$\text{SVM}^{\text{tight}}_{\text{pAUC}}$: A New Support Vector Method for Optimizing Partial AUC Based on a Tight Convex Upper Bound

Harikrishna Narasimhan
Indian Institute of Science
Bangalore 560012, India
harikrishna@csa.iisc.ernet.in

Shivani Agarwal
Indian Institute of Science
Bangalore 560012, India
shivani@csa.iisc.ernet.in

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, $\text{SVM}^{\text{tight}}_{\text{pAUC}}$, 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

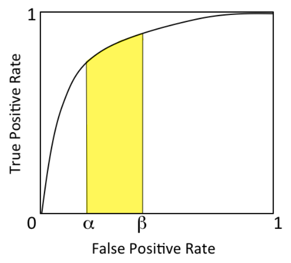

Figure 1: Partial AUC between false positive rates α and β.

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 $\text{SVM}^{\text{struct}}_{\text{pAUC}}$, 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, $\text{SVM}^{\text{tight}}_{\text{pAUC}}$, 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 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 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 $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ [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 $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ in Section 2. Section 3 characterizes the upper bound on partial AUC loss optimized by $\text{SVM}^{\text{struct}}_{\text{pAUC}}$, and motivates the development of our new method. Section 4 describes our new formulation $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ 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 $f : X \to \mathbb{R}$ that has good performance in terms of the partial AUC between some specified false positive rates α and β, where $0 \le \alpha < \beta \le 1$.

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:[1]

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

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 $\text{FPR}_f^{-1}(u) = \inf\big\{t \in \mathbb{R} \mid \text{FPR}_f(t) \le u\big\}$. Assuming there are no ties, the AUC can be written as [8]

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

Our interest here is in the area under the curve between FPRs α and β. 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 $f : X \to \mathbb{R}$; 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

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

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][2]

$$\widehat{\text{pAUC}}_f(\alpha, \beta) = \sum_{i=1}^m \sum_{j=j_\alpha+1}^{j_\beta} \mathbf{1}\big(f(x_i^+) > f(x^-_{(j)_f})\big),$$

where $(j)_f$ 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 $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ [18]

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

$$\begin{aligned}\widehat{R}(f) \;&=\; 1 - \widehat{\text{pAUC}}_f(\alpha, \beta)\\ &=\; \sum_{i=1}^m \sum_{j=j_\alpha+1}^{j_\beta} \mathbf{1}\big(f(x_i^+) < f(x^-_{(j)_f})\big). \qquad (1)\end{aligned}$$

[1]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.

[2]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 $f(x) = w^\top x$ 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:

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

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 π^* has $\pi^*_{ij} = 0\ \forall i,j$. For any $\pi \in \Pi_{m,n}$, we can define the pAUC loss of π with respect to π^* as the empirical pAUC risk of an ordering consistent with π:

$$\Delta(\pi^*, \pi) \;=\; \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=j_\alpha+1}^{j_\beta} \pi_{i,(j)_\pi}\,, \qquad (2)$$

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.

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

$$\phi(S, \pi) = \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=1}^{n} (1-\pi_{ij})(x_i^+ - x_j^-)\,, \qquad (3)$$

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

$$\min_{w,\xi\ge 0} \frac{1}{2}||w||_2^2 + C\xi \qquad \text{(OP1)}$$

$$\text{s.t. } \forall \pi \in \Pi_{m,n}: \quad w^\top\big(\phi(S,\pi^*) - \phi(S,\pi)\big) \;\ge\; \Delta(\pi^*,\pi) - \xi\,,$$

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$^{\textbf{struct}}_{\textbf{pAUC}}$

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 $\gamma \ge 0$, denote the pairwise γ-margin hinge loss of $w \in \mathbb{R}^d$ on the instance pair (x_i^+, x_j^-) as

$$\ell_{ij}^{(\gamma)}(w) = \big(\gamma - w^\top(x_i^+ - x_j^-)\big)_+\,, \qquad (4)$$

[3]The methods discussed can be extended to non-linear functions and non-Euclidean instance spaces using kernels.

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 $0 < \beta \le 1$. Then for any $w \in \mathbb{R}^d$, the smallest $\xi \ge 0$ satisfying the constraints of OP1 evaluates to*

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

where

$$\xi_{\text{pAUC}} \;=\; \frac{1}{mn\beta} \sum_{i=1}^{m} \sum_{j=1}^{j_\beta} \ell^{(1)}_{i,(j)_w}(w)$$

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

We omit the proof here due to space constraints; please see [19] for details. In the above theorem, the slack variable ξ (error term in OP1) is decomposed into a sum of two terms: the first term ξ_{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 ξ_{extra} is an additional non-negative term. We can show a similar result for FPR intervals $[\alpha, \beta]$ with $\alpha > 0$:

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

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

where

$$\xi_{\text{pAUC}} \;\ge\; \frac{1}{mn(\beta-\alpha)} \sum_{i:\, w^\top x_i^+ < w^\top x^-_{(j_\alpha)_w}} \; \sum_{j=j_\alpha+1}^{j_\beta} \ell^{(1)}_{i,(j)_w}(w);$$

$$\xi_{\text{pAUC}} \;\le\; \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=j_\alpha+1}^{j_\beta} \ell^{(1)}_{i,(j)_w}(w);$$

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

See [19] for the proof. Again, one can see that the slack variable ξ 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 Δ (see Eq. (2)) and the joint-feature map ϕ (see Eq. (3)) terms occurring in the constraints of OP1; while the former is computed over only the negative instances ranked in positions $\{j_\alpha + 1, \ldots, j_\beta\}$ (among all negative instances) by π, 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 Δ and ϕ so as to reduce this mismatch, thus yielding an upper bound on the pAUC risk without the additional negative term.

4. NEW FORMULATION: SVM$^{\textbf{tight}}_{\textbf{pAUC}}$

We now derive a new SVM formulation for optimizing the partial AUC that allows us to get rid of the ξ_{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 $\widehat{R}$. Let $\mathcal{Z}_\beta = \binom{S_-}{j_\beta}$ denote the set of all subsets of negative training instances of size j_β. Let us first consider the special case when $\alpha = 0$. In this case, the pAUC risk for a score function f is given by

$$\widehat{R}(f) = \sum_{j=1}^{j_\beta} \sum_{i=1}^{m} \mathbf{1}\big(f(x_i^+) < f(x_{(j)_f}^-)\big) . \tag{5}$$

This can be rewritten as

$$\widehat{R}(f) = \max_{z \in \mathcal{Z}_\beta} \underbrace{\sum_{x_j^- \in z} \sum_{i=1}^{m} \mathbf{1}\big(f(x_i^+) < f(x_j^-)\big)}_{(1-\widehat{AUC}_f) \text{ evaluated on } (S_+, z)}, \tag{6}$$

which can be viewed as the maximum value of $(1 - \widehat{\mathrm{AUC}}_f)$ attainable on subsets of negative instances of size j_β. To see that the expressions in Eq. (5) and Eq. (6) are equivalent, note that the maximum value of $(1-\widehat{\mathrm{AUC}}_f)$ over all subsets z in Eq. (6) is attained for the subset of negative instances ranked in the top j_β 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:

$$\widehat{R}_z(f) = \sum_{j=j_\alpha+1}^{j_\beta} \sum_{i=1}^{m} \mathbf{1}\big(f(x_i^+) < f(x_{(j)_{f|z}}^-)\big) ,$$

where $(j)_{f|z}$ 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

$$\widehat{R}(f) = \max_{z \in \mathcal{Z}_\beta} \widehat{R}_z(f). \tag{7}$$

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_β 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 $\widehat{R}_z$ 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 $z = \{x_{k_1}^-, \ldots, x_{k_{j_\beta}}^-\} \in \mathcal{Z}_\beta$ of size j_β, we define the joint feature map $\phi_z : (X^m \times X^n) \times \Pi_{m,j_\beta} \to \mathbb{R}^d$ 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:

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

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

$$\Delta_\beta(\pi^*, \pi) = \frac{1}{mn(\beta-\alpha)} \sum_{i=1}^{m} \sum_{j=j_\alpha+1}^{j_\beta} \pi_{i,(j)_\pi} .$$

Then our new formulation $\mathrm{SVM}_{\mathrm{pAUC}}^{\mathrm{tight}}$ consists of solving the following convex optimization problem:

$$\min_{w, \xi \geq 0} \ \frac{1}{2}||w||_2^2 + C\xi \tag{OP2}$$

$$\text{s.t. } \forall z \in \mathcal{Z}_\beta, \ \pi \in \Pi_{m,j_\beta} :$$
$$w^\top\big(\phi_z(S, \pi^*) - \phi_z(S, \pi)\big) \ \geq \Delta_\beta(\pi^*, \pi) - \xi ,$$

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 $\widehat{R}$ Optimized by $\mathbf{SVM}_{\mathbf{pAUC}}^{\mathbf{tight}}$. 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:*

$$\xi = \xi_{\mathrm{pAUC}}$$

where ξ_{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 ξ_{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 $\mathrm{SVM}_{\mathrm{pAUC}}^{\mathrm{tight}}$

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 $\mathrm{SVM}_{\mathrm{pAUC}}^{\mathrm{struct}}$ 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 $\mathcal{C} = \emptyset$, and on each iteration, adds the most-violated constraint to $\mathcal{C}$, thereby solving a tighter relaxation of OP1 in the subsequent iteration; the algorithm stops when no constraint is violated by more than ϵ (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).

Algorithm 1 Cutting-Plane Method for $\text{SVM}^{\text{tight}}_{\text{pAUC}}$

```
1: Inputs: $S = (S_+, S_-),\ \alpha,\ \beta,\ C,\ \epsilon$
2: Initialize: $\mathcal{C} = \emptyset$
3: $H(S, z, \pi; w) \equiv \Delta_\beta(\pi^*, \pi) - w^\top\big(\phi_z(S, \pi^*) - \phi_z(S, \pi)\big)$
4: Repeat
5:     $(w, \xi) = \operatorname{argmin}_{w, \xi \geq 0} \frac{1}{2}||w||_2^2 + C\xi$
           s.t. $\forall (z, \pi) \in \mathcal{C}:\ \xi \geq H(S, z, \pi; w)$
6:     $(\bar{z}, \bar{\pi}) = \operatorname{argmax}_{z \in \mathcal{Z}_\beta,\ \pi \in \Pi_{m, j_\beta}} H(S, z, \pi; w)$
           (compute the most-violated constraint)
7:     $\mathcal{C} = \mathcal{C} \cup \{(\bar{z}, \bar{\pi})\}$
8: Until $(H(S, \bar{z}, \bar{\pi}; w) \leq \xi + \epsilon)$
9: Output: $w$
```

Algorithm 2 $\text{SVM}^{\text{tight}}_{\text{pAUC}}$: Find Most-Violated Constraint

```
1: Inputs: $S = (S_+, S_-),\ \alpha,\ \beta,\ w$
2: Set $\bar{z}$ to the set of negative instances in the top $j_\beta$ positions in
   the ranking of negative instances in $S_-$ (in descending order
   of scores) by $w^\top x$
3: Obtain $\bar{\pi}$ by applying the procedure for finding the most-
   violated constraint in $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ on $(S_+, \bar{z})$ (see [18])
4: Output: $(\bar{z}, \bar{\pi})$
```

Finding Most-Violated Constraint. It can be shown that in the solution $(\bar{z}, \bar{\pi})$ to the combinatorial optimization problem in Algorithm 1 (line 6), the set $\bar{z}$ contains the top j_β negative instances ranked (among all negative instances) by $w^\top x$ (see [19]). Hence, finding $\bar{z}$ simply involves sorting the negative instances in S_- according to $w^\top x$. Having fixed $\bar{z}$, finding the ordering matrix $\bar{\pi}$ then reduces to finding the most-violated constraint in $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ [18], but with a smaller set of instances $(S_+, \bar{z})$ (see Algorithm 2) .

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

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

$$\left\lceil \log_2\left(\frac{1}{4R^2_{\text{tight}}C}\right)\right\rceil + \left\lceil \frac{16R^2_{\text{tight}}C}{\epsilon}\right\rceil,$$

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

$$\left\lceil \log_2\left(\frac{1}{4R^2_{\text{struct}}C}\right)\right\rceil + \left\lceil \frac{16R^2_{\text{struct}}C}{\epsilon}\right\rceil,$$

where $R_{\text{struct}} = \frac{1}{\beta - \alpha}\max_{i,j}||x_i^+ - x_j^-||_2$; this gives $R_{\text{struct}} = \frac{1}{\beta}R_{\text{tight}} \geq R_{\text{tight}}$. The larger value of R_{struct} than R_{tight} is due to the fact that the joint feature map ϕ in $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ is defined over the entire set of negative instances, whereas the joint feature map ϕ_z in $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ is defined over only a subset of negative instances z of size j_β.

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 $\text{SVM}^{\text{tight}}_{\text{pAUC}}$

```
1: Inputs: $S = (S_+, S_-),\ \alpha,\ \beta,\ C,\ \eta_0,\ t_{\max}$
2: Initialize: $w_0$ = Initial solution in $W$
3: $H(S, z, \pi; w) \equiv \Delta_\beta(\pi^*, \pi) - w^\top\big(\phi_z(S, \pi^*) - \phi_z(S, \pi)\big)$
4: For $t = 1$ to $t_{\max}$ do:
5:     $(\bar{z}, \bar{\pi}) = \operatorname{argmax}_{z \in \mathcal{Z}_\beta,\ \pi \in \Pi_{m, j_\beta}} H(S, z, \pi; w_t)$
6:     $\nabla Q_w(w_t) = \phi_{\bar{z}}(S, \pi^*) - \phi_{\bar{z}}(S, \bar{\pi})$
7:     $w_{t+1/2} = w_t - \frac{\eta_0}{\sqrt{t}}\nabla Q_w(w_t)$ [Subgradient Update Step]
8:     $w_{t+1} = \mathcal{P}_W\big(w_{t+1/2}\big)$ [Projection Step]
9: End For
10: Output: $w_{t^*}$,   where $t^* = \operatorname{argmin}_{1 \leq t \leq t_{\max}+1} Q(w_t)$
```

[34]).[4] Consider the following unconstrained form of OP2:

$$\min_w \frac{1}{2}||w||_2^2 + C \max_{z \in \mathcal{Z}_\beta,\ \pi \in \Pi_{m, j_\beta}} H(S, z, \pi; w), \qquad \text{(OP3)}$$

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

$$\min_w \Big[\max_{z \in \mathcal{Z}_\beta,\ \pi \in \Pi_{m, j_\beta}} H(S, z, \pi; w)\Big], \ \text{ s.t. } ||w||_2 \leq \lambda, \ \text{(OP4)}$$

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 $W = \{w \in \mathbb{R}^d \mid ||w||_2 \leq \lambda\}$ 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:

$$w_{t+1} = \mathcal{P}_W\big(w_t - \eta_t \nabla_w Q(w_t)\big),$$

where $\mathcal{P}_W$ denotes the Euclidean projection onto W, $\nabla_w Q$ is a subgradient of Q with respect to w, and η_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]:

$$\nabla Q_w(w_t) = \phi_{\bar{z}}(S, \pi^*) - \phi_{\bar{z}}(S, \bar{\pi}),$$

where $(\bar{\pi}, \bar{z})$ is the maximizer of $H(S, z, \pi; w_t)$ over $z \in \mathcal{Z}_\beta$ and $\pi \in \Pi_{m, j_\beta}$, 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 $\widetilde{O}(1/\epsilon^2)$ iterations[5] to reach a solution that is ϵ-close to the optimal solution. Since the subgradient computation can be performed in $O\big(n \log n + (m{+}j_\beta) \log(m{+}j_\beta) + (m{+}n)d\big)$ time and the projection onto the ℓ_2-ball can be performed in $O(d)$ time, the total time taken by the algorithm to reach an ϵ-optimal solution is $\widetilde{O}\big((n \log n + (m + j_\beta) \log(m + j_\beta) + (m + n)d)/\epsilon^2\big)$.

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

[5]Here $\widetilde{O}$ hides only polylogarithmic factors in $1/\epsilon$.

6. SPARSE EXTENSIONS OF SVM$^{\text{tight}}_{\text{pAUC}}$

In this section, we discuss how the SVM$^{\text{tight}}_{\text{pAUC}}$ method can be extended to learn sparse models, often of interest in biological applications. In particular, we discuss how the SVM$^{\text{tight}}_{\text{pAUC}}$ optimization problem can be solved when the regularizer used is not the ℓ_2 norm, but instead a sparsity-inducing regularizer $\Omega : \mathbb{R}^d \to \mathbb{R}$, such as the ℓ_1 regularizer (known as lasso penalty in regression settings [28]), the elastic net regularizer [35], or the mixed ℓ_1/ℓ_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 ℓ_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 $0 \le \kappa \le 1$, which is a convex combination of the ℓ_1 and ℓ_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 ℓ_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 $\{\mathcal{G}_1, \ldots, \mathcal{G}_P\}$, where $\cup_{p=1}^{P}\mathcal{G}_p = \{1, \ldