $\tilde{\mathbf{x}}_{\bar{k}}$ is ranked below $\tilde{\mathbf{x}}_k$, i.e., $f(\tilde{\mathbf{x}}_{\bar{k}}) \leq f(\tilde{\mathbf{x}}_k)$. We have:

$$\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]} \geq \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_{\bar{k}})]}.$$

In other words, there are more positives below $\tilde{\mathbf{x}}_k$ since it is ranked higher than $\tilde{\mathbf{x}}_{\bar{k}}$.

Let us now add the push at the top end. We want to concentrate harder on negative examples that have high scores, that is, we want to push these negative examples down from the top. Since, as shown above, the highest scoring negative examples also achieve the largest values of (1), these are the examples for which we impose a larger price. Namely, for convex, non-negative, monotonically increasing function $g : \mathcal{R}_+ \to \mathcal{R}_+$, we place the following price on negative example k:

$$g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right).$$

If g is very steep, we pay an extremely large price for a high-ranked negative example. Examples of steep functions include $g(r) = \exp(r)$ and $g(r) = r^p$ for p large. If g is the identity, we pay substantially less for an offending negative example, relatively speaking. Thus we have derived an objective to minimize:

$$R_{g,\mathbf{1}}(f) := \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right).$$

If the value of $R_{g,\mathbf{1}}(f)$ is small, it means that no negative example is ranked very highly; this is exactly our design.

As usual, we cannot minimize $R_{g,\mathbf{1}}$ directly due to the 0-1 loss in the inner sum. Instead, we will minimize an upper bound, $R_{g,\ell}$, which incorporates $\ell : \mathcal{R} \to \mathcal{R}_+$, a convex, non-negative, monotonically decreasing upper bound on the 0-1 loss. Popular loss functions include the exponential, logistic, and hinge losses. We can now define the general form of objective promised in the introduction:

$$R_{g,\ell}(f) := \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \ell(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k))\right).$$

To construct a specific version of this objective, one chooses the loss ℓ, the price function g, and as usual, an appropriate hypothesis space $\mathcal{F}$ over which to minimize $R_{g,\ell}$.

For the moment, let us assume we care only about the very top of the list, that is, we wish to push the most offending negative example as far down the list as possible. Equivalently, we wish to minimize $R_{\max}$, the number of positives below the highest ranked negative example:

$$R_{\max}(f) := \max_k \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}.$$

Although it is hard to minimize $R_{\max}(f)$ directly, $R_{g,\ell}$ can give us some control over this quantity. Namely, the following relationships exist between $R_{g,\ell}$, $R_{g,\mathbf{1}}$ and $R_{\max}$.

Theorem 1.

$$Kg\left(\frac{1}{K} R_{\max}(f)\right) \leq R_{g,\mathbf{1}}(f) \leq R_{g,\ell}(f) \quad \textit{and} \quad R_{g,\mathbf{1}}(f) \leq Kg(R_{\max}(f)).$$

Proof. The proof of the first part is as follows:

$$\begin{aligned} Kg\left(\frac{1}{K} R_{\max}(f)\right) &= Kg\left(\frac{1}{K} \max_k \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right) \leq Kg\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right) \\ &\leq \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right) = R_{g,\mathbf{1}}(f) \leq \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \ell([f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)])\right) = R_{g,\ell}(f) \end{aligned}$$

Above, we have used the fact that g is monotonic, ℓ is an upper bound on the 0-1 loss, and Jensen's inequality for convex function g. The fact that the function $Kg(\frac{1}{K} r)$ is monotonic in r adds credibility to our choice of objective $R_{g,\ell}$. For the second part of the proof, we only use the fact that g is monotonic:

$$\begin{aligned} R_{g,\mathbf{1}}(f) &= \sum_{k=1}^{K} g(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}) \leq K \max_k g(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}) \\ &= Kg(\max_k \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}) = Kg(R_{\max}(f)). \quad \square \end{aligned}$$

Theorem 1 suggests that $R_{g,\ell}$ is a reasonable quantity to minimize in order to incorporate a push at the top, e.g., in order to diminish $R_{\max}$. It is well-known that if g is especially steep, for instance $g(r) = \exp(r)$ or $g(r) = r^p$ for p large, then $g^{-1}(\sum_{k=1}^{K} g(r_k)) \approx \max_k r_k$. That is, the quantity L, for steep functions g, will approximate $R_{\max}$. For the rest of the paper, we are considering only the p-norm objectives.

3 A Generalization Bound for the p-Norm Objective

This bound is an adaptation of Rudin and Schapire [12] (also [11]), which is inspired by the works of Cucker and Smale [6], Koltchinskii and Panchenko [9], and Bousquet [2].

Assume that the positive instances $\mathbf{x}_i \in \mathcal{X}$, $i = 1, ..., I$ are chosen independently and at random (iid) from a fixed but unknown probability distribution $\mathcal{D}_+$ on $\mathcal{X}$. Assume the negative instances $\tilde{\mathbf{x}}_k \in \mathcal{X}$, $k = 1, ..., K$ are chosen iid from $\mathcal{D}_-$. We always use i for positive instances and k for negative instances. The notation $\mathbf{x} \sim \mathcal{D}$ means $\mathbf{x}$ is chosen randomly according to distribution $\mathcal{D}$. The notation $S_+ \sim \mathcal{D}_+^I$ means each of the I elements of the training set S_+ are chosen independently at random according to $\mathcal{D}_+$. Similarly for $S_- \sim \mathcal{D}_-^K$.

We now define the "true" objective function for which our algorithm has been designed. Our goal is to make the following quantity small:

$$R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f := \left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\mathbf{1}_{[f(\mathbf{x}_+)-f(\mathbf{x}_-)\leq 0]})^p\right)^{1/p} = \|\mathbb{P}_{\mathbf{x}_+\sim\mathcal{D}_+}(f(\mathbf{x}_+)-f(\mathbf{x}_-)\leq 0|\mathbf{x}_-)\|_{\mathrm{L}_p(\mathcal{X}_-,\mathcal{D}_-)}\,.$$

The empirical loss associated with $R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f$ is the following:

$$R^p_{S_+,S_-}\mathbf{1}_f := \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\mathbf{1}_{[f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k)\leq 0]}\right)^p\right)^{1/p}.$$

Here, for a particular $\tilde{\mathbf{x}}_k$, $R^p_{S_+,S_-}\mathbf{1}_f$ takes into account the average number of positive examples that have scores below $\tilde{\mathbf{x}}_k$. To make this notion more general, let us consider the average number of positive examples that have scores that are *close to* or below $\tilde{\mathbf{x}}_k$. A more general version of $R^p_{S_+,S_-}\mathbf{1}_f$ is thus defined as:

$$R^p_{S_+,S_-}\mathbf{1}^\theta_f := \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\mathbf{1}_{[f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k)\leq \theta]}\right)^p\right)^{1/p}.$$

This terminology incorporates the "margin" value θ; as usual, we suffer some loss whenever positive example $\mathbf{x}_i$ is ranked below negative example $\tilde{\mathbf{x}}_k$, but now we also suffer loss whenever $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_k$ have scores within θ of each other. Note that $R^p_{S_+,S_-}\mathbf{1}^\theta_f$ is an empirical quantity, so it can be measured for any θ. We prove a generalization bound in terms of $R^p_{S_+,S_-}\mathbf{1}^\theta_f$:

Theorem 2. *For all $\epsilon > 0, \theta > 0$, and $f \in \mathcal{F}$:*

$$\mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left[R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f \leq R^p_{S_+,S_-}\mathbf{1}^\theta_f + \epsilon\right]$$
$$\geq 1 - 2\mathcal{N}\left(\mathcal{F}, \frac{\epsilon\theta}{8}\right)\left[\exp\left[-2\left(\frac{\epsilon}{4}\right)^{2p}K\right] + \exp\left[-\frac{\epsilon^2}{8}I\right]\right].$$

This expression states that, provided I and K are large, then with high probability, the true error $R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f$ is not too much more than the empirical error $R^p_{S_+,S_-}\mathbf{1}^\theta_f$.

