ABSTRACT

The area under the ROC curve (AUC) is a well known performance measure in machine learning and data mining. In an increasing number of applications, however, ranging from ranking applications to a variety of important bioinformatics applications, performance is measured in terms of the partial area under the ROC curve between two specified false positive rates. In recent work, we proposed a structural SVM based approach for optimizing this performance measure (Narasimhan and Agarwal, 2013). In this paper, we develop a new support vector method, SVM\textsubscript{pAUC\_tight}, that optimizes a tighter convex upper bound on the partial AUC loss, which leads to both improved accuracy and reduced computational complexity. In particular, by rewriting the empirical partial AUC risk as a maximum over subsets of negative instances, we derive a new formulation, where a modified form of the earlier optimization objective is evaluated on each of these subsets, leading to a tighter hinge relaxation on the partial AUC loss. As with our previous method, the resulting optimization problem can be solved using a cutting-plane algorithm, but the new method has better run time guarantees. We also discuss a projected subgradient method for solving this problem, which offers additional computational savings in certain settings. We demonstrate on a wide variety of bioinformatics tasks, ranging from protein-protein interaction prediction to drug discovery tasks, that the proposed method does, in many cases, perform significantly better on the partial AUC measure than the previous structural SVM approach. In addition, we also develop extensions of our method to learn sparse and group sparse models, often of interest in biological applications.

Categories and Subject Descriptors
I.2.6 [Artificial Intelligence]: Learning

Keywords
Partial AUC, SVM, Cutting-Plane Method, ROC Curve

Figure 1: Partial AUC between false positive rates $\alpha$ and $\beta$.

1. INTRODUCTION

The receiver operating characteristic (ROC) curve plays an important role as an evaluation tool in machine learning and data mining. 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 also used as a performance measure in bipartite ranking [8, 3]. 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 (Figure 1). For example, in ranking applications where accuracy at the top is critical, good performance in the left-most part of the ROC curve [25, 1, 22] is warranted; in several important bioinformatics applications such as protein-protein interaction prediction where data is often imbalanced, the partial AUC up to a low false positive rate is preferred over standard classification accuracy [21]; in medical diagnosis applications such as those involving biomarker selection, one is often interested in good performance in a clinically relevant portion of the ROC curve rather than in the entire ROC curve [20, 30, 23].

In recent work, we proposed a structural SVM based approach, which we shall refer to here as SVM\textsubscript{pAUC\_struct}, for optimizing the partial AUC performance measure [18]. This method builds on Joachims’ approach for optimizing the full AUC [13], where the resulting optimization problem is solved using an efficient cutting-plane solver. In this paper, we develop a new support vector method, SVM\textsubscript{pAUC\_tight}, that optimizes a tighter convex upper bound on the partial AUC loss, which leads to both improved accuracy and reduced computational complexity. In particular, by rewriting the partial AUC loss as a maximum of a certain quantity over subsets of negative instances, we derive a new formulation, where a truncated form of the earlier optimization objective is eval-
uted on each of these subsets, leading to a tighter hinge relaxation on the partial AUC loss. As with our previous method, the resulting optimization problem can be solved using a cutting-plane solver, but the new method has better run time guarantees. We also discuss a (primal) projected subgradient descent solver for the new problem, which offers additional computational savings in certain settings.

We evaluate our new method on a variety of bioinformatics tasks where the partial AUC measure is of interest, ranging from protein-protein interaction prediction to drug discovery tasks, and find that in most cases, the new method gives significant improvement in partial AUC performance over the previous structural SVM approach. We also develop extensions of our method to learn sparse and group sparse models, often of interest in biological applications, and demonstrate their efficacy on real-world data.

Related Work. The problem of developing methods that optimize the partial AUC measure has received much attention from the bioinformatics and biometrics communities [20, 9, 17, 30, 24], and also has been addressed to a lesser extent in the machine learning and data mining communities [32, 1, 22, 27]. Many of the existing methods however are either heuristic in nature or focus on specialized cases of this problem. The recently developed structural SVM approach SVM$p_{\text{struct}}$ [18] is a general approach that can be used to optimize the partial AUC between any two given false positive rates, and on several real-world data sets, was found to outperform many of the other existing approaches for this problem; it will therefore serve as a baseline here.

Organization. We give preliminaries together with a brief background on SVM$p_{\text{struct}}$ in Section 2. Section 3 characterizes the upper bound on partial AUC loss optimized by SVM$p_{\text{struct}}$, and motivates the development of our new method. Section 4 describes our new formulation SVM$p_{\text{pAUC}}$ that optimizes a tighter convex upper bound on the partial AUC loss. Section 5 gives efficient solvers for the resulting optimization problem. Section 6 discusses sparse and group sparse extensions of the proposed method. Section 7 gives experimental results on a variety of bioinformatics tasks.

2. PRELIMINARIES AND BACKGROUND

2.1 Problem Setup

Let $X$ be an instance space and $\mathcal{D}_+, \mathcal{D}_-$ 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 $f : X \rightarrow \mathbb{R}$ that has good performance in terms of the partial AUC between any two specified false positive rates $\alpha$ and $\beta$, where $0 \leq \alpha < \beta \leq 1$.

Partial AUC. Recall that for a scoring function $f : X \rightarrow \mathbb{R}$ 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:

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

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

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

The ROC curve for the scoring function $f$ is then defined as the plot of TPR$_f(t)$ against 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 $\text{FPR}_f^{-1}(u) = \inf\{t \in \mathbb{R} \mid \text{FPR}_f(t) \leq u\}$. Assuming there are no ties, the AUC can be written as [8]

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

Our interest here is in the area under the curve between FPRs $\alpha$ and $\beta$. The (normalized) partial AUC (pAUC) 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 $f : X \rightarrow \mathbb{R}$; assuming there are no ties, this is obtained by using

$$\hat{\text{TPR}}_f(t) = \frac{1}{m} \sum_{i=1}^m 1(f(x^+_i) > t);$$

$$\hat{\text{FPR}}_f(t) = \frac{1}{n} \sum_{j=1}^n 1(f(x^-_j) > t)$$

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

$$\hat{\text{AUC}}_f = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n 1(f(x^+_i) > f(x^-_j)).$$

For simplicity of exposition, we assume $n\alpha$ and $n\beta$ are integers; in this case, the (normalized) empirical partial AUC (pAUC) of $f$ in the FPR range $[\alpha, \beta]$ can be written as [9]

$$\hat{\text{pAUC}}_f(\alpha, \beta) = \sum_{i=1}^m \sum_{j=\alpha+1}^{\beta} 1(f(x^+_i) > f(x^-_{(j)})),$$

where $(j)_j$ denotes the index of the negative instance in $S_-$ ranked in $j$-th position (among negatives, in descending order of scores) by $f$.

2.2 Quick Review of SVM$\text{struct}^{\beta}_{\text{pAUC}}$ [18]

Given a training sample $S = (S_+, S_-) \times X^m \times X^n$, the SVM$\text{struct}^{\beta}_{\text{pAUC}}$ algorithm [18] aims to find a scoring function $f : X \rightarrow \mathbb{R}$ that approximately maximizes the empirical pAUC in an FPR range $[\alpha, \beta]$, or equivalently, that minimizes the following empirical pAUC risk:

$$\hat{R}(f) = 1 - \hat{\text{pAUC}}_f(\alpha, \beta)$$

$$= \frac{1}{2} \sum_{i=1}^m \sum_{j=\alpha+1}^{\beta} 1(f(x^+_i) < f(x^-_{(j)})).$$

See [18] for a slightly more general definition when $n\alpha$ and/or $n\beta$ are not integers. We note that all our results extend easily to the more general setting; moreover, all our experiments use the general formulation.
We briefly review the algorithm below. In the following, we assume $X \subseteq \mathbb{R}^d$ for some $d \in \mathbb{Z}_+$ and consider linear scoring functions of the form $f(x) = w^T x$ for some $w \in \mathbb{R}^{d,3}$.

For any ordering of the training instances, we 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:

$$
\pi_{ij} = \begin{cases} 
1 & \text{if } x^+_i \text{ is ranked below } x^-_j, \\
0 & \text{if } x^+_i \text{ is ranked above } x^-_j.
\end{cases}
$$

Let $\Pi_{m,n}$ denote the set of all matrices in $\{0,1\}^{m \times n}$ that correspond to valid orderings (satisfying anti-symmetry and transitivity requirements). Clearly, the correct relative ordering $\pi^*$ has $\pi^*_{ij} = 0 \ \forall i, j$. For any $\pi \in \Pi_{m,n}$, we can define the pAUC loss of $\pi$ with respect to $\pi^*$ as the empirical pAUC risk of an ordering consistent with $\pi^*$:

$$
\Delta(\pi^*, \pi) = \frac{1}{mn(\beta - \alpha)} \sum_{i=1}^m \sum_{j=1}^n \pi_{i,(j)\pi},
$$

(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 $\pi$.

Next, we define a joint feature map $\phi : (X^m \times X^n) \times \Pi_{m,n} \rightarrow \mathbb{R}^d$ of the form

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

(3)

and reduce the problem of optimizing the partial AUC to the following convex optimization problem:

$$
\min_{w, \xi \geq 0} \frac{1}{2} ||w||^2 + C\xi \\
\text{s.t. } \forall \pi \in \Pi_{m,n} : \\
w^T (\phi(S, \pi^*) - \phi(S, \pi)) \geq \Delta(\pi^*, \pi) - \xi,
$$

(4)

where $C > 0$ is an appropriate regularization parameter. Note that the above quadratic program has an exponential number of constraints, one for each ordering matrix $\pi \in \Pi_{m,n}$. In [18], we describe a cutting-plane algorithm for solving this problem, which for a fixed regularization parameter $C > 0$ and a tolerance parameter $\epsilon > 0$ runs in time polynomial in the size of the training set.

3. CHARACTERIZATION OF THE UPPER BOUND OPTIMIZED BY SVM\text{pAUC}

The structural SVM optimization problem described in the previous section (OP1) essentially amounts to minimizing a (regularized) convex upper bound on the empirical pAUC risk in Eq. (1) (see [13] for details). We show here that this upper bound can be characterized in terms of a summation of certain hinge loss terms over individual pairs of positive and negative training instances; this helps us draw insights for deriving a new formulation that minimizes a tighter upper bound on the empirical pAUC risk.

For any $\gamma \geq 0$, denote the pairwise $\gamma$-margin hinge loss of $w \in \mathbb{R}^d$ on the instance pair $(x^+_i, x^-_j)$ as

$$
\ell^{(1)}_{ij}(w) = (\gamma - w^T (x^+_i - x^-_j))^+,
$$

(4)

where $u_+ = \max(u, 0)$. Then for the special case of FPR intervals $[0, \beta]$, we have the following result:

**Theorem 1.** Let $\alpha = 0$ and $0 < \beta \leq 1$. Then for any $w \in \mathbb{R}^d$, the smallest $\xi \geq 0$ satisfying the constraints of OP1 evaluates to

$$
\xi = \xi_{\text{pAUC}} + \xi_{\text{extra}},
$$

where

$$
\xi_{\text{pAUC}} = \frac{1}{mn(\beta - \alpha)} \sum_{i=1}^m \sum_{j=1}^n \ell^{(1)}_{i,(j)\omega}(w),
$$

$$
\xi_{\text{extra}} = \frac{1}{mn(\beta - \alpha)} \sum_{i=1}^m \sum_{j=j+1}^n \ell^{(0)}_{i,(j)\omega}(w).
$$

We omit the proof here due to space constraints; please see [19] for details. In the above theorem, the slack variable $\xi$ (error term in OP1) is decomposed into a sum of two terms: the first term $\xi_{\text{pAUC}}$ is an upper bound on the empirical pAUC risk in Eq. (1) (with each 0-1 indicator term upper bounded by a pair-wise hinge loss term); the second term $\xi_{\text{extra}}$ is an additional non-negative term. We can show a similar result for FPR intervals $[\alpha, \beta]$ with $\alpha > 0$:

**Theorem 2.** Let $0 < \alpha < \beta \leq 1$. Then for any $w \in \mathbb{R}^d$, the smallest $\xi \geq 0$ satisfying the constraints of OP1 can be characterized as follows:

$$
\xi = \xi_{\text{pAUC}} + \xi_{\text{extra}},
$$

where

$$
\xi_{\text{pAUC}} \geq \frac{1}{mn(\beta - \alpha)} \sum_{i=1}^m \sum_{j=1}^n \sum_{j=ja+1} \ell^{(1)}_{i,(j)\omega}(w);
$$

$$
\xi_{\text{extra}} \leq \frac{1}{mn(\beta - \alpha)} \sum_{i=1}^m \sum_{j=ja+1} \sum_{j=ja+1} \ell^{(0)}_{i,(j)\omega}(w).
$$

See [19] for the proof. Again, one can see that the slack variable $\xi$ is the sum of a term upper bounding the empirical pAUC risk and an additional non-negative term.

**Insights for a Better Formulation.** In both the above cases, the presence of an additional non-negative term in the upper bound on the pAUC risk optimized by OP1 is due to the mismatch in structure between the loss term $\Delta$ (see Eq. (2)) and the joint-feature map $\phi$ (see Eq. (3)) terms occurring in the constraints of OP1; while the former is computed over only the negative instances ranked in positions $(ja + 1, \ldots, j)$ (among all negative instances) by $\pi$, the latter is computed over all the negative instances. In order to obtain a formulation with a tighter upper bound on the pAUC risk, we redefine $\Delta$ and $\phi$ so as to reduce this mismatch, thus yielding an upper bound on the pAUC risk without the additional negative term.

4. NEW FORMULATION: SVM\text{tight}\text{pAUC}

We now derive a new SVM formulation for optimizing the partial AUC that allows us to get rid of the $\xi_{\text{extra}}$ terms in Theorems 1 and 2, thus yielding a tighter upper bound on the empirical pAUC risk in Eq. (1). The key idea is to
Rewrite the pAUC risk as a maximum of a certain quantity over appropriate subsets of negative instances, and to formulate an optimization problem in which an optimization objective with a smaller value is evaluated on each subset.

**Rewriting \( \hat{R} \).** Let \( \mathcal{Z}_\beta \equiv \left\{ z_{j\beta} \right\} \) denote the set of all subsets of negative training instances of size \( j_\beta \). Let us first consider the special case when \( \alpha = 0 \). In this case, the pAUC risk for a score function \( f \) is given by

\[
\hat{R}(f) = \sum_{j=j_\beta}^{j_m} \sum_{i=1}^{m} \mathbb{1}\left(f(x_i^+) < f(x_{(j_\beta)}^-)\right).
\]

This can be rewritten as

\[
\hat{R}(f) = \max_{z \in \mathcal{Z}_\beta} \sum_{j=j_\beta}^{j_m} \sum_{i=1}^{m} \mathbb{1}\left(f(x_i^+) < f(x_{(j_\beta)}^-)\right),
\]

which can be viewed as the maximum value of \((1 - \hat{AUC}_f)\) attainable on subsets of negative instances of size \( j_\beta \). To see that the expressions in Eq. (5) and Eq. (6) are equivalent, note that the maximum value of \((1 - \hat{AUC}_f)\) over all subsets \( z \) in Eq. (6) is attained for the subset of negative instances ranked in the top \( j_\beta \) positions (among all negative instances in \( S_- \), in descending order of scores) by \( f \).

To obtain a similar rewriting for the general case of FPR intervals \([\alpha, \beta]\), let us first define a form of the general pAUC risk restricted to a subset of negative instances \( z \):

\[
\hat{R}_z(f) = \sum_{j=j_\beta}^{j_m} \sum_{i=1}^{m} \mathbb{1}\left(f(x_i^+) < f(x_{(j_\beta)}^-)\right),
\]

where \( (j_\beta)_z \) denotes the index of the negative instance in the set \( z \) ranked in \( j \)-th position among negative instances in \( z \) by \( f \). Then the pAUC risk can be rewritten as

\[
\hat{R}(f) = \max_{z \in \mathcal{Z}_\beta} \hat{R}_z(f).
\]

In particular, we can show the following:

**Theorem 3.** The maximum in Eq. (7) is attained for the subset of negative instances \( z^* \) ranked in the top \( j_\beta \) positions (among all negative instances in \( S_- \), in descending order of scores) by \( f \).

A proof sketch can be found in [19]. Given this, it is easy to see that the expression in Eq. (7) is equivalent to that in the definition of pAUC risk in Eq. (1).

**New Formulation.** Based on the expression for the pAUC risk in Eq. (7), we now derive a new SVM formulation for optimizing the partial AUC that yields a tighter upper bound on the pAUC loss. In the new formulation, the restricted pAUC risk \( \hat{R}_z \) evaluated on a subset of negative instances \( z \) in Eq. (7) is upper bounded by a restricted form of the earlier optimization objective in OP1, as seen next.

As before, consider linear scoring functions of the form \( f(x) = w^\top x \) for some \( w \in \mathbb{R}^d \). For a given subset of negative instances \( z = \{x_{k_1}, \ldots, x_{k_j}\} \in \mathcal{Z}_\beta \) of size \( j_\beta \), we define the joint feature map \( \phi_z : (X^m \times X^n) \times \Pi_{m,j_\beta} \rightarrow \mathbb{R}^d \) restricted to \( z \), which takes in as input a set of \( m \) positive and \( n \) negative training instances and an ordering matrix of dimension \( m \times j_\beta \) and outputs a vector in \( \mathbb{R}^d \), as follows:

\[
\phi_z(S,\pi) = \frac{1}{mn(\beta - \alpha)} \sum_{j=1}^{m} \sum_{i=1}^{n} (1 - \pi_{ij})(x_i^+ - x_{k_j}^-).
\]

Similarly, for any \( \pi \in \Pi_{m,j_\beta} \), define the modified loss function \( \Delta_\beta \) with respect to \( \pi^* = \{0^{m \times j_\beta}\} \) as

\[
\Delta_\beta(\pi^*,\pi) = \frac{1}{mn(\beta - \alpha)} \sum_{j=1}^{j_m} \sum_{j=j_\beta+1}^{j_m} \pi_{ij} \cdot x_i^+ - x_{k_j}^-.
\]

Then our new formulation \( \text{SVM}_{\text{pAUC}} \) consists of solving the following convex optimization problem:

\[
\min_{w,\xi} \frac{1}{2} ||w||^2 + C \xi \\
\text{s.t.} \quad \forall z \in \mathcal{Z}_\beta, \quad \pi \in \Pi_{m,j_\beta} : \\
w^\top (\phi_z(S,\pi^*) - \phi_z(S,\pi)) \geq \Delta_\beta(\pi^*,\pi) - \xi,
\]

where \( C > 0 \) as before is a regularization parameter. As with the earlier structural SVM formulation, OP2 is a quadratic program with an exponential number of constraints. We describe efficient solvers for this problem in Section 5.

**Upper Bound on \( \hat{R} \) Optimized by SVM\(_{\text{pAUC}}\).** We can show that the new SVM formulation in OP2 optimizes a tighter upper bound on the pAUC risk than the previous structural SVM formulation in OP1. In particular, we have the following characterization of the upper bound optimized by OP2:

**Theorem 4.** Let \( 0 \leq \alpha < \beta \leq 1 \). Then for any \( w \in \mathbb{R}^d \), the smallest \( \xi \geq 0 \) satisfying the constraints of OP2 can be characterized as follows:

\[
\xi = \xi_{\text{pAUC}}
\]

where \( \xi_{\text{pAUC}} \) is as in Theorems 1 and 2.

The proof can be found in [19]. Note that the error term in the new formulation does not have the additional non-negative term \( \xi_{\text{extra}} \) that was present with the error term in the previous formulation (see Theorems 1 and 2), thus resulting in a tighter upper bound on the pAUC risk.

### 5. OPTIMIZATION METHODS FOR SVM\(_{\text{pAUC}}\)

In this section, we describe two optimization techniques for solving OP2, namely, a cutting-plane algorithm that has better run time guarantees than the cutting-plane solver of SVM\(_{\text{strict}}\) and a (primal) projected subgradient method.

#### 5.1 Cutting-Plane Method

The optimization problem OP2 has an exponential number of constraints, one for each set \( z \in \mathcal{Z}_\beta \) and matrix \( \pi \in \Pi_{m,j_\beta} \). One approach to solving this problem is through a cutting-plane method [14], which starts with an empty constraint set \( C = \emptyset \), and on each iteration, adds the most violated constraint to \( C \), thereby solving a tighter relaxation of OP1 in the subsequent iteration; the algorithm stops when no constraint is violated by more than \( \epsilon \) (see Algorithm 1).

It can be shown that for any fixed regularization parameter \( C > 0 \) and tolerance parameter \( \epsilon > 0 \), Algorithm 1 converges in a constant number of iterations [14]. Since the quadratic program in each iteration (line 5) is of constant size, the only bottleneck in the algorithm is the combinatorial optimization over \( \mathcal{Z}_\beta \times \Pi_{m,j_\beta} \) required to find the most-violated constraint (line 6).
of iterations needed for Algorithm 1 to converge is at most $R$.

