A Structural SVM Based Approach for Optimizing Partial AUC

Harikrishna Narasimhan HARIKRISHNA@CSA.IISC.ERNET.IN
Shivani Agarwal SHIVANI@CSA.IISC.ERNET.IN
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Abstract

The area under the ROC curve (AUC) is a widely used performance measure in machine learning. Increasingly, however, in several applications, ranging from ranking and biometric screening to medical diagnosis, performance is measured not in terms of the full area under the ROC curve, but instead, in terms of the *partial* area under the ROC curve between two specified false positive rates. In this paper, we develop a structural SVM framework for directly optimizing the partial AUC between any two false positive rates. Our approach makes use of a cutting plane solver along the lines of the structural SVM based approach for optimizing the full AUC developed by Joachims (2005). Unlike the full AUC, where the combinatorial optimization problem needed to find the most violated constraint in the cutting plane solver can be decomposed easily to yield an efficient algorithm, the corresponding optimization problem in the case of partial AUC is harder to decompose. One of our key technical contributions is an efficient algorithm for solving this combinatorial optimization problem that has the same computational complexity as Joachims' algorithm for optimizing the usual AUC. This allows us to efficiently optimize the partial AUC in any desired false positive range. We demonstrate the approach on a variety of real-world tasks.

1. Introduction

The receiver operating characteristic (ROC) curve plays an important role as an evaluation tool in machine learning. In particular, the area under the ROC curve (AUC) is widely used to summarize the performance of a scoring function in binary classification, and is often the primary performance measure of interest in bipartite ranking problems (Cortes & Mohri, 2004; Agarwal et al., 2005). In an increasing number of applications, however, the performance measure of interest is not the area under the full ROC curve, but instead, the *partial* area under the ROC curve between two specified false positive rates (FPRs) (see Figure 1). For example, in ranking applications where accuracy at the top is critical, one is often interested in the left-most part of the ROC curve (Rudin, 2009; Agarwal, 2011; Rakotomamonjy, 2012); this corresponds to maximizing partial AUC in a false positive range of the form $[0, \beta]$. In biometric screening, where false positives are intolerable, one is again interested in maximizing the partial AUC in a false positive range $[0, \beta]$ for some suitably small β. In the KDD Cup 2008 challenge on breast cancer detection, performance was measured in terms of the partial AUC in a specific false positive range $[\alpha, \beta]$ that was deemed clinically relevant (Rao et al., 2008).[1]

Figure 1. Partial AUC in false positive range $[\alpha, \beta]$.

In this paper, we develop a general structural SVM based method for directly optimizing the partial AUC between any two given false positive rates α and β. Our approach makes use of a cutting plane solver along the lines of the structural SVM based approach for optimizing the full AUC developed by (Joachims, 2005).

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

[1] More specifically, the KDD Cup 2008 challenge used the partial area under the *free-response* operating characteristic curve, where a scaled version of usual FPR is used.

Each iteration of the cutting plane method involves a combinatorial search over an exponential number of orderings of the positive vs. negative training instances – with each ordering represented as a binary matrix – to find the currently most violated constraint. In the case of the AUC, this combinatorial optimization problem decomposes neatly into one in which each matrix entry can be chosen independently (Joachims, 2005). Unfortunately, for the partial AUC, such a straightforward decomposition is no longer possible since the negative instances involved in the relevant false positive range can be different for different orderings. We characterize the set of relevant negative instances in the optimal ordering and formulate an equivalent optimization problem with a restricted search space, which can then be broken down into smaller tractable optimization problems. For a false positive range of the form $[0, \beta]$, it turns out that following the reformulation, the individual matrix entries can again be optimized independently. For the general case $[\alpha, \beta]$, the individual entries of the matrix cannot be chosen independently, but each *row* of the matrix can be optimized separately – and efficiently. In both cases, the optimization procedure has the same computational complexity as in the case of Joachims' algorithm for optimizing the usual AUC.

Related Work. There has been much work on developing algorithms to optimize the full AUC, mostly in the context of ranking (Herbrich et al., 2000; Joachims, 2002; Freund et al., 2003; Burges et al., 2005; Joachims, 2005). There has also been interest in the ranking literature in optimizing measures focusing on the left end of the ROC curve, corresponding to maximizing accuracy at the top of the list (Rudin, 2009); in particular, the recent Infinite Push algorithm (Agarwal, 2011; Rakotomamonjy, 2012) can be viewed as maximizing the partial AUC in the range $[0, \frac{1}{n}]$, where n is the number of negative training examples.

The partial AUC in false positive ranges of the form $[0, \beta]$ has received some attention in the bioinformatics and biometrics literature (Pepe & Thompson, 2000; Dodd & Pepe, 2003; Wang & Chang, 2011; Ricamato & Tortorella, 2011; Hsu & Hsueh, 2012); however in most cases, the algorithms developed are heuristic in nature. The asymmetric SVM algorithm of (Wu et al., 2008), an extension of one-class SVM that relies on a parameter tuning procedure, also aims to maximize the partial AUC in a range $[0, \beta]$. Some recent work on optimizing the partial AUC in general false positive ranges of the form $[\alpha, \beta]$ includes the boosting-based algorithms pAUCBoost (Komori & Eguchi, 2010) and pU-AUCBoost (Takenouchi et al., 2012).

To our knowledge, this is the first work to develop a principled support vector method that can directly optimize the partial AUC in an arbitrary false positive range $[\alpha, \beta]$.

Organization. We start with some preliminaries in Section 2. Section 3 describes our structural SVM based approach for optimizing the partial AUC. Section 4 gives efficient algorithms for finding the most violated constraint for the cutting plane solver for both the special case $[0, \beta]$ and the general case $[\alpha, \beta]$. Section 5 gives experimental results on several real-world tasks: ranking applications; protein-protein interaction (PPI) prediction; and medical diagnosis.