It is important to note the implications of this bound for scalability. Since we are only concentrating on the negative examples near the top of the ranked list (corresponding to a small chunk of negative input space), we must require more negative examples to achieve high accuracy; this is precisely what the bound shows. There is no way to avoid this, so we hope, in the interest of concentrating near the top of the ranked list, that in practice we have enough examples for the algorithm to generalize. In our experiments of 200-300 examples this has not been a problem, but our theory predicts that it may be problematic for experiments with very small data sets. We will discuss this further in Section 6 and illustrate our point experimentally. The bound above gives us a quantitative indication

of the scalability of K with p, namely, according to the bound we require order $\left(\frac{4}{\epsilon}\right)^{2(p-1)}$ times the number of negative examples in order to achieve the same accuracy as when $p = 1$. To summarize the cautionary note provided by Theorem 2, a sufficient number of examples are needed for generalization, especially for large p.

The main purpose of Theorem 2 is to provide the theoretical justification required for our choice of objective, provided a sufficient number of training examples. Let us now write an algorithm for minimizing that objective.

4 A Specific Family of Objectives and a Boosting-Style Algorithm

We now choose a specific form for our objective $R_{g,\ell}$ by choosing ℓ. We have already chosen g to be a power law, $g(r) = r^p$. From now on, ℓ will be the exponential loss $\ell(r) = \exp(-r)$. One could just as easily choose the hinge loss or another loss; we choose the exponential loss in order to compare with RankBoost. The objective when $p = 1$ is exactly that of RankBoost. The larger the value of p, the more we are concentrating at the top of the list (the larger the push). Thus we have our family of objective functions:

$$F_p(f) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} \exp[-f(\mathbf{x}_i) + f(\tilde{\mathbf{x}}_k)] \right)^p .$$

Note that F_p is not normalized to approximate $R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f$, but this can easily be accomplished via $\frac{1}{I(K)^{1/p}}(F_p(f))^{1/p}$, which is monotonically related to $F_p(f)$.

Now we describe our boosting-style approach. The hypothesis space $\mathcal{F}$ is the class of convex combinations of "weak" rankers $\{h_j\}_{j=1,\ldots,n}$, where $h_j : \mathcal{X} \to [0, 1]$. Note that we have allowed real-valued (confidence-valued) weak rankers. The function f is constructed as a normalized linear combination of the h_j's: $f = \sum_j \lambda_j h_j$, where $\boldsymbol{\lambda} \in \mathcal{R}^n$. At each iteration of the algorithm, the coefficient vector $\boldsymbol{\lambda}$ is updated. At iteration t, we denote the coefficient vector by $\boldsymbol{\lambda}_t$.

We construct a structure $\mathbf{M}$, which describes how each individual weak ranker j ranks each positive-negative pair i, k. We define $\mathbf{M}$ element-wise as: $M_{ikj} := h_j(\mathbf{x}_i) - h_j(\tilde{\mathbf{x}}_k)$. Thus, $M_{ikj} \in [-1, 1]$. Since $\mathbf{M}$ has three indices, we need to define right multiplication: $(\mathbf{M}\boldsymbol{\lambda})_{ik} := \sum_{j=1}^{n} M_{ikj}\lambda_j = \sum_{j=1}^{n} \lambda_j h_j(\mathbf{x}_i) - \lambda_j h_j(\tilde{\mathbf{x}}_k)$ for $\boldsymbol{\lambda} \in \mathcal{R}^n$. Thus, $\ell(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k))$ can now be written as $\exp(-\mathbf{M}\boldsymbol{\lambda})_{ik}$.

Remark: The function F_p is convex in $\boldsymbol{\lambda}$. This is because $\exp(-\mathbf{M}\boldsymbol{\lambda})_{ik}$ is a convex function of $\boldsymbol{\lambda}$, any sum of convex functions is convex, and any composition of convex functions is convex.

We now derive a boosting-style coordinate descent algorithm for minimizing F_p as a function of $\boldsymbol{\lambda}$, notating F_p now as $F_p(\boldsymbol{\lambda})$. We will add extra normalization to promote numerical stability in practice. There is much background material available on the convergence of similar coordinate descent algorithms (for instance [14]). We start with the objective at iteration t:

$$F_p(\boldsymbol{\lambda}_t) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} \exp[(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}] \right)^p .$$

We then compute the variational derivative along each "direction", and choose weak ranker j_t to have largest absolute variational derivative. The notation $\mathbf{e}_j$ means a vector of 0's with a 1 in the j^{th} entry.

$$j_t \in \operatorname*{argmax}_j \left| \frac{dF_p(\boldsymbol{\lambda}_t + \alpha\mathbf{e}_j)}{d\alpha}\Big|_{\alpha=0} \right|, \text{ where}$$

$$\frac{dF_p(\boldsymbol{\lambda}_t + \alpha\mathbf{e}_j)}{d\alpha}\Big|_{\alpha=0} = p \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} \exp[(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}] \right)^{p-1} \left(\sum_{i=1}^{I} -M_{ikj} \exp(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik} \right) \right] .$$

Define the vector $\mathbf{q}_t$ as usual on pairs i,k: $q_{t,ik} := \exp[(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}]$, and $\mathbf{d}_t$ as usual: $d_{t,ik} := q_{t,ik}/\sum_{ik} q_{t,ik}$. Our choice of j_t becomes (ignoring constant factors that do not affect the argmax):

$$j_t \in \operatorname*{argmax}_j \left| \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} d_{t,ik} \right)^{p-1} \sum_{i=1}^{I} d_{t,ik} M_{ikj_t} \right] \right|.$$

To update the coefficient of weak ranker j_t, we now perform a linesearch for the minimum of F_p along the j_t^{th} direction. The distance to travel in the j_t^{th} direction, denoted α_t, solves $0 = \frac{dF_p(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})}{d\alpha}\Big|_{\alpha_t}$. Incorporating renormalization and division by other constants, this equation becomes:

$$0 = \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} d_{t,ik} \exp[-\alpha_t M_{ikj_t}] \right)^{p-1} \left(\sum_{i=1}^{I} M_{ikj_t} d_{t,ik} \exp[-\alpha_t M_{ikj_t}] \right) \right]. \tag{2}$$

The value of α_t can be computed analytically in some cases, for instance, when the weak rankers are binary-valued and $p = 1$ (this is exactly RankBoost). Otherwise, we simply use a linesearch to solve this equation for α_t. To complete the algorithm, we set $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$. To avoid having to compute $\mathbf{d}_{t+1}$ directly from $\boldsymbol{\lambda}_t$, we can still perform the update by

$$d_{t+1,ik} = \frac{d_{t,ik} \exp(-\alpha_t M_{ikj_t})}{z_t} \quad \text{where} \quad z_t := \sum_{ik} d_{t,ik} \exp(-\alpha_t M_{ikj_t}).$$

The full algorithm is shown in Figure 1.