Algorithm 1 Cutting-Plane Method for SVM\textsuperscript{tight\_pAUC}

1: Inputs: $S = (S_+, S_-), \alpha, \beta, C, \epsilon$
2: Initialize: $C = \emptyset$
3: $H(S, z, \pi; w) \equiv \Delta_{\beta}(\pi^*, \pi) - w^T(\phi_z(S, \pi^*) - \phi_z(S, \pi))$
4: Repeat
5: \hspace{1em} $(w, \xi) = \arg\min_{w, \xi \geq 0} \frac{1}{2}||w||^2 + \xi C$
\hspace{2em} s.t. $\forall (z, \pi) \in C : \xi \geq H(S, z, \pi; w)$
6: \hspace{1em} $(\tilde{\xi}, \tilde{\pi}) = \arg\max_{z \in Z, \pi \in \Pi_{m,j}} H(S, z, \pi; w)$
\hspace{2em} (compute the most-violated constraint)
7: \hspace{1em} $C = C \cup \{(\tilde{z}, \tilde{\pi})\}$
8: Until $H(S, \tilde{z}, \tilde{\pi}; w) \leq \xi + \epsilon$
9: Output: $w$

Algorithm 2 SVM\textsuperscript{tight\_pAUC}: Find Most-Violated Constraint

1: Inputs: $S = (S_+, S_-), \alpha, \beta, w$
2: Set $\tilde{z}$ to the set of negative instances in the top $j_3$ positions in the ranking of negative instances in $S_-$ (in descending order of scores) by $w^T x$ (see [19]). Hence, finding $\tilde{z}$ simply involves sorting the negative instances in $S_-$ according to $w^T x$. Having fixed $\tilde{z}$, finding the ordering matrix $\tilde{\pi}$ then reduces to finding the most-violated constraint in SVM\textsuperscript{struct\_pAUC} [18], but with a smaller set of instances $(S_+, \tilde{z})$ (see Algorithm 2).
3: Obtain $\tilde{\pi}$ by applying the procedure for finding the most-violated constraint in SVM\textsuperscript{struct\_pAUC} on $(S_+, \tilde{z})$ (see [18])
4: Output: $(\tilde{\xi}, \tilde{\pi})$

Finding Most-Violated Constraint. It can be shown that in the solution $(\tilde{z}, \tilde{\pi})$ to the combinatorial optimization problem in Algorithm 1 (line 6), the set $\tilde{z}$ contains the top $j_3$ negative instances ranked (among all negative instances) by $w^T x$ (see [19]). Hence, finding $\tilde{z}$ simply involves sorting the negative instances in $S_-$ according to $w^T x$. Having fixed $\tilde{z}$, finding the ordering matrix $\tilde{\pi}$ then reduces to finding the most-violated constraint in SVM\textsuperscript{struct\_pAUC} [18], but with a smaller set of instances $(S_+, \tilde{z})$.

Time Complexity. A naive implementation of this procedure would take $O(n \log n + m j_3 + (m+n)d)$. However, by using a more compact representation of the orderings [13], the time complexity can be reduced to $O(n \log n + (m + j_3) \log (m+j_3) + (m+n)d)$, which is clearly lower than the $O((m+n)\log(m+n) + (m+n)d)$ time required to find the most-violated constraint in SVM\textsuperscript{struct\_pAUC}; moreover, unlike in SVM\textsuperscript{struct\_pAUC}, the time complexity decreases with $\beta$.

Convergence. It can be shown from [15] that the number of iterations needed for Algorithm 1 to converge is at most

$$\left\lfloor \log_2 \left( \frac{1}{2R_{\text{light}}^{2C}} \right) \right\rfloor + \left\lceil \frac{16 R_{\text{light}}^{2C}}{\epsilon} \right\rceil,$$

where $R_{\text{light}} = \frac{\beta}{2} \max_{i,j} ||x_i^+ - x_j^-||_2$; see [19] for details. This is a stronger guarantee than the one for SVM\textsuperscript{struct\_pAUC}, where the number of iterations required by the cutting-plane procedure to converge is at most

$$\left\lfloor \log_2 \left( \frac{1}{2R_{\text{struct}}^{2C}} \right) \right\rfloor + \left\lceil \frac{16 R_{\text{struct}}^{2C}}{\epsilon} \right\rceil,$$

where $R_{\text{struct}} = \frac{\beta}{4} \max_{i,j} ||x_i^+ - x_j^-||_2$; this gives $R_{\text{struct}} = \frac{1}{2} R_{\text{light}} \geq R_{\text{light}}$. The larger value of $R_{\text{struct}}$ than $R_{\text{light}}$ is due to the fact that the joint feature map $\phi$ in SVM\textsuperscript{struct\_pAUC} is defined over the entire set of negative instances, whereas the joint feature map $\phi_z$ in SVM\textsuperscript{struct\_pAUC} is defined over only a subset of negative instances $z$ of size $j_3$.

5.2 Primal Projected Sub-gradient Method

We now describe a projected sub-gradient method for solving an equivalent reformulation of OP2 (along the lines of

Algorithm 3 Projected Subgradient Method for SVM\textsuperscript{tight\_pAUC}

1: Inputs: $S = (S_+, S_-), \alpha, \beta, C, \eta_0, t_{\text{max}}$
2: Initialize: $w_0 = \text{Initial solution in W}$
3: $H(S, z, \pi; w) = \Delta_{\beta}(\pi^*, \pi) - w^T(\phi_z(S, \pi^*) - \phi_z(S, \pi))$
4: For $t = 1$ to $t_{\text{max}}$
5: \hspace{1em} $(\xi, \pi) = \arg\max_{z \in Z, \pi \in \Pi_{m,j}} H(S, z, \pi; w_t)$
6: \hspace{1em} $\nabla Q_w(w_t) = \phi_z(S, \pi^*) - \phi_z(S, \pi)$
7: \hspace{1em} $w_{t+1} = w_t + \frac{1}{\eta_t} \nabla Q_w(w_t)$ \textit{[Subgradient Update Step]}
8: $w_{t+1} = P_W(w_{t+1/2}) \textit{[Projection Step]}
9: End For
10: Output: $w_{t^*}$, where $t^* = \arg\min_{t \leq t_{\text{max}} + 1} Q(w_t)$

[34]). Consider the following unconstrained form of OP2:

$$\min_w \frac{1}{2}||w||^2 + C \max_{z \in Z, \pi \in \Pi_{m,j}} H(S, z, \pi; w), \textit{ (OP3)}$$

where $H(S, z, \pi; w) = \Delta_{\beta}(\pi^*, \pi) - w^T(\phi_z(S, \pi^*) - \phi_z(S, \pi))$. As in [34], this optimization problem is in turn can be reformulated into the following equivalent constrained optimization problem, where the regularization term is now part of an inequality constraint:

$$\min_w \max_{z \in Z, \pi \in \Pi_{m,j}} H(S, z, \pi; w), \textit{ s.t. } ||w||_2 \leq \lambda \textit{ (OP4)}$$

where for every value of $C > 0$ in OP3, there exists a value of $\lambda > 0$ in OP4 for which the two optimization problems have the same solution. OP4 can be efficiently solved using a projected subgradient method, as described below.

Let $Q(w)$ denote the objective function in OP4 and let $W = \{w \in R^d : ||w||_2 \leq \lambda\}$ denote the feasible set. The projected subgradient method (outlined in Algorithm 3) starts with an initial solution $w_0$ in $W$, and on each iteration $t$, performs a two step update, involving a subgradient based update and a projection:

$$w_{t+1} = P_W(w_t - \eta_t \nabla_w Q(w_t)),$$

where $P_W$ denotes the Euclidean projection onto $W$, $\nabla_w Q$ is a subgradient of $Q$ with respect to $w$, and $\eta_t$ is an appropriate step size.

Note that $Q(w)$ is a point-wise maximum of a set of linear functions; hence, one subgradient of $Q$ (with respect to $w$) at $w_t$ is the gradient of the linear function that attains the highest value at $w_t$ [5]:

$$\nabla Q_w(w_t) = \phi_z(S, \pi^*) - \phi_z(S, \pi),$$

where $(\pi, \tilde{z})$ is the maximizer of $H(S, z, \pi; w_t)$ over $z \in Z_\beta$ and $\pi \in \Pi_{m,j_3}$, which can be computed efficiently using the procedure outlined in Algorithm 2.

From standard results [6], one can show that when $\eta_t = \eta_0 / \sqrt{t}$, for some $\eta_0 > 0$, the projected subgradient method takes $O(1/\epsilon^2)$ iterations\(^5\) to reach a solution that is $\epsilon$-close to the optimal solution. Since the subgradient computation can be performed in $O(n \log n + (m+j_3) \log (m+j_3) + (m+n)d)$ time and the projection onto the $\ell_2$-ball can be performed in $O(d)$ time, the total time taken by the algorithm to reach an $\epsilon$-optimal solution is $O(n \log n + (m+j_3) \log (m + j_3) + (m+n)d)/\epsilon^2$.

\(^5\)While the method described uses linear scoring functions, one can extend it to handle non-linear scoring functions by incorporating kernels in the primal formulation (see [26]).

\(^6\)Here $\tilde{O}$ hides only polylogarithmic factors in $1/\epsilon$. 

6. SPARSE EXTENSIONS OF SVM\textsuperscript{tight}\_\textsubscript{pAUC}

In this section, we discuss how the SVM\textsuperscript{tight}\_\textsubscript{pAUC} method can be extended to learn sparse models, often of interest in biological applications. In particular, we discuss how the SVM\textsuperscript{pAUC} optimization problem can be solved when the regularizer used is not the \( \ell_2 \) norm, but instead a sparsity-inducing regularizer \( \Omega : \mathbb{R}^d \to \mathbb{R} \), such as the \( \ell_1 \) regularizer (known as lasso penalty in regression settings [28]), the elastic net regularizer [35], or the mixed \( \ell_1/\ell_2 \) regularizer (known as group lasso penalty in regression settings [33]). As discussed further in our experiments, such sparsity-inducing regularizers are useful in several applications where the partial AUC performance measure is of interest.

Indeed, both the \( \ell_1 \) penalty, \( \Omega(w) = \|w\|_1 \), and the elastic net penalty, \( \Omega(w) = \kappa\|w\|_1 + (1-\kappa)\frac{1}{2}\|w\|_2^2 \) with \( 0 \leq \kappa \leq 1 \), which is a convex combination of the \( \ell_1 \) and \( \ell_2 \) penalties, are useful in applications such as drug discovery and gene selection for cancer diagnosis, where a small subset of features that yield high accuracy needs to be selected; while the \( \ell_1 \) penalty is known to yield highly sparse models, often at the cost of accuracy, the elastic net penalty strikes a trade off between sparsity and accuracy.

On the other hand, the group lasso penalty is of interest in applications where the features fall into natural groups and a small set of feature groups needs to be selected; this is the case for example with the protein-protein interaction prediction task we consider, where the features fall into natural groups (each corresponding to a different data source), and it is desirable to learn a prediction model that uses a small set of such feature groups (data sources). More formally, given a set of \( P \) non-overlapping groups into which the \( d \) features can be divided, say \( \{G_1, \ldots, G_P\} \), where \( \cup_{p=1}^P G_p = \{1, \ldots, d\} \) and \( \cap_{p=1}^P G_p = \phi \), the group lasso penalty can be defined as \( \Omega(w) = \left( \sum_{p=1}^P \|w_{G_p}\|_2^2 \right)^2 \), where \( w_{G_p} \) is a vector of weights corresponding to the features in \( G_p \). Note that the group lasso applies \( \ell_1 \) regularization at the group level and \( \ell_2 \) regularization on weights within each group, thus promoting sparsity at the group level, while penalizing model complexity within each group.

While for each of these sparsity-inducing regularizers, one could potentially use a cutting-plane style method to solve the resulting optimization problem, owing to high training times observed with the cutting-plane method when applied to learn sparse models [34, 4] (this is also confirmed by our experiments in the next section), we instead resort to the projected subgradient method, which offers a clean and elegant way of incorporating different regularizers (as long as the projection step can be performed efficiently). In particular, we are interested in solving the following optimization problem using the projected subgradient technique for different sparsity-inducing norms \( \Omega(w) \).

\[
\min_w \max_{S \in \mathcal{Z}, z \in \Omega_{\min}} H(S, z, \pi; w), \quad \text{s.t. } \Omega(w) \leq \lambda.
\]

In the case of the \( \ell_1 \) penalty, the projection step in Algorithm 3 can be performed efficiently in time linear in the number of dimensions using the algorithm developed in [11]; in the case of the elastic net penalty, an extension of the same algorithm [10] allows efficient projection in linear time; with the group lasso penalty, the projection step can be efficiently computed in linear time using the algorithm in [29].

7. EXPERIMENTAL RESULTS

In this section, we give extensive experimental evaluations of the proposed method on real-world and synthetic data. We present three sets of experimental results: comparison between the partial AUC performances of the proposed SVM\textsuperscript{tight}\_\textsubscript{pAUC} method and the earlier structural SVM method, SVM\textsubscript{struct}_\textsubscript{pAUC}; comparison of the run time performances of the different optimization techniques for solving the SVM\textsuperscript{pAUC} and SVM\textsuperscript{struct}_\textsubscript{pAUC} optimization problems; and evaluation of the different sparse extensions of SVM\textsuperscript{tight}\_\textsubscript{pAUC}. We start by describing various bioinformatics tasks where the partial AUC is of interest and which will be used in our experiments.

7.1 Bioinformatics Tasks and Data Sets

Drug Discovery. 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; in this application, it is of interest to optimize partial AUC in a small FPR range \([0, \beta]\), which corresponds to optimizing ranking accuracy at the top of the list. We used two data sets for this task. The first is a virtual screening data set from [16], which contains 2142 compounds, each represented as a 1021-bit vector using the FP2 molecular fingerprint representation as in [2]; there are 5 sets of 50 active compounds each (active against 5 different targets), and 1892 inactive compounds, where for each target, the 50 active compounds are treated as positive, and all others as negative. The second data set is from the KDD Cup 2001 challenge\textsuperscript{6}; this contains 1909 compounds, each represented by 139,351 binary features, of which 42 compounds are active (known to bind well to a target receptor, thrombin), while the remaining are inactive.

Protein-Protein Interaction (PPI) Prediction. Here the goal is to predict whether a pair of proteins interact or not; owing to the highly imbalanced nature of PPI data, the partial AUC in a small FPR range \([0, \beta]\) has been advocated as a performance measure for this task [21]. We used the PPI data for yeast obtained from [21]\textsuperscript{7}; which contains 2865 protein pairs known to be interacting and a random set of 237,384 protein pairs taken as non-interacting. Each protein pair is represented using 162 features, grouped into 17 groups based on the data source they were obtained from.

Gene Ranking. Here the goal is to rank genes by relevance to a disease; here again the partial AUC in a small FPR range \([0, \beta]\), which captures ranking performance at the top of the list, is useful. We used a Leukemia microarray gene expression data set for this task, obtained from [12]; this consists of 7129 genes, each represented using 72 features (corresponding to gene expression levels in different tissue samples); out of these 18 genes are known to be associated with leukemia (positive), while 157 genes are known to be irrelevant to the disease (negative).

Medical Diagnosis. In many medical diagnosis tasks, the performance measure of interest is the partial AUC in a clinically relevant portion of the ROC curve; this can either be an FPR range of the form \([0, \beta]\) or more generally a range \([\alpha, \beta]\) for \( \alpha > 0 \). We consider three such tasks. The first is ovarian cancer diagnosis from protein biomarkers;
Table 1: SVM\textsubscript{pAUC}(0, β) on different bioinformatics data sets.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>SVM\textsubscript{pAUC}(0, β)</th>
<th>SVM\textsubscript{AUC}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheminformatics</td>
<td>57.10</td>
<td>53.43</td>
</tr>
<tr>
<td>KDD Cup 2001</td>
<td>65.30</td>
<td>62.08</td>
</tr>
<tr>
<td>PPI</td>
<td>62.20</td>
<td>62.03</td>
</tr>
<tr>
<td>Leukemia</td>
<td>70.12</td>
<td>65.30</td>
</tr>
<tr>
<td>Ovarian Cancer</td>
<td>52.95</td>
<td>43.98</td>
</tr>
<tr>
<td>Diabetes</td>
<td>56.28</td>
<td>62.23</td>
</tr>
<tr>
<td>PPI</td>
<td>25.49</td>
<td>43.98</td>
</tr>
<tr>
<td>Ovarian Cancer</td>
<td>90.64</td>
<td>89.14</td>
</tr>
</tbody>
</table>

Table 2: SVM\textsubscript{pAUC}(α, β) on medical data sets.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>SVM\textsubscript{pAUC}(α, β)</th>
<th>SVM\textsubscript{AUC}</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD Cup 2008</td>
<td>53.43</td>
<td>50.66</td>
</tr>
<tr>
<td>Diabetes</td>
<td>62.04</td>
<td>48.72</td>
</tr>
<tr>
<td>Cheminformatics</td>
<td>51.89</td>
<td>62.08</td>
</tr>
</tbody>
</table>

Figure 2: Timing statistics: SVM\textsubscript{pAUC}(0, β) vs. SVM\textsubscript{struct}(0, β) on synthetic data.

7.3 Run-time Comparisons

Our second set of experiments involved a comparison of the run-time performance of the different optimization algorithms.\textsuperscript{11} In order to evaluate the effect of number of examples and data dimensionality, we used synthetic data containing \(N\) examples in \(\mathbb{R}^d\) for different \(N\) and \(d\); in each case, 10% of the examples were positive and the rest were negative. Positive examples were drawn from a multivariate Gaussian distribution \(N(\mu, \Sigma)\) with mean \(\mu \in \mathbb{R}^d\) and covariance matrix \(\Sigma \in \mathbb{R}^{d \times d}\); negative examples were drawn from \(N(-\mu, \Sigma)\). Here \(\mu\) was drawn uniformly from \([-1, 1]^d\), while \(\Sigma\) was drawn from a Wishart distribution.

We first compared the performances of the cutting-plane solvers used with SVM\textsubscript{pAUC} and SVM\textsubscript{struct} methods in terms of (a) time taken to find the most-violated constraint (MVC), and (b) the number of calls to this routine, focusing on FPR parameter \(\epsilon\) was set to \(10^{-4}\). For the PPI and KDD Cup 2001 data sets, the parameter \(C\) was selected using the validation set from the ranges \([10^{-2}, 10^{-1}, 1, 10, 10^2]\) and \([10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]\), respectively; for the remaining data sets, \(C\) was selected via 5-fold cross-validation (on the training set) from the ranges \([10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]\), \([10^{-2}, 10^{-1}, 1, 10, 10^2]\), and \([10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1}\), respectively. For the PPI data set, only a subset of 85 features with less than 25% missing values was used.\textsuperscript{12} The KDD Cup 2008 data set was split into 5%-95% train-test sets; the Diabetes data set was split into 2:1 train-test sets; \(\epsilon\) was set to \(10^{-4}\); \(C\) was selected via 5-fold cross-validation (on the train set) from \([10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]\) and \([10^{-3}, 10^{-2}, 10^{-1}, 1, 10]\), respectively.\textsuperscript{13} All experiments in this section were run on an Intel Xeon (2.13 GHz) machine with 12 GB RAM.
Table 3: SVM\textsuperscript{t\_pAUC\_tight} with sparsity-inducing regularizers on Cheminformatics and KDD Cup 2001 data sets; % of features selected is reported in brackets.

Table 4: Group Sparsity: SVM\textsuperscript{t\_pAUC\_tight} (with \ell_2-regularizer and \ell_1/\ell_2-regularizer) on PPI data set.

7.4 Evaluation of Sparse Extensions of SVM\textsuperscript{t\_pAUC\_tight}

Our final set of experiments involved evaluating the different sparse versions of SVM\textsuperscript{t\_pAUC\_tight} on real-world data sets; all sparse methods were implemented using the projected subgradient method. We evaluated the \ell_1 and elastic net regularized versions of SVM\textsuperscript{t\_pAUC\_tight}, on the Cheminformatics (1021 features) and KDD Cup 2001 (139,351 features) drug discovery data sets and compared their performance (in terms of partial AUC and number of features selected) with that of the \ell_2 regularized SVM\textsuperscript{t\_pAUC\_tight}. The results, averaged over 10 random train-test splits, are shown in Table 3.\textsuperscript{16} On both data sets, the \ell_2 regularizer does not give sparse models, while the elastic net regularizer for small values of \kappa (0.001) yields models that use only around 40% of the features on average, but in terms of partial AUC, perform comparable to the models learnt using the \ell_2 regularizer. The \ell_1 regularizer on the other hand yields highly sparse models (selecting around 10% of the features on average), but performs poorly on partial AUC. We also evaluated the (group) \ell_1/\ell_2 regularized version of SVM\textsuperscript{t\_pAUC\_tight} for the PPI dataset, with all the 162 features (17 groups) used; the results, averaged over 10 random train-validation-test splits, are shown in Table 4. The \ell_1/\ell_2 regularized version picked 11.3 groups on average, yielding partial AUC values close to the \ell_2 regularized version which picked all 17 groups.\textsuperscript{17,18}

8. CONCLUSION

We have developed a new support vector formulation for optimizing the partial AUC performance measure using a tighter convex upper bound on the partial AUC loss than a...
previous structural SVM approach, yielding both improved accuracy and reduced computational complexity. We proposed two different optimization techniques for solving the resulting optimization problem. Our experiments on a wide range of bioinformatics tasks demonstrate the effectiveness of our approach. We also develop sparse extensions of the proposed method, often of interest in biological applications.

9. ACKNOWLEDGMENTS

HN thanks P. Balamurugan for discussions on sparsity. HN thanks Google India for support to attend the conference. This work is supported in part by a Ramanujan Fellowship from DST to SA.

10. REFERENCES