2. Preliminaries

Let X be an instance space, and let $\mathcal{D}_+, \mathcal{D}_-$ be probability distributions on X. Given a training sample $S = (S_+, S_-)$ consisting of m positive instances $S_+ = (x_1^+, \ldots, x_m^+) \in X^m$ drawn iid according to $\mathcal{D}_+$ and n negative instances $S_- = (x_1^-, \ldots, x_n^-) \in X^n$ drawn iid according to $\mathcal{D}_-$, the goal is to learn a scoring function $f : X \to \mathbb{R}$ that has good performance in terms of the partial AUC between some specified false positive rates α and β, where $0 \leq \alpha < \beta \leq 1$, as described in more detail below.

Partial AUC. Recall that for a scoring function $f : X \to \mathbb{R}$ and threshold $t \in \mathbb{R}$, the true positive rate (TPR) of the classifier $\text{sign}(f(x) - t)$ is the probability that it correctly classifies a random positive instance from $\mathcal{D}_+$ as positive:[2]

$$\text{TPR}_f(t) = \mathbf{P}_{x^+ \sim \mathcal{D}_+}[f(x^+) > t] \,.$$

Similarly, the false positive rate (FPR) of the classifier is the probability that it misclassifies a random negative instance from $\mathcal{D}_-$ as positive:

$$\text{FPR}_f(t) = \mathbf{P}_{x^- \sim \mathcal{D}_-}[f(x^-) > t] \,,$$

The ROC curve for the scoring function f is then defined as the plot of $\text{TPR}_f(t)$ against $\text{FPR}_f(t)$ for different values of t. The area under this curve can be computed as

$$\text{AUC}_f \;=\; \int_0^1 \text{TPR}_f(\; \text{FPR}_f^{-1}(u) \;)\; du \,,$$

where $\text{FPR}_f^{-1}(u) = \inf\left\{t \in \mathbb{R} \mid \text{FPR}_f(t) \leq u\right\}$. Assuming there are no ties, it can be shown (Cortes & Mohri, 2004) that the AUC can be written as

$$\text{AUC}_f = \mathbf{P}_{(x^+, x^-) \sim \mathcal{D}_+ \times \mathcal{D}_-}[f(x^+) > f(x^-)] \,.$$

[2] If f has ties with non-zero probability, then one needs to add a $\frac{1}{2}\mathbf{P}_{x^+ \sim \mathcal{D}_+}[f(x^+) = t]$ term in the definition.

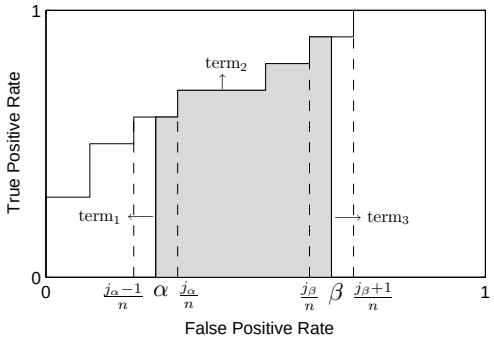

Figure 2. Empirical partial AUC between FPRs α and β.

Our interest here is in the area under the curve between FPRs α and β. The (normalized) partial AUC of f in the range $[\alpha, \beta]$ is defined as

$$\text{pAUC}_f(\alpha,\beta) \;=\; \frac{1}{\beta-\alpha}\int_\alpha^\beta \text{TPR}_f(\,\text{FPR}_f^{-1}(u)\,)\;\; du\,.$$

Empirical Partial AUC. Given a sample $S = (S_+, S_-)$ as above, one can plot an empirical ROC curve corresponding to a scoring function $f : X \to \mathbb{R}$; assuming there are no ties, this is obtained by using

$$\widehat{\text{TPR}}_f(t) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\big(f(x_i^+) > t\big)$$

$$\widehat{\text{FPR}}_f(t) = \frac{1}{n}\sum_{j=1}^{n} \mathbf{1}\big(f(x_j^-) > t\big)$$

instead of $\text{TPR}_f(t)$ and $\text{FPR}_f(t)$. Again assuming there are no ties, the area under this empirical curve is given by

$$\widehat{\text{AUC}}_f = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{1}\big(f(x_i^+) > f(x_j^-)\big)\,.$$

The (normalized) empirical partial AUC of f in the FPR range $[\alpha, \beta]$ can then be written as (Dodd & Pepe, 2003)[3]

$$\widehat{\text{pAUC}}_f(\alpha,\beta) = \frac{1}{mn(\beta-\alpha)}\sum_{i=1}^{m}\Bigg[\big(j_\alpha - n\alpha\big)\cdot\mathbf{1}\big(f(x_i^+) > f(x_{(j_\alpha)}^-)\big) + \sum_{j=j_\alpha+1}^{j_\beta}\mathbf{1}\big(f(x_i^+) > f(x_{(j)}^-)\big) + \big(n\beta - j_\beta\big)\cdot\mathbf{1}\big(f(x_i^+) > f(x_{(j_\beta+1)}^-)\big)\Bigg]\,,$$

[3]Dodd & Pepe (2003) used only the sum over $j_\alpha + 1$ to j_β; we give a more complete definition here.

Table 1. Scores assigned by two scoring functions f_1 and f_2 to a sample containing 4 positive instances and 5 negative instances.

	x_1^+	x_2^+	x_3^+	x_4^+	x_1^-	x_2^-	x_3^-	x_4^-	x_5^-
f_1	9.1	6.8	6.1	5.7	8.5	8.1	4.2	3.6	2.3
f_2	9.9	8.7	3.3	2.1	7.6	5.3	4.9	4.4	0.8

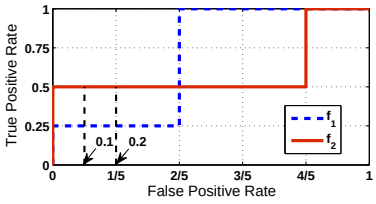

Figure 3. ROC curves for scoring functions f_1, f_2 from Table 1.

where $j_\alpha = \lceil n\alpha \rceil$, $j_\beta = \lfloor n\beta \rfloor$, and $x_{(j)}^-$ denotes the negative instance in S_- ranked in j-th position (among negatives, in descending order of scores) by f (see Figure 2 for a visual explanation of the three terms in the sum). Note that we have assumed $n \geq \frac{1}{\beta-\alpha}$, so that $j_\alpha \leq j_\beta$.[4]

Partial AUC vs. AUC. Before closing this section, we note that a scoring function with a high AUC value need not be optimal in terms of partial AUC in a particular FPR range. This is illustrated in Table 1 and Figure 3, which show scores assigned by two scoring functions f_1 and f_2 on a hypothetical sample of 4 positive and 6 negative instances, and the corresponding ROC curves. As can be seen, while f_1 gives a higher AUC value, f_2 has higher partial AUC in the FPR range $[0.1, 0.2]$. This motivates the need to design algorithms tailored for optimizing partial AUC.

3. Structural SVM Approach for Optimizing Partial AUC

Given a training sample $S = (S_+, S_-) \in X^m \times X^n$, our goal is to find a scoring function $f : X \to \mathbb{R}$ that maximizes the partial AUC in an FPR range $[\alpha, \beta]$, or equivalently, that minimizes the following empirical risk:

$$\widehat{R}_{\text{pAUC}(\alpha,\beta)}(f) \;=\; 1 - \widehat{\text{pAUC}}_f(\alpha,\beta)\,. \qquad (1)$$

We now cast this problem into a structural SVM framework (Tsochantaridis et al., 2005). In the following, we shall assume $X \subseteq \mathbb{R}^d$ for some $d \in \mathbb{Z}_+$ and con-

[4]If $n < \frac{1}{\beta-\alpha}$, the empirical partial AUC in the FPR range $[\alpha, \beta]$ becomes $\frac{1}{mn(\beta-\alpha)}\sum_{i=1}^{m} n(\beta-\alpha)\cdot\mathbf{1}\big(f(x_i^+) > f(x_{(j_\alpha)}^-)\big)$.

sider linear scoring functions of the form $f(x) = w^\top x$ for some $w \in \mathbb{R}^d$; the approach extends to non-linear functions / non-Euclidean instance spaces using kernels (Yu & Joachims, 2008).

For any ordering of the training instances, we can represent (errors in) the relative ordering of the m positive instances in S_+ and n negative instances in S_- via a matrix $\pi = [\pi_{ij}] \in \{0,1\}^{m \times n}$ as follows:

$$\pi_{ij} = \begin{cases} 1 & \text{if } x_i^+ \text{ is ranked } \textit{below } x_j^- \\ 0 & \text{if } x_i^+ \text{ is ranked above } x_j^- . \end{cases}$$

Not all 2^{mn} matrices in $\{0,1\}^{m \times n}$ represent a valid relative ordering (due to transitivity requirements). We let $\Pi_{m,n}$ denote the set of all matrices in $\{0,1\}^{m \times n}$ that do correspond to valid orderings. Clearly, the correct relative ordering π^* has $\pi^*_{ij} = 0 \; \forall i, j$. For any $\pi \in \Pi_{m,n}$, we can define the partial AUC loss of π with respect to π^* as

$$\Delta_{\text{pAUC}(\alpha,\beta)}(\pi^*, \pi) = \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \Bigg[\big(j_\alpha - n\alpha\big) \cdot \pi_{i,(j_\alpha)_\pi} + \sum_{j=j_\alpha+1}^{j_\beta} \pi_{i,(j)_\pi} + \big(n\beta - j_\beta\big) \cdot \pi_{i,(j_\beta+1)_\pi} \Bigg], \quad (2)$$

where $(j)_\pi$ denotes the index of the negative instance in S_- ranked in j-th position (among negatives) by any fixed ordering consistent with the matrix π (note that all such orderings yield the same value of partial AUC loss).

Building on the approach of (Joachims, 2005), we define a joint feature map $\phi : (X^m \times X^n) \times \Pi_{m,n} \to \mathbb{R}^d$ as

$$\phi(S, \pi) = \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=1}^{n} (1 - \pi_{ij})(x_i^+ - x_j^-) .$$

The above expression evaluates to a (normalized) sum of feature vector differences over all pairs of positive-negative instances in S in which the positive instance is ordered by π above the negative instance.[5] This choice of $\phi(S, \pi)$ ensures that for any fixed $w \in \mathbb{R}^d$, maximizing $w^\top \phi(S, \pi)$ over $\pi \in \Pi_{m,n}$ yields an ordering matrix consistent with the scoring function $f(x) = w^\top x$. The problem of optimizing the partial AUC now reduces to finding a $w \in \mathbb{R}^d$ for which the maximizer over $\pi \in \Pi_{m,n}$ of $w^\top \phi(S, \pi)$ has the highest partial AUC (or minimum partial AUC loss). This can be framed as the following convex optimization problem:

$$\min_{w, \xi \geq 0} \; \frac{1}{2}||w||^2 + C\xi$$
$$\text{s.t.} \;\; \forall \pi \in \Pi_{m,n} :$$
$$w^\top \big(\phi(S, \pi^*) - \phi(S, \pi)\big) \;\geq \Delta_{\text{pAUC}(\alpha,\beta)}(\pi^*, \pi) - \xi \,,$$

which in turn can be written as

$$\min_{w, \xi \geq 0} \; \frac{1}{2}||w||^2 + C\xi \quad \text{(OP1)}$$
$$\text{s.t.} \;\; \forall \pi \in \Pi_{m,n} :$$
$$\frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_{ij} \, w^\top (x_i^+ - x_j^-) \;\geq \Delta_{\text{pAUC}(\alpha,\beta)}(\pi^*, \pi) - \xi \,,$$

where $C > 0$ is an appropriate regularization parameter. It can be shown that the slack variable ξ in the above optimization problem evaluates to an upper bound on the empirical partial AUC risk in Eq. (1) (details will be provided in a longer version of the paper). For $\alpha = 0$ and $\beta = 1$, the above optimization problem reduces to that associated with the full AUC considered in (Joachims, 2005). As we shall see, solving the above optimization problem for the partial AUC turns out to be more tricky.

The optimization problem OP1 has an exponential number of constraints, one for each matrix $\pi \in \Pi_{m,n}$. To solve this problem, we use the cutting plane method, which is based on the fact that for any $\epsilon > 0$, a small subset of the constraints is sufficient to find an ϵ-approximate solution to the problem (Joachims, 2006). In particular, the cutting plane method starts with an empty constraint set $\mathcal{C} = \emptyset$, and on each iteration, adds the most violated constraint to $\mathcal{C}$, thereby solving a tighter relaxation of OP1 in the subsequent iteration; this continues until no constraint is violated by more than ϵ (see supplementary material for an outline of this algorithm).

It can be shown that for any fixed regularization parameter $C > 0$ and accuracy parameter $\epsilon > 0$, the cutting plane method converges in a constant number of iterations (Joachims, 2006). Since the quadratic program in each iteration of this algorithm is of constant size, the only bottleneck in the algorithm is the combinatorial optimization over $\Pi_{m,n}$ required to find the most violated constraint. In the following, we show how this combinatorial optimization can be performed

[5] Note that while the definition of partial AUC loss in Eq. (2) involves only a subset of the negative instances in S_- corresponding to the FPR interval $[\alpha, \beta]$, the definition of the feature map $\phi(S, \pi)$ includes all negative instances in S_-. This 'mismatch' between the loss and feature vector is necessary and is handled in the argmax operation in OP2; while one could imagine defining $\phi(S, \pi)$ by summing over only the negative instances that fall in the range $[\alpha, \beta]$ using a ranking consistent with π, this would not yield a unique feature vector since such a ranking is not unique.

efficiently for the partial AUC, yielding an overall time complexity that is polynomial in the size of the training sample, and moreover, that matches the computational complexity for the full AUC.

4. Efficient Algorithms for Finding the Most Violated Constraint

As noted above, obtaining a polynomial-time cutting plane method for optimizing the partial AUC in the structural SVM setting of OP1 hinges on having a polynomial-time algorithm for the combinatorial optimization problem (over $\Pi_{m,n}$) associated with finding the most violated constraint on each iteration, which can be stated explicitly as follows:

$$\bar{\pi} \;=\; \operatorname*{argmax}_{\pi \in \Pi_{m,n}} Q_w(\pi)\,, \tag{OP2}$$

where

$$\begin{aligned} Q_w(\pi) \;&=\; \Delta_{\mathrm{pAUC}(\alpha,\beta)}(\pi^*, \pi) \\ &\quad - \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_{ij}\, w^\top (x_i^+ - x_j^-)\,. \end{aligned} \tag{3}$$

In the case of AUC, the above argmax (over an exponential number of matrices) can be easily computed by neatly decomposing the problem into one where each π_{ij} can be chosen independently (Joachims, 2005). For the case of partial AUC in an arbitrary FPR interval $[\alpha, \beta]$, it is not obvious how such a decomposition is possible as the loss term in the objective $Q_w(\pi)$ now involves different subsets of negative instances for different orderings $\pi \in \Pi_{m,n}$. However it turns out that by working with a restricted search space of orderings in $\Pi_{m,n}$, one can indeed break down OP2 into smaller maximization problems over (groups of) π_{ij}'s, and that the resulting maximization problems can then be solved with the same computational cost as that required for the usual AUC. To proceed, for any $w \in \mathbb{R}^d$, let us define the set

$$\Pi_{m,n}^w = \left\{ \pi \in \Pi_{m,n} \;\middle|\; \forall i, j_1 < j_2 : \pi_{i,(j_1)_w} \geq \pi_{i,(j_2)_w} \right\}.$$

This is the set of all ordering matrices π in which any two negative instances that are separated by a positive instance are sorted according to w (i.e. in descending order of the scores $w^\top x_j^-$). Then we have the following result (see supplementary material for a proof):

Theorem 1. The solution $\bar{\pi}$ to OP2 lies in $\Pi_{m,n}^w$.[6]

[6] We note that a similar observation was made for the optimizer associated with the mean average precision (MAP) objective in (Yue et al., 2007).