1. **Input:** $\{\mathbf{x}_i\}_{i=1,...,I}$ positive examples, $\{\tilde{\mathbf{x}}_k\}_{k=1,...,K}$ negative examples, $\{h_j\}_{j=1,...,n}$ weak classifiers, $t_{\max}$ number of iterations, p power.
2. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1,...,n$, $\quad d_{1,ik} = 1/IK$ for $i = 1,...,I$, $k = 1,...,K$ $\quad M_{ikj} = h_j(\mathbf{x}_i) - h_j(\tilde{\mathbf{x}}_k)$ for all i,k,j
3. **Loop for** $t = 1,...,t_{\max}$
   (a) $j_t \in \operatorname{argmax}_j \left| \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} d_{t,ik} \right)^{p-1} \sum_{i=1}^{I} d_{t,ik} M_{ikj_t} \right] \right|$.
   (b) Perform a linesearch for α_t. That is, find a value α_t which solves (2).
   (c) $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$, where $\mathbf{e}_{j_t}$ is 1 in position j_t and 0 elsewhere.
   (d) $z_t := \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} d_{t,ik} \right)^{p-1} \left(\sum_{i=1}^{I} d_{t,ik} \right) \right]$
   (e) $d_{t+1,ik} := d_{t,ik} \exp[(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}]/z_t$ for $i = 1,...,I$, $k = 1,...,K$
4. **Output:** $\boldsymbol{\lambda}_{t_{\max}}$

Fig. 1. Pseudocode for the "Ranking with a Convex Push" algorithm.

5 Experiments

We will now show the effect of adding a push. Namely, we will evaluate the effect of the p-norm on the leftmost portion of the ROC curve. We will sometimes focus on the value of $R_{\max}$, the number of positives scoring higher than the highest ranking negative. $R_{\max}$ is indicated exactly by the leftmost

value of the ROC curve. It is the number of "true positives" when there is one "false positive". There are numerous reasonable choices for ℓ and g, for example, popular choices for ℓ include the hinge loss, etc. Our goal is to illustrate the effect of the price g on the quality of the solution; in order to compare with RankBoost, we use the exponential loss. Here, we hope to show that $R_{\max}$, or more generally, the leftmost portion of the ROC curve, increases steadily with p. Our demonstration shows this firmly; we will see that the value of $R_{\max}$ does often increase (fairly dramatically) with p, for both training and testing.

Experiments have been performed for 6 different values of p, in each of 5 settings, for a total of 30 trials. Values of p chosen are $p = 1$(RankBoost),2, 4, 8, 16 and 64. Data for these experiments were obtained from the UCI machine learning repository [13]. We have used the data in its original form in all but one case; for the wdbc setting described below, we found that a linear combination of all features could often learn the entire training set, i.e., the problem is separable. Since we prefer to evaluate non-trivial ROC curves, the problem needs to be made more difficult; this is accomplished by using fewer features. Here are summaries of the 5 settings:

- **pima-indians-diabetes** with real features: 768 examples, 8 features. This dataset is particularly difficult, in terms of classification error, as reported on the UCI summary table. 300 randomly chosen examples were used for training, and the rest for testing. To construct weak ranker h_j, the values of attribute j were normalized to the interval $[-1, 1]$; thus there are 8 weak rankers. Our results are shown in Figure 2.

- **pima-indians-diabetes** with threshold features: 4 threshold features were obtained from each real valued feature, via $h_{\text{thresh}}(\mathbf{x}) = 1$ iff $h(\mathbf{x}) >$ thresh , and $h_{\text{thresh}}(\mathbf{x}) = 0$ otherwise. Thresholds used were chosen so that no two threshold features would be equivalent with respect to the training data. Thresholds are as follows : feature 1: 2,3,6,7, feature 2: 100,130,150,160, feature 3: 60,65,72,90, feature 4: 1,10,20,30, feature 5: 30,50,80,100, feature 6: 30,32,35,37, feature 7: 0.1,0.2,0.3,0.5, feature 8: 30,33,36,40. Again, 300 examples were used for training, and the rest for testing. See Figure 3.

- **wdbc (Wisconsin Breast Cancer)**: Out of 569 examples, 200 randomly chosen examples were used for testing and the rest for training. In order to make sure the algorithm was unable to achieve a separable solution, only the first six features (columns 3-8) were used. Again, all features were normalized to have values within $[-1, 1]$. See Figure 4.

- **tic-tac-toe**: Here, the data comes as "o", "x", and "b". We set o=0, x=1, b=2, and used all 9 features, normalized. We purposely did not build any knowledge of the game into our model, and used just the raw data. 500 examples were used for training, and the rest for testing, out of 958 total examples. See Figure 5.

- **housing (Boston Housing)**: 506 total examples, 300 used for training. In order to use the housing dataset for a binary classification problem, we used the fourth feature (which is binary) as the label y. The fourth feature describes whether a tract bounds the Charles River. Since there is some correlation between this feature and the other features, it is reasonable for our learning algorithm to predict whether the tract bounds the Charles River based on the other features, of which there are 13. Again, all features were normalized. See Figure 6.

For all experiments, the linesearch for α_t was performed using matlab's 'fminunc' subroutine. The total number of iterations, $t_{\max}$, was fixed at 200 for all experiments. We examine both training and testing curves for all settings. In agreement with our algorithm's derivation, a larger push (p large) causes the algorithm to perform better near the top of the ranked list. As mentioned in the introduction,

this ability to correct the top of the list is not without sacrifice; we do sacrifice the ranks of items farther down on the list, but we have made this choice on purpose. We believe it is important to show this sacrifice explicitly, thus full ROC curves have been included for all experiments. In no cases did we find a large deviation in the AUC, hence the sacrifice has been small. This point will be discussed further in Section 7. For each training and testing ROC curve, we show a "zoomed-in" version of the leftmost portion. We include the crossover region wherever possible, i.e., the point where the curves cross and our sacrifice begins. Often there is not a clear crossover point since the curves may cross many times, so we have just included the first crossover point and somewhat beyond.

Consider the first setting, pima-indians-diabetes with real features. Here, the value of $R_{\max}$ is significantly improved for large p; with RankBoost alone, the ranked list has only 4 positive examples before the first negative. For $p = 64$, the algorithm has 22 positives before the first negative. Furthermore, the increase in $R_{\max}$ is completely monotonic in p for both training and testing. It would be nicer to see the crossover point appear more to the right, but it is clear that the algorithm is strongly generalizing since the $R_{\max}$ values are indeed monotonic.

The results for pima-indians-diabetes with threshold features are shown in Figure 3. Again, there is complete monotonicity in $R_{\max}$ in both training and testing (although there are some ties in testing). The overall trend that we expect is present; as p increases, the leftmost part of the ROC curve tends to increase.

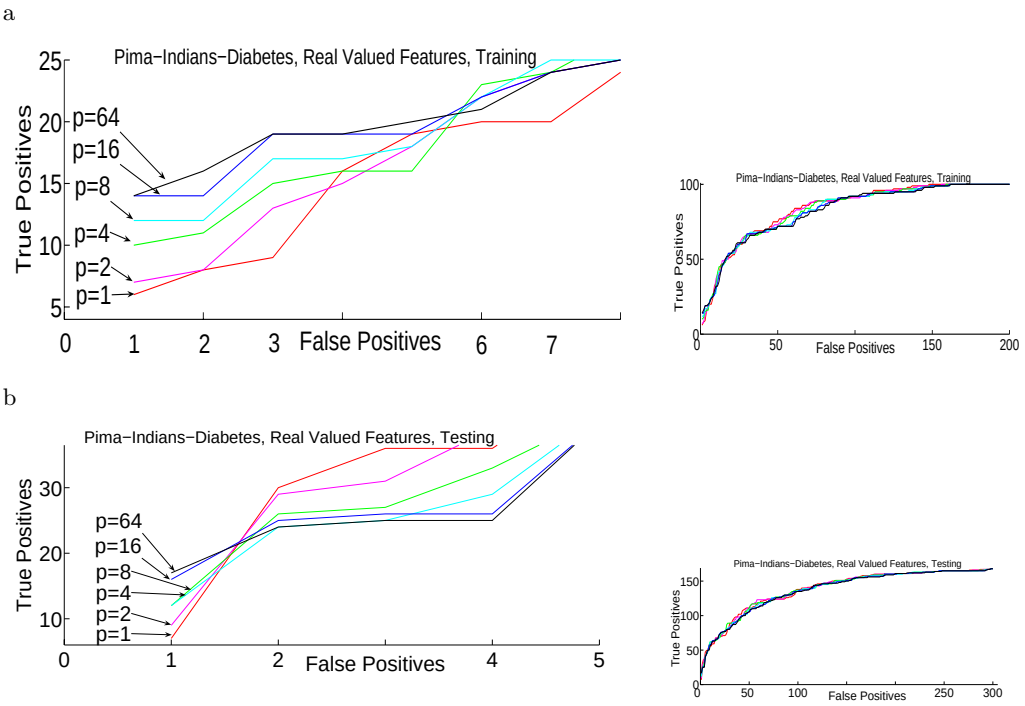


Fig. 2. Leftmost portion of ROC curves for $p = 1, 2, 4, 8, 16$ and 64, followed by full ROC curves. It is customary to perform constant scaling on the axes to obtain a rate, but we have eliminated this for clarity. The values of $R_{\max}$ are nondecreasing. (a) pima-indians-diabetes with real valued weak classifiers, training curves. (b) pima-indians-diabetes with real valued weak classifiers, testing curves.

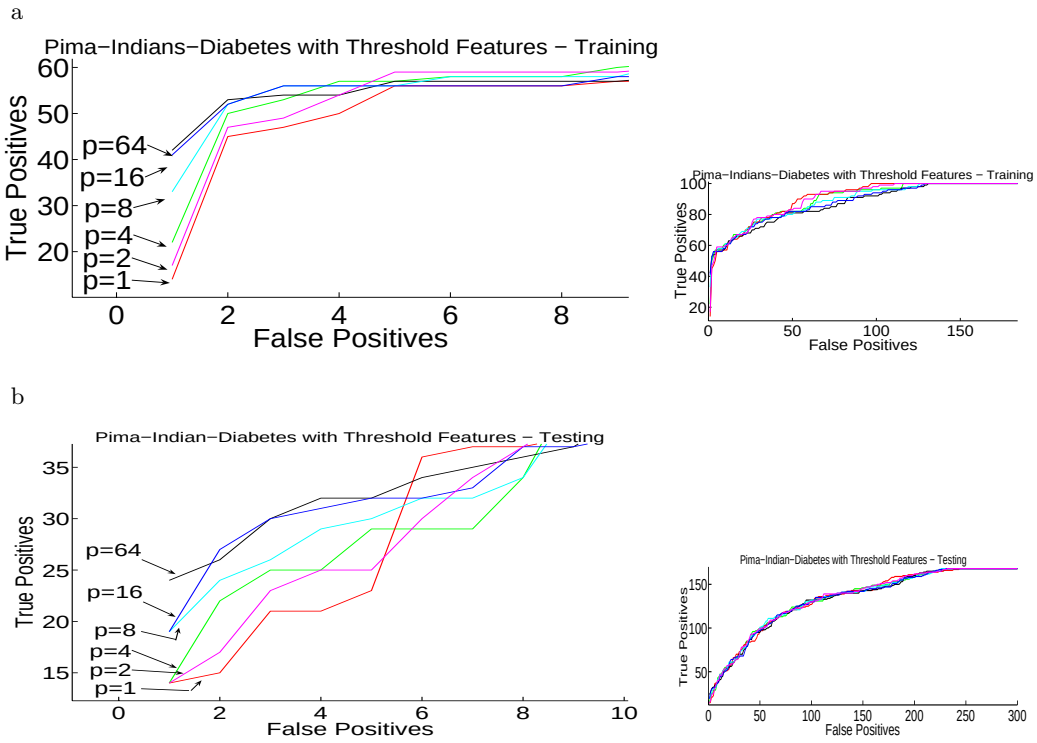


Fig. 3. (a) pima-indians-diabetes with thresholded weak classifiers, training curves. (b) pima-indians-diabetes with thresholded weak classifiers, testing curves.

For the wdbc setting, we observe the same overall trend in the leftmost part of the ROC curve, showing that the algorithm is able to generalize. The only curve out of place is the $p = 64$ curve, which is slightly below the $p = 16$ curve. It is true that more examples are needed in order to generalize to higher values of p, yet the $p = 64$ curve is still significantly improved over the $p = 1$ curve (crossing at the tenth false positive). Thus, the $p = 64$ curve is still generalizing.

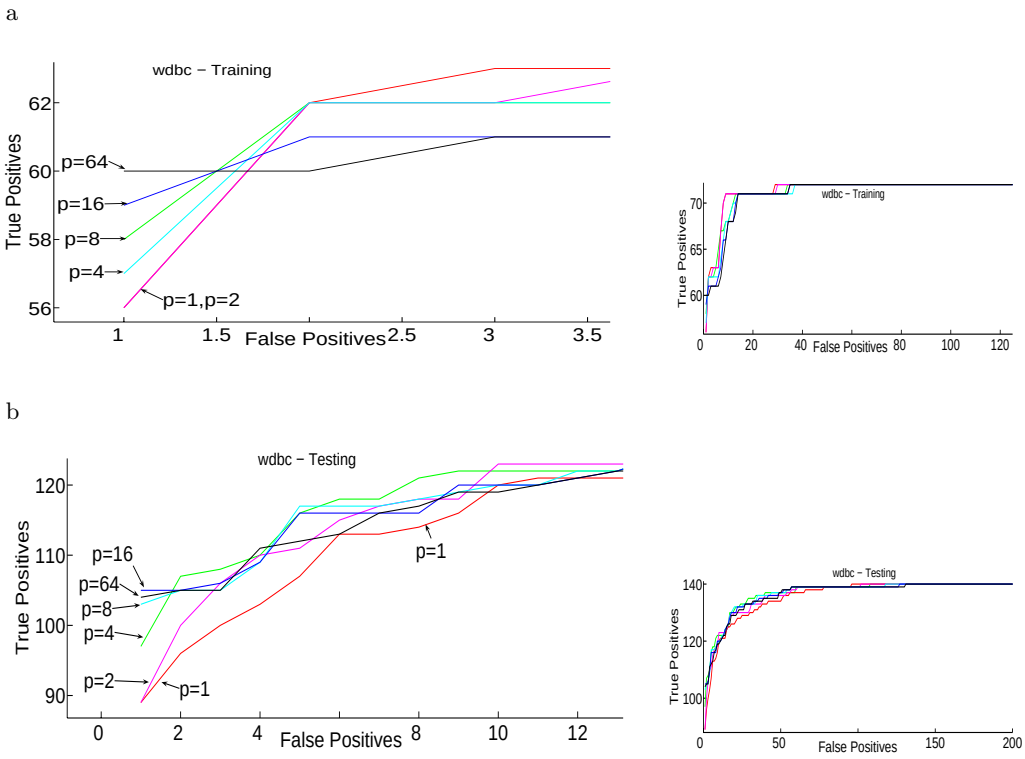


Fig. 4. (a) wdbc training curves (b) wdbc testing curves

For the tic-tac-toe setting, we again observe the expected trend, however, an outlier has been encountered during testing. Namely, there is one negative example appearing out of place near the top of the list, which disturbs the values of $R_{\max}$. Aside from this one outlier, the desired trend is present even as far out as the 16th negative example.

The housing setting yields the clearest view of the effect of the algorithm. The trend from $p = 1$ to $p = 64$ is clearly present and close to monotonic. There is a distinct crossover region, showing exactly what parts of the ROC curve are gained, and what parts are sacrificed.

6 Limitations

We have decided to include this section in order to more explicitly describe the problem domain for which the algorithm is useful. As no one algorithm is the best for every problem setting, we wish to make as clear as possible the settings in which our algorithm is meant to succeed, and in which domains it is not meant to be used. In other words, we are specifying the boundaries of the appropriate

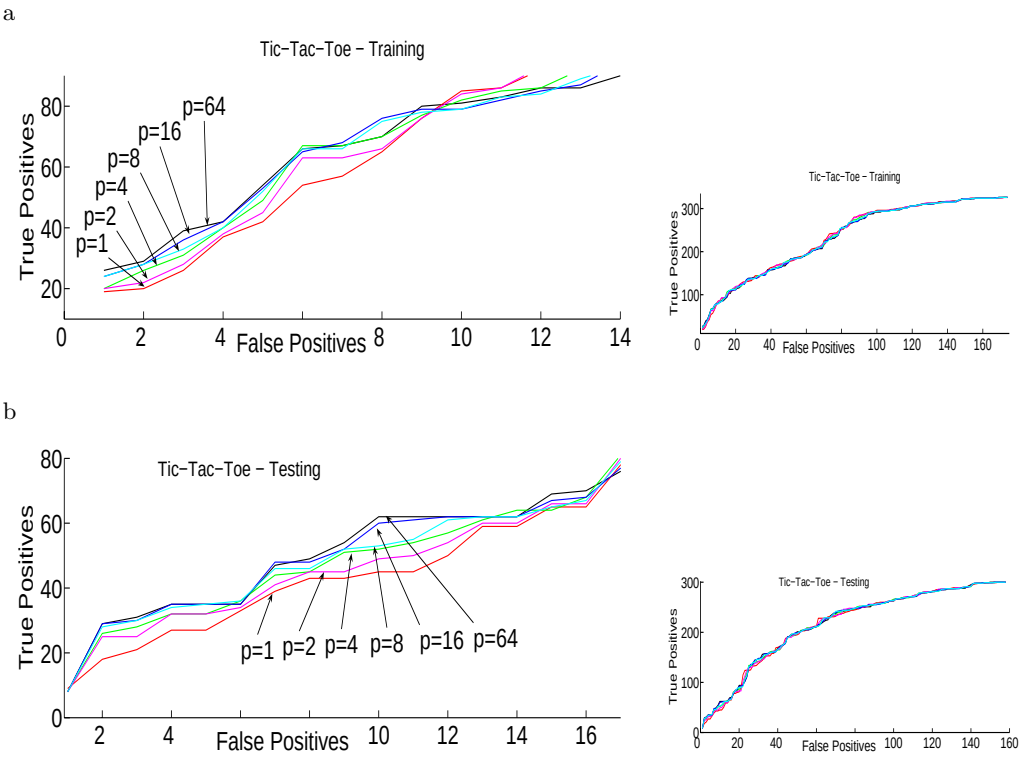
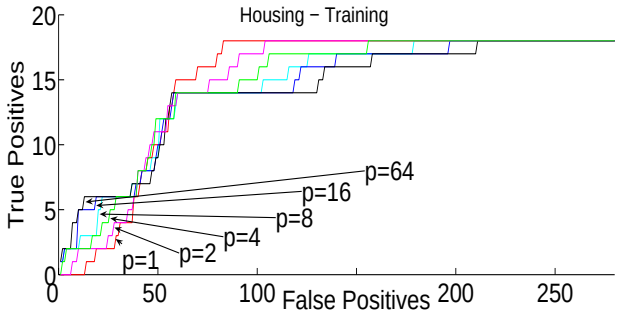


Fig. 5. (a) tic-tac-toe training curves (b) tic-tac-toe testing curves

a



b

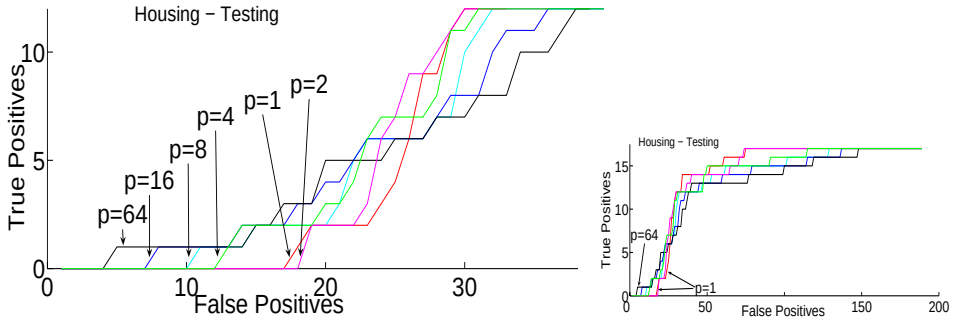


Fig. 6. (a) housing full training curve (b) housing testing curves

problem domain. To be blunt, we will show in this section where the algorithm "breaks", i.e., where it becomes not useful.

First, it is important to note that the algorithm does not produce a maximum value of the AUC (area under the ROC curve). This is by design; as mentioned earlier, if we are concentrating on the top of the ranked list, we do not care about the full ROC curve, only the leftmost part. Users who wish to concentrate on the AUC might use RankBoost ($p = 1$). As p increases, we do expect the AUC to be reduced.

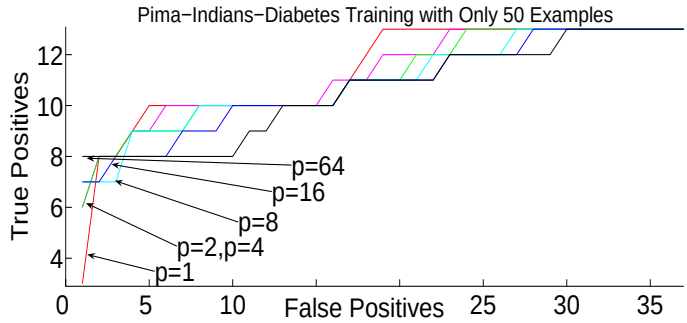
The most definitive boundary of our problem domain involves the sample size. Referring back to Section 3, the generalization bound of Theorem 2 indicates that for larger values of p, many more examples are needed in order to allow generalization ability; we are concentrating on a smaller region of the probability distribution, so this is natural. When the sample size is too small, the algorithm may still be able to generalize for smaller values of p, but for larger values, we cannot expect the training curve to represent the testing curve. For the settings shown in Section 5, we have used a few hundred examples per experiment, which is enough to allow the algorithm to generalize in these settings. Even though the wdbc setting has the p=16 and 64 curves reversed on the left, the $p = 64$ curve is still significantly better than the $p = 1$ curve, so the algorithm is still capable of generalizing here. In contrast, we now present a setting that compliments our theoretical prediction; in this setting, a larger value of p will hinder our ability to generalize, due to the small size of the training set. The setting is the pima-indian-diabetes dataset with only 50 training examples. Here, the algorithm is able to generalize for small p, but it is not able to generalize for large p. Above a certain p value, its performance degrades as p increases. Figure 7 shows this explicitly. At low values of p, the algorithm generalizes; the $p = 1$ curve lies below the $p = 2$ curve, which lies below the $p = 4$ curve. At large values of p the algorithm does not generalize; the $R_{\max}$ value for the $p = 8$ curve is below that of the $p = 1$ curve, and the $p = 16$ and 64 curves are even below this. Thus, in agreement with our theoretical prediction, the algorithm is suitable only for settings where a sufficiently large training set is available, and in this case, 50 training examples is not sufficient.

This example shows (what we believe is) the main cautionary note to experimentalists when using this algorithm, and for that matter, when using any other algorithm that concentrates on a small part of the input space.

7 Discussion and Open Problems

As mentioned in Section 5, we have shown that an increase in p tends to increase the value of $R_{\max}$, but how severe is the sacrifice that we make farther down the ranked list? All of the full ROC training curves in Section 5 (with perhaps the exception of housing) do not show any significant sacrifice, even between the $p = 1$ and $p = 64$ curves. To explain this observation, recall that we are working with learning machines of very limited capacity. The number of real valued features has not exceeded 13, and none of these learning machines are able to create a completely consistent decision boundary with respect to the training examples. In other words, there is not too much flexibility in the set of solutions that yield good rankings; the algorithm chooses the best solution from this limited choice. If a learning machine is high capacity, it has the flexibility to change the shape of the ROC curve dramatically. Accordingly, in experimental situations, a high capacity learning machine generally is able to produce a consistent (or nearly consistent) decision boundary. This is one possible reason for our sacrifice to have been small in experimental settings; otherwise the AUC would be close to 1 and the ROC curve would be a straight line, which is not so interesting from the perspective of comparison. In order to achieve a dramatic change in the ROC curves (by changing p), one might attempt to find a dataset and a high complexity hypothesis space such that no hypothesis can achieve a good decision boundary. We leave it as an interesting open problem to find such a dataset and hypothesis space, hopefully in

a



b

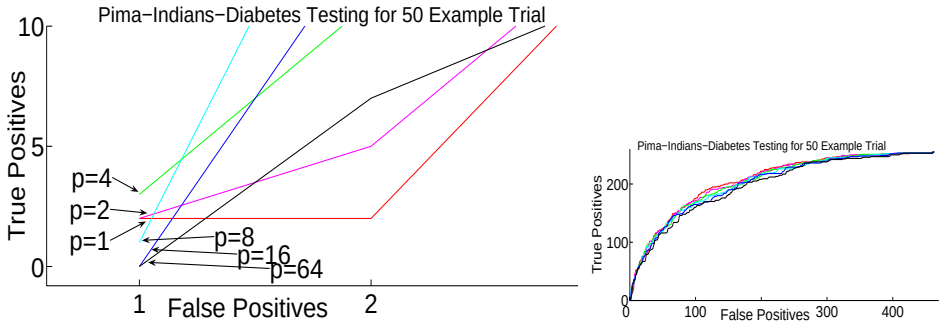


Fig. 7. The pima-indians-diabetes dataset with only 50 training examples. The algorithm is able to generalize for early values of p, but it does not generalize for large values of p. This underscores the need for a sufficiently large training set for large p values. Top: full training ROC curve. Bottom: Zoomed in version of testing ROC curve and full testing ROC curve.

a realistic setting. That is, we challenge the reader to find a dataset and hypothesis space such that an increase in p causes a dramatic change in the full ROC curve.

Another important direction for future research is the choice of loss function ℓ and price function g. The choice of loss function is a thoroughly-studied topic, however, the choice of price function adds a new dimension to this problem. One appealing possibility is to choose a non-monotonic function for g. The only algorithmic requirement is that g be convex. This matter is currently under investigation.

8 Conclusions

We have provided a method for constructing a ranked list where correctness at the top of the list is most important. Our main contribution is a general set of convex objective functions determined by a loss ℓ and price function g. A boosting-style algorithm based on a specific family of these objectives is derived. We have demonstrated the effect of a number of different price functions, and it is clear, both theoretically and empirically, that a steeper price function concentrates harder at the top of the list.

Acknowledgments Thanks to Sinan Gunturk, Rob Schapire, and Eero Simoncelli. Funding for this research is provided by an NSF BIO-division postdoctoral research fellowship.

References

1. Shivani Agarwal, Thore Graepel, Ralf Herbich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *JMLR*, 6:393–425, 2005.
2. Olivier Bousquet. New approaches to statistical learning theory. *Annals of the Institute of Statistical Mathematics*, 55(2):371–389, 2003.
3. Michael Collins. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.
4. Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*, 2004.
5. Koby Crammer and Yoram Singer. PRanking with ranking. In *Advances in Neural Information Processing Systems 14*, 2001.
6. Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, (39):1–49, 2002.
7. Ofer Dekel, Christopher Manning, and Yoram Singer. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems 16*, 2004.
8. Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*, 1998.
9. Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1), February 2002.
10. M. C. Mozer, R. Dodier, M. D. Colagrosso, C. Guerra-Salcedo, and R. Wolniewicz. Prodding the roc curve: Constrained optimization of classifier performance. In *Advances in Neural Information Processing Systems 14*, pages 1409–1415, 2002.
11. Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, 2005.
12. Cynthia Rudin and Robert E. Schapire. Margin-based ranking and adaboost. Submitted, 2005.
13. C.L. Blake S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.
14. Tong Zhang and Bin Yu. Boosting with early stopping - convergence and consistency. *The Annals of Statistics*, 2003 (to appear).

A Proof of Theorem 2

We define a Lipschitz function $\phi : \mathcal{R} \to \mathcal{R}$ (with Lipschitz constant $\text{Lip}(\phi)$) which will act as our loss function, and gives us the margin. We will eventually use the same piecewise linear definition of ϕ as Koltchinskii and Panchenko [9], but for now, we require $0 \leq \phi(z) \leq 1 \; \forall z$ and $\phi(z) = 1$ for $z < 0$. Since $\phi(z) \geq \mathbf{1}_{[z \leq 0]}$, we can define an upper bound for $R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f$:

$$R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f := \left(\mathbb{E}_{\mathbf{x}_- \sim \mathcal{D}_-}\left(\mathbb{E}_{\mathbf{x}_+ \sim \mathcal{D}_+}\phi(f(\mathbf{x}_+) - f(\mathbf{x}_-))\right)^p\right)^{1/p}.$$

By definition, $R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f \leq R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f$. The empirical error associated with $R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f$ is:

$$R^p_{S_+,S_-}\phi_f := \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k))\right)^p\right)^{1/p}.$$

First, we upper bound $R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f$ by two terms: the empirical error term $R^p_{S_+,S_-}\phi_f$, and a term characterizing the deviation of $R^p_{S_+,S_-}\phi_f$ from $R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f$ uniformly:

$$R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f \leq R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f = R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f - R^p_{S_+,S_-}\phi_f + R^p_{S_+,S_-}\phi_f \leq \sup_{\bar{f}\in\mathcal{F}}\left(R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{\bar{f}} - R^p_{S_+,S_-}\phi_{\bar{f}}\right) + R^p_{S_+,S_-}\phi_f.$$

The proof of Theorem 2 involves an upper bound on the first term. We provide this upper bound through a couple of lemmas. First, define $L(f)$ as follows:

$$L(f) := R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f - R^p_{S_+,S_-}\phi_f.$$

The following lemma is true for every training set S:

Lemma 1. *For any two functions $f_1, f_2 \in L_\infty(\mathcal{X})$,*

$$L(f_1) - L(f_2) \leq 4\text{Lip}(\phi)||f_1 - f_2||_\infty.$$

Proof. First, we rearrange the terms:

$$\begin{aligned} L(f_1) - L(f_2) &= R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{f_1} - R^p_{S_+,S_-}\phi_{f_1} - R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{f_2} + R^p_{S_+,S_-}\phi_{f_2} \\ &= [R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{f_1} - R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{f_2}] - [R^p_{S_+,S_-}\phi_{f_1} - R^p_{S_+,S_-}\phi_{f_2}]. \end{aligned} \tag{3}$$

We upper bound the second bracketed term of (3),

$$\begin{aligned} &\left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\phi(f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k))\right]^p\right]^{1/p} - \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\phi(f_2(\mathbf{x}_i) - f_2(\tilde{\mathbf{x}}_k))\right]^p\right]^{1/p} \\ &\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left|\frac{1}{I}\sum_{i=1}^{I}\phi(f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k)) - \frac{1}{I}\sum_{i=1}^{I}\phi(f_2(\mathbf{x}_i) - f_2(\tilde{\mathbf{x}}_k))\right|^p\right]^{1/p} \\ &\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}|\phi(f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k)) - \phi(f_2(\mathbf{x}_i) - f_2(\tilde{\mathbf{x}}_k))|\right]^p\right]^{1/p} \\ &\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\text{Lip}(\phi)|f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k) - f_2(\mathbf{x}_i) + f_2(\tilde{\mathbf{x}}_k)|\right]^p\right]^{1/p} \\ &\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\text{Lip}(\phi)2\sup_{\mathbf{x}}|f_1(\mathbf{x}) - f_2(\mathbf{x})|\right]^p\right]^{1/p} = \text{Lip}(\phi)2\|f_1 - f_2\|_\infty. \end{aligned}$$

Here, we have used both the triangle inequality for $\ell_p(\mathcal{R}^K)$ and the definition of the Lipschitz constant for ϕ. An identical calculation for the first bracketed term of (3) yields:

$$|R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{f_1} - R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_{f_2}| \leq 2\text{Lip}(\phi)||f_1 - f_2||_\infty.$$

Combining the two terms yields the statement of the lemma. □

The following step appears in Cucker and Smale [6]. Let $\ell_\epsilon := \mathcal{N}\left(\mathcal{F}, \frac{\epsilon}{8\text{Lip}(\phi)}\right)$, the covering number of $\mathcal{F}$ by L_∞ disks of radius $\frac{\epsilon}{8\text{Lip}(\phi)}$. Define $f_1, f_2, ..., f_{\ell_\epsilon}$ to be the centers of such a cover, i.e., the collection of L_∞ disks B_r centered at f_r and with radius $\frac{\epsilon}{8\text{Lip}(\phi)}$ is a cover for $\mathcal{F}$. In the proof of the theorem, we will use the center of each disk to act as a representative for the whole disk. So, we must show that we do not lose too much by using f_r as a representative for disk B_r.

Lemma 2. *For all $\epsilon > 0$,*

$$\mathbb{P}_{S\sim\mathcal{D}^m}\{\sup_{f\in B_r} L(f) \geq \epsilon\} \leq \mathbb{P}_{S\sim\mathcal{D}^m}\{L(f_r) \geq \frac{\epsilon}{2}\}.$$

Proof. By Lemma 1, for every training set S and for all $f \in B_r$,

$$\sup_{f\in B_r} L(f) - L(f_r) \leq 4\text{Lip}(\phi)||f - f_r||_\infty \leq 4\text{Lip}(\phi)\frac{\epsilon}{8\text{Lip}(\phi)} = \frac{\epsilon}{2}$$

Thus,

$$\sup_{f\in B_r} L(f) \geq \epsilon \quad \Longrightarrow \quad L(f_r) \geq \frac{\epsilon}{2}.$$

The statement of the lemma follows directly. □

Here is a small lemma from calculus that will be useful in the next proof.

Lemma 3. *For $a, b \in \mathcal{R}_+$, it is true that $|a^{1/p} - b^{1/p}| \leq |a - b|^{1/p}$.*

Proof. It is true that for $c \geq 0$, we have $|c - 1|^p \leq |c^p - 1|$. Let $c = a^{1/p}/b^{1/p}$. The lemma directly follows.

We now incorporate the fact that the training set is chosen randomly.

Lemma 4. *For all $\epsilon_1 > 0$,*

$$\mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}(L(f) \geq \epsilon_1) \leq 2\exp\left[-2\left(\frac{\epsilon_1}{2}\right)^{2p} K\right] + 2\exp\left[-\frac{\epsilon_1^2}{2} I\right].$$

Proof. Define

$$R^p_{S_+,\mathcal{D}_-}\phi_f := \left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i) - f(\mathbf{x}_-))\right)^p\right)^{1/p}.$$

Now,

$$\begin{aligned}
&\mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}(L(f) \geq \epsilon_1) \\
&\quad = \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}(R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f - R^p_{S_+,\mathcal{D}_-}\phi_f + R^p_{S_+,\mathcal{D}_-}\phi_f + R^p_{S_+,S_-}\phi_f \geq \epsilon_1) \\
&\quad \leq \mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f - R^p_{S_+,\mathcal{D}_-}\phi_f \geq \frac{\epsilon_1}{2}\right) + \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(R^p_{S_+,\mathcal{D}_-}\phi_f - R^p_{S_+,S_-}\phi_f \geq \frac{\epsilon_1}{2}\right) \qquad (4) \\
&\quad =: \text{term}_1 + \text{term}_2.
\end{aligned}$$