This allows us to restrict the search space to $\Pi_{m,n}^w$ and rewrite OP2 as follows:

$$\bar{\pi} \;=\; \operatorname*{argmax}_{\pi \in \Pi_{m,n}^w} Q_w(\pi)\,. \tag{OP3}$$

Now note that for any $\pi \in \Pi_{m,n}$, any ordering of the instances that is consistent with π yields the same value of $Q_w(\pi)$. In particular, for any $\pi \in \Pi_{m,n}^w$, there is an ordering consistent with π in which all negative instances are sorted according to w (this follows from the definition of $\Pi_{m,n}^w$). This gives

$$Q_w(\pi) = \widetilde{Q}_w(\pi) \qquad \forall \pi \in \Pi_{m,n}^w\,,$$

where

$$\begin{aligned} \widetilde{Q}_w(\pi) \;=\; \sum_{i=1}^{m} \Bigg[& - \sum_{j=1}^{j_\alpha - 1} \pi_{i,(j)_w}\, w^\top x^{\pm}_{i,(j)_w} \\ & + \pi_{i,(j_\alpha)_w} \left((j_\alpha - n\alpha) - w^\top x^{\pm}_{i,(j_\alpha)_w} \right) \\ & + \sum_{j=j_\alpha+1}^{j_\beta} \pi_{i,(j)_w} \left(1 - w^\top x^{\pm}_{i,(j)_w} \right) \\ & + \pi_{i,(j_\beta+1)_w} \left((n\beta - j_\beta) - w^\top x^{\pm}_{i,(j_\beta+1)_w} \right) \\ & - \sum_{j=j_\beta+2}^{n} \pi_{i,(j)_w}\, w^\top x^{\pm}_{i,(j)_w} \Bigg], \end{aligned} \tag{4}$$

where $(j)_w$ refers to the index of the negative instance in S_- ranked in j-th position by w, and where we have used the shorthand notation $x_{ij}^{\pm} = (x_i^+ - x_j^-)$. This allows us to rewrite OP3 as follows:

$$\bar{\pi} \;=\; \operatorname*{argmax}_{\pi \in \Pi_{m,n}^w} \widetilde{Q}_w(\pi)\,. \tag{OP4}$$

This optimization problem is now easier to solve as the set of negative instances over which the loss term in the objective is computed is the same for all orderings in the search space. Below we give efficient algorithms for solving this problem both for the special case when the FPR range of interest is of the form $[0, \beta]$, and for the general case $[\alpha, \beta]$.

4.1. Efficient Algorithm for Special Case $[0, \beta]$

For $\alpha = 0$, we have $j_\alpha = \lceil n\alpha \rceil = 0$. In this case, $\widetilde{Q}_w(\pi)$ reduces to the following:

$$\begin{aligned} \widetilde{Q}_w(\pi) \;=\; \sum_{i=1}^{m} \Bigg[& \sum_{j=1}^{j_\beta} \pi_{i,(j)_w} \left(1 - w^\top x^{\pm}_{i,(j)_w} \right) \\ & + \pi_{i,(j_\beta+1)_w} \left((n\beta - j_\beta) - w^\top x^{\pm}_{i,(j_\beta+1)_w} \right) \\ & - \sum_{j=j_\beta+2}^{n} \pi_{i,(j)_w}\, w^\top x^{\pm}_{i,(j)_w} \Bigg]. \end{aligned} \tag{5}$$

We consider solving a relaxation of OP4 in which the above objective is optimized over all $\pi \in \{0,1\}^{m\times n}$:

$$\bar{\pi} \;=\; \operatorname*{argmax}_{\pi\in\{0,1\}^{m\times n}} \widetilde{Q}_w(\pi)\,. \qquad \text{(OP5)}$$

As can be seen from Eq. (5), the objective $\widetilde{Q}_w(\pi)$ decomposes into a sum of terms involving the individual elements $\pi_{i,j}$, and therefore the solution to OP5 is obtained by maximizing $\widetilde{Q}_w(\pi)$ over each element $\pi_{i,j} \in \{0,1\}$ separately, which clearly yields:

$$\bar{\pi}_{i,(j)_w} = \begin{cases} \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le 1) & \text{if } j \in \{1,\ldots,j_\beta\} \\ \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le n\beta - j_\beta) & \text{if } j = j_\beta + 1 \\ \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le 0) & \text{otherwise.} \end{cases} \tag{6}$$

The following result shows that this solution to the relaxed problem OP5 actually lies in $\Pi^w_{m,n}$, and therefore this is also a solution to the desired problem OP4:

Theorem 2. The ordering matrix $\bar{\pi}$ defined by Eq. (6) lies in $\Pi^w_{m,n}$.

Proof. (sketch) Let $i \in [m]$ and $j_1 < j_2$. There are 5 possibilities: (1) $j_1, j_2 \in \{1,\ldots,j_\beta\}$; (2) $j_1, j_2 \in \{j_\beta+2,\ldots,n\}$; (3) $j_1 \in \{1,\ldots,j_\beta\}$ and $j_2 = j_\beta + 1$; (4) $j_1 = j_\beta + 1$ and $j_2 \in \{j_\beta+2,\ldots,n\}$; and (5) $j_1 \in \{1,\ldots,j_\beta\}$ and $j_2 \in \{j_\beta+2,\ldots,n\}$. In each case, it can be verified from Eq. (6) that $\bar{\pi}_{i,(j_1)_w} \ge \bar{\pi}_{i,(j_2)_w}$ (in each case, taking $\bar{\pi}_{i,(j_1)_w} < \bar{\pi}_{i,(j_2)_w}$ gives $w^\top x^-_{(j_1)_w} < w^\top x^-_{(j_2)_w}$, a contradiction). □

A straightforward implementation to compute the solution in Eq. (6) has computational complexity $O(mn + n\log n)$. Using a more compact representation of the orderings, this can be further reduced to $O((m+n)\log(m+n))$ (Joachims, 2005) (details will be provided in a longer version of the paper).

4.2. Efficient Algorithm for General Case $[\alpha, \beta]$

In the general case, where we allow FPR intervals of the form $[\alpha, \beta]$ for $\alpha > 0$, it is no longer sufficient to solve a relaxation of OP4 over all $\pi \in \{0,1\}^{m\times n}$ as the resulting solution no longer lies in $\Pi^w_{m,n}$. Therefore the individual π_{ij}'s can no longer be chosen independently. However, it turns out that each *row* of the matrix, $\pi_i \in \{0,1\}^n$, can still be considered separately, and moreover, that the optimization over each π_i can be done efficiently. In particular, note that for each i, the i-th row of $\bar{\pi}$ essentially corresponds to an interleaving of the lone positive instance x^+_i with the list of negative instances sorted according to $w^\top x^-_j$; thus each $\bar{\pi}_i$ is of the form

$$\bar{\pi}_{i,(j)_w} \;=\; \begin{cases} 1 & \text{if } j \in \{1,\ldots,r_i\} \\ 0 & \text{if } j \in \{r_i+1,\ldots,n\} \end{cases} \tag{7}$$

Algorithm 1 Find Most-Violated Constraint

Inputs: $S = (S_+, S_-)$, α, β, w
For $i = 1,\ldots,m$ **do**
  Optimize over $r_i \in \{0,\ldots,j_\alpha - 1\}$:
  $$\pi^{(1)}_{i,(j)_w} = \begin{cases} \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le 0)\,, & j \in \{1,\ldots,j_\alpha-1\} \\ 0\,, & j \in \{j_\alpha,\ldots,n\} \end{cases}$$
  Optimize over $r_i \in \{j_\alpha\}$:
  $$\pi^{(2)}_{i,(j)_w} = \begin{cases} 1\,, & j \in \{1,\ldots,j_\alpha\} \\ 0\,, & j \in \{j_\alpha+1,\ldots,n\} \end{cases}$$
  Optimize over $r_i \in \{j_\alpha+1,\ldots,n\}$:
  $$\pi^{(3)}_{i,(j)_w} = \begin{cases} 1\,, & j \in \{1,\ldots,j_\alpha+1\} \\ \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le 1)\,, & j \in \{j_\alpha+2,\ldots,j_\beta\} \\ \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le n\beta - j_\beta)\,, & j = j_\beta+1 \\ \mathbf{1}(w^\top x^{\pm}_{i,(j)_w} \le 0)\,, & j \in \{j_\beta+2,\ldots,n\} \end{cases}$$
  $\bar{k} = \operatorname*{argmax}_{k\in\{1,2,3\}} \left\{ \begin{array}{l} \text{term inside sum over } i \text{ in} \\ \text{Eq. (4) evaluated at } \pi^{(k)}_i \end{array} \right\}$
  $\bar{\pi}_i = \pi^{(\bar{k})}_i$
End For
Output: $\bar{\pi}$

for some $r_i \in \{0,1,\ldots,n\}$. In other words, the optimization over $\pi_i \in \{0,1\}^n$ reduces to an optimization over $r_i \in \{0,1,\ldots,n\}$, or equivalently, an optimization over $\pi_i \in R^w_i$, where

$$R^w_i = \left\{\pi_i \in \{0,1\}^n \;\middle|\; \forall j_1 < j_2 : \pi_{i,(j_1)_w} \ge \pi_{i,(j_2)_w}\right\}$$

with $|R^w_i| = n+1$. Clearly, we have $\Pi^w_{m,n} = R^w_1 \times \ldots \times R^w_m$, and therefore we can rewrite OP4 as

$$\bar{\pi} \;=\; \operatorname*{argmax}_{\pi\in R^w_1\times\ldots\times R^w_m} \widetilde{Q}_w(\pi)\,. \qquad \text{(OP6)}$$

Since the objective $\widetilde{Q}_w(\pi)$ (as given by Eq. (4)) decomposes into a sum of terms involving the individual rows π_i, OP6 can be solved by maximizing $\widetilde{Q}_w(\pi)$ over each row $\pi_i \in R^w_i$ separately. In a straightforward implementation of this optimization, for each $i \in \{1,\ldots,m\}$, one would evaluate the term inside the sum over i in Eq. (4) for each of the $n+1$ values of r_i (corresponding to the $n+1$ choices of $\pi_i \in R^w_i$; see Eq. (7)) and select the optimal among these; each such evaluation takes $O(n)$ time, yielding an overall time complexity of $O(mn^2)$. It turns out, however, that one can partition the $n+1$ values of r_i into 3 groups, $\{0,\ldots,j_\alpha-1\}$, $\{j_\alpha\}$, and $\{j_\alpha+1,\ldots,n\}$, such that the optimization over r_i in each of these 3 groups (after the negative instances have been sorted according to w) can be implemented in $O(n)$ time; this yields an overall time complexity of $O(mn + n\log n)$. A description is given in Algorithm 1. Again, using a more compact representation of the orderings, it is possible to further reduce the computational complexity of Algorithm 1 to $O((m+n)\log(m+n))$ (Joachims, 2005) (details will be provided in an extended version).

Thus, for both $[0, \beta]$ and $[\alpha, \beta]$ cases, it is possible to find the most violated constraint for the partial AUC in the same time as that required for the usual AUC.

5. Experiments

This section contains an experimental evaluation of our structural SVM based method for optimizing partial AUC, which we refer to as SVM_{pAUC}, on several real-world tasks: ranking applications; protein-protein interaction (PPI) prediction; and medical diagnosis.[7,8] All experiments involve learning a linear function.

5.1. Partial AUC in $[0, \beta]$ for Ranking Applications

As noted in the introduction, several ranking applications require optimizing performance at the top of the list, which corresponds to optimizing partial AUC in an FPR range of the form $[0, \beta]$. We consider two such applications here.

Cheminformatics. Here one is given examples of chemical compounds that are active or inactive against a therapeutic target, and the goal is to rank new compounds such that active ones appear at the top of the list. We used a virtual screening data set from (Jorissen & Gilson, 2005); this contains 2142 compounds, each represented as a 1021-bit vector using the FP2 molecular fingerprint representation as in (Agarwal et al., 2010). There are 5 sets of 50 active compounds each (active against 5 different targets), and 1892 inactive compounds. For each target, the 50 active compounds are treated as positive, and all others as negative. We considered optimizing the partial AUC in the FPR range $[0, 0.1]$. The results, averaged over the 5 targets and 10 random 10%-90% train-test splits for each target (subject to preserving the proportion of positives) are shown in Table 2.[9] For comparison, we also show results obtained using the SVM_{AUC} algorithm of (Joachims, 2005), as well as three existing algorithms for optimizing partial AUC: asymmetric SVM (ASVM) (Wu et al., 2008), pAUCBoost (Komori & Eguchi, 2010), and a greedy heuristic method due to (Ricamato & Tortorella, 2011).

[7]Code for SVM_{pAUC} (implemented using APIs from (Joachims, 2008) and (Vedaldi, 2008)) is available at `http://clweb.csa.iisc.ernet.in/harikrishna/Papers/SVMpAUC/`.

[8]See supplementary material for details of parameter tuning and data preprocessing.

[9]In all our evaluations, we used a slightly modified definition of partial AUC that allows for ties. A star against a baseline method in the table indicates a statistically significant difference between SVM_{pAUC} and the method (using the two-sided Wilcoxon test) at a 95% confidence level.

Table 2. Results on cheminformatics data set.

	pAUC(0, 0.1)
SVM_{pAUC}[0,0.1]	**65.25**
SVM_{AUC}	62.64 *
ASVM[0,0.1]	63.80
pAUCBoost[0,0.1]	43.89 *
Greedy-Heuristic[0,0.1]	8.33 *

Information retrieval (IR). Another ranking application where accuracy at the top is important is IR. Here one is given a certain number of training queries together with documents labeled as relevant for the query (positive) or irrelevant (negative), and the goal is to rank documents for new queries. We evaluated our algorithm on two widely used IR data sets: TD2004, which is part of the LETOR 2.0 collection (Liu et al., 2007) (75 queries; total of 444 positive documents and 73726 negative documents); and the TREC10 data set (50 queries, 2892 positive documents and 203507 negative documents) used in (Yue et al., 2007). In TD2004, each document is represented using 44 features, while those in TREC10 are represented using 750 features. For each data set, we used random splits containing 60% of the queries for training, 20% for validation and the remaining for testing (subject to preserving the proportion of positives). The results, averaged over 10 such random splits, are shown in Table 3. In this case, the existing algorithms for optimizing partial AUC do not apply easily as they are not designed to handle queries; we include a comparison with SVM_{AUC} as a baseline. Since in IR it is common to focus on just a small number of documents at the top, we show performance in terms of a range of partial AUC values that capture this, as well as the AUC. As can be seen, optimizing partial AUC tends to give higher accuracy close to the very top of the list; where SVM_{AUC} has similar performance, the difference is not statistically significant.

5.2. Partial AUC in $[0, \beta]$ for PPI Prediction

In protein-protein interaction (PPI) prediction, given a pair of proteins, the task is to predict whether they interact or not; here again the partial AUC has been used as an evaluation measure (Qi et al., 2006). We used the PPI data for Yeast from (Qi et al., 2006), which contains 2865 protein pairs known to be interacting (positive) and a random set of 237384 protein pairs assumed to be non-interacting (negative). Each protein pair is represented by 162 features, but there are several missing features; we used a subset of 85 features that contained less than 25% missing values (with missing feature values replaced by mean/mode values). The results, averaged over 10 random 1%-9%-90% train-validation-test splits (subject to preserving

Table 3. Results on information retrieval data sets (here n_{test} = number of negative test examples).

		pAUC($0, \frac{1}{n_{\text{test}}}$)	pAUC($0, \frac{5}{n_{\text{test}}}$)	pAUC($0, 0.1$)	pAUC($0, 1$) = AUC
TD2004	SVM_{pAUC}[0,0.1]	**16.74**	**30.19**	**70.51**	**93.95**
	SVM_{AUC}	13.62 *	23.39 *	67.60	93.38
TREC10	SVM_{pAUC}[0,0.1]	**4.58**	**7.47**	49.07	89.23
	SVM_{AUC}	3.85 *	7.16	**50.69**	**89.51**

Table 4. Results on PPI data set.

	pAUC($0, 0.1$)
SVM_{pAUC}[0,0.1]	**51.79**
SVM_{AUC}	39.72 *
ASVM[0,0.1]	44.51 *
pAUCBoost[0,0.1]	48.65 *
Greedy-Heuristic[0,0.1]	47.33 *

Table 5. Results on KDD Cup 08 data set.

	pAUC($0.2s, 0.3s$)
SVM_{pAUC}[$0.2s, 0.3s$]	**51.44**
SVM_{AUC}	50.50
pAUCBoost[$0.2s, 0.3s$]	48.06 *
Greedy-Heuristic[$0.2s, 0.3s$]	46.99 *

the proportion of positives), are shown in Table 4. Here SVM_{pAUC} significantly outperforms all the four baseline algorithms.

5.3. Partial AUC in $[\alpha, \beta]$ for Medical Diagnosis

Our final evaluation is on the KDD Cup 2008 challenge on early breast cancer detection (Rao et al., 2008). Here the task is to predict whether a given region of interest (ROI) from a breast X-ray image is malignant (positive) or benign (negative). The data set is collected from 118 malignant patients and 1594 normal patients. Four X-ray images are available for each patient; overall, there are 102294 candidate ROIs selected from these X-ray images, with each ROI represented by 117 features. In the KDD Cup challenge, performance was evaluated in terms of the partial area under the *free-response* operating characteristic (FROC) curve in a false positive range $[0.2, 0.3]$ deemed clinically relevant based on radiologist surveys. The FROC curve (Miller, 1969) effectively uses a scaled version of the false positive rate; for our purposes, the corresponding false positive rate is obtained by re-scaling by a factor of $s = 6848/101671$ (this is the total number of images divided by the total number of negative ROIs). Thus, the goal in our experiments was to maximize the partial AUC in the clinically relevant FPR range $[0.2s, 0.3s]$. The results, averaged over 10 random 5%-95% train-test splits (subject to preserving the proportion of positives) are shown in Table 5. Baselines here include SVM_{AUC}, pAUCBoost which can optimize partial AUC over FPR ranges $[\alpha, \beta]$, and an extension of the greedy heuristic method in (Ricamato & Tortorella, 2011) to handle arbitrary FPR ranges.

5.4. Run Time Analysis

Finally, we analyzed the training time of the proposed SVM_{pAUC} method for different FPR intervals. We used the TREC10 data set for these experiments, focusing on FPR intervals of the form $[0, \beta]$ for different values of β. Figure 4 shows (a) the average CPU time taken by the routine for finding the most violated constraint (MVC) for different values of β, and (b) the average number of calls to this routine. As can be seen, the average time taken to find the most violated constraint is similar for all values of β, demonstrating as shown in Section 4 that the time complexity of this procedure for partial AUC is the same as that for full AUC ($\beta = 1$); on the other hand, the average number of calls to this procedure increases as β decreases, i.e. as the FPR interval becomes smaller.

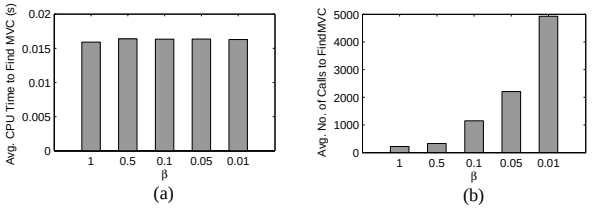

Figure 4. Timing statistics for $\text{SVM}_{\text{pAUC}}[0, \beta]$ on TREC10 data set.

6. Conclusion

The partial AUC is increasingly used as a performance measure in several machine learning applications. We have developed a structural SVM based method, termed SVM_{pAUC}, for optimizing the partial AUC between any two given false positive rates with similar computational complexity as that required for optimizing the usual AUC. Our empirical evaluations on several real-world tasks indicate the proposed method indeed optimizes partial AUC in the desired false positive range, performing comparable to or better than existing baseline techniques.

Acknowledgments. Thanks to the anonymous reviewers for helpful comments. HN thanks Microsoft Research India for a partial travel grant to attend the conference. This work is supported in part by a Ramanujan Fellowship from DST to SA.

References

Agarwal, S. The Infinite Push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *Proceedings of the SIAM International Conference on Data Mining*, 2011.

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

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

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

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

Dodd, L. E. and Pepe, M. S. Partial AUC estimation and regression. *Biometrics*, 59(3):614–623, 2003.

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

Herbrich, R., Graepel, T., and Obermayer, K. Large margin rank boundaries for ordinal regression. In Smola, A., Bartlett, P., Schoelkopf, B., and Schuurmans, D. (eds.), *Advances in Large Margin Classifiers*, pp. 115–132. MIT Press, 2000.

Hsu, M.-J. and Hsueh, H.-M. The linear combinations of biomarkers which maximize the partial area under the ROC curve. *Computational Statistics*, pp. 1–20, 2012.

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

Joachims, T. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Joachims, T. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

Joachims, T. SVM$^{\text{struct}}$ support vector machine for complex outputs, 2008. URL `http://svmlight.joachims.org/svm_struct.html`.

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

Komori, O. and Eguchi, S. A boosting method for maximizing the partial area under the ROC curve. *BMC Bioinformatics*, 11:314, 2010.

Liu, T-Y., Xu, J., Qin, T., Xiong, W., and Li, H. Letor: benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

Miller, H. The FROC curve: A representation of the observer's performance for the method of free response. *Journal of the Acoustical Society of America*, 46(6B): 1473–1476, 1969.

Pepe, M. S. and Thompson, M. L. Combining diagnostic test results to increase accuracy. *Biostatistics*, 1(2):123–140, 2000.

Qi, Y., Bar-joseph, Z., and Klein-seetharaman, J. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins*, 63:490–500, 2006.

Rakotomamonjy, A. Sparse support vector infinite push. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Rao, R. B., Yakhnenko, O., and Krishnapuram, B. KDD Cup 2008 and the workshop on mining medical data. *SIGKDD Explorations Newsletter*, 10(2):34–38, 2008.

Ricamato, M. T. and Tortorella, F. Partial AUC maximization in a linear combination of dichotomizers. *Pattern Recognition*, 44(10-11):2669–2677, 2011.

Rudin, C. 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.

Takenouchi, T., Komori, O., and Eguchi, S. An extension of the receiver operating characteristic curve and AUC-optimal classification. *Neural Computation*, 24 (10):2789–2824, 2012.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Vedaldi, A. A matlab wrapper of SVM$^{\text{struct}}$, 2008. URL `http://www.vlfeat.org/~vedaldi/code/svm-struct-matlab.html`.

Wang, Z. and Chang, Y.-C.I. Marker selection via maximizing the partial area under the ROC curve of linear risk scores. *Biostatistics*, 12(2):369–385, 2011.

Wu, S.-H., Lin, K.-P., Chen, C.-M., and Chen, M.-S. Asymmetric support vector machines: low false-positive learning under the user tolerance. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

Yu, C. J. and Joachims, T. Training structural svms with kernels using sampled cuts. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 794–802, 2008.

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