Let us bound the second term of (4), denoted term$_2$. We are going to use McDiarmid's Inequality on the negative examples. In order to do this, we will calculate the largest possible change in $(R^p_{S_+,S_-}\phi_f)^p$ that one negative example can cause. Since ϕ_f is bounded between 0 and 1, the largest possible change is:

$$\frac{1}{K}\left(\frac{1}{I}\sum_{i=1}^{I} 1\right)^p = 1/K.$$

Thus, McDiarmid's Inequality implies that for all $\epsilon_2 > 0$:

$$\begin{aligned}
&\mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left[\left|\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\tfrac{1}{I}\textstyle\sum_{i=1}^{I}\phi(f(\mathbf{x}_i) - f(\mathbf{x}_-))\right)^p - \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\left(\tfrac{1}{I}\textstyle\sum_{i=1}^{I}\phi(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k))\right)^p\right| \geq \epsilon_2\right] \\
&\qquad \leq 2\exp\left[\frac{-2\epsilon_2^2}{K\frac{1}{K^2}}\right] = 2\exp\left[-2\epsilon_2^2 K\right]. \qquad (5)
\end{aligned}$$

The following is true for any S_+, due to Lemma 3 above:

$$R^p_{S_+,\mathcal{D}_-}\phi_f - R^p_{S_+,S_-}\phi_f = \left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\mathbf{x}_-))\right)^p\right)^{1/p} - \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))\right)^p\right)^{1/p}$$
$$\leq \left|\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\mathbf{x}_-))\right)^p - \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))\right)^p\right|^{1/p}. \quad (6)$$

Combining (5) and (6) yields a bound on term$_2$. Namely, for all $\epsilon_3 > 0$:

$$\begin{aligned}
&\mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(R^p_{S_+,\mathcal{D}_-}\phi_f - R^p_{S_+,S_-}\phi_f \geq \epsilon_3\right)\\
&\quad\leq \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left[\left|\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\mathbf{x}_-))\right)^p - \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))\right)^p\right|^{1/p} \geq \epsilon_3\right]\\
&\quad= \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left[\left|\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\mathbf{x}_-))\right)^p - \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))\right)^p\right| \geq \epsilon_3^p\right]\\
&\quad\leq 2\exp\left[-2\epsilon_3^{2p}K\right]. \quad (7)
\end{aligned}$$

Letting $\epsilon_3 := \epsilon_1/2$ finishes our work on term$_2$. Now we consider term$_1$ of (4).

$$\begin{aligned}
&\mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(R^p_{\mathcal{D}_+\mathcal{D}_-}\phi_f - R^p_{S_+,\mathcal{D}_-}\phi_f \geq \frac{\epsilon_1}{2}\right)\\
&= \mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(\left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi(f(\mathbf{x}_+)-f(\mathbf{x}_-))\right)^p\right)^{1/p} - \left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\mathbf{x}_-))\right)^p\right)^{1/p} \geq \frac{\epsilon_1}{2}\right)\\
&= \mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(\left\|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi(f(\mathbf{x}_+)-f(\cdot))\right\|_{L_p(X_-,\mathcal{D}_-)} - \left\|\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\cdot))\right\|_{L_p(X_-,\mathcal{D}_-)} \geq \frac{\epsilon_1}{2}\right)\\
&\leq \mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(\left\|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi(f(\mathbf{x}_+)-f(\cdot)) - \frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\cdot))\right\|_{L_p(X_-,\mathcal{D}_-)} \geq \frac{\epsilon_1}{2}\right)\\
&\leq \mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(\left\|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi(f(\mathbf{x}_+)-f(\cdot)) - \frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\cdot))\right\|_{L_\infty(X_-,\mathcal{D}_-)} \geq \frac{\epsilon_1}{2}\right).
\end{aligned}$$

We use McDiarmid's Inequality again to complete the proof. The largest possible change in $\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i) - f(\mathbf{x}_-))$ due to the replacement of one positive example is $1/I$. Thus, for all $\mathbf{x}_-$,

$$\mathbb{P}_{S_+\sim\mathcal{D}_+^I}\left(\left|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi(f(\mathbf{x}_+)-f(\mathbf{x}_-)) - \frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\mathbf{x}_-))\right| \geq \frac{\epsilon_1}{2}\right) \leq 2\exp\left[-\frac{2\left(\frac{\epsilon_1}{2}\right)^2}{I\frac{1}{I^2}}\right] = 2\exp\left[-\frac{\epsilon_1^2 I}{2}\right].$$

Combining this result with (4) and (7) yields the statement of Lemma 4.

Proof. (Of Theorem 2) Since the D_p are a cover of $\mathcal{F}$, it is true that

$$\sup_{f\in\mathcal{F}} L(f) \geq \epsilon \quad\Longleftrightarrow\quad \exists p \leq \ell_\epsilon \text{ such that } \sup_{f\in D_p} L(f) \geq \epsilon.$$

First applying the union bound, then applying Lemma 2, and then Lemma 4, we find:

$$\begin{aligned}
\mathbb{P}_{S_+\sim\mathcal{D}^I,S_-\sim\mathcal{D}^K}\left\{\sup_{f\in\mathcal{F}} L(f) \geq \epsilon\right\} &\leq \sum_{r=1}^{\ell_\epsilon}\mathbb{P}_{S_+\sim\mathcal{D}^I,S_-\sim\mathcal{D}^K}\left\{\sup_{f\in D_r} L(f) \geq \epsilon\right\} \leq \sum_{r=1}^{\ell_\epsilon}\mathbb{P}_{S_+\sim\mathcal{D}^I,S_-\sim\mathcal{D}^K}\left\{L(f_r) \geq \epsilon/2\right\}\\
&\leq \mathcal{N}\left(\mathcal{F}, \frac{\epsilon}{8\mathrm{Lip}(\phi)}\right)\left[2\exp\left[-2\left(\frac{\epsilon}{4}\right)^{2p}K\right] + 2\exp\left[-\frac{\epsilon^2}{8}I\right]\right].
\end{aligned}$$

Now we put everything together. With probability at least

$$1-\mathcal{N}\left(\mathcal{F},\frac{\epsilon}{8\text{Lip}(\phi)}\right)\left[2\exp\left[-2\left(\frac{\epsilon}{4}\right)^{2p}K\right]+2\exp\left[-\frac{\epsilon^2}{8}I\right]\right],$$

we have

$$R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f \leq R^p_{S_+,S_-}\phi_f+\epsilon.$$

Let us choose $\phi(z)=1$ for $z\leq 0$, $\phi(z)=0$ for $z\geq\theta$, and linear in between, with slope $1/\theta$. Thus, $\text{Lip}(\phi)=1/\theta$. Since $\phi(z)\leq 1$ for $z\leq\theta$, we have:

$$R^p_{S_+,S_-}\phi_f=\left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))\right)^p\right)^{1/p}\leq\left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\mathbf{1}_{[f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k)\leq\theta]}\right)^p\right)^{1/p}=R^p_{S_+,S_-}\mathbf{1}^\theta_f.$$

Thus, with probability at least

$$1-\mathcal{N}\left(\mathcal{F},\frac{\epsilon}{8\text{Lip}(\phi)}\right)\left[2\exp\left[-2\left(\frac{\epsilon}{4}\right)^{2p}K\right]+2\exp\left[-\frac{\epsilon^2}{8}I\right]\right],$$

we have

$$R^p_{\mathcal{D}_+\mathcal{D}_-}\mathbf{1}_f \leq R^p_{S_+,S_-}\mathbf{1}^\theta_f+\epsilon.$$

Thus, the theorem has been proved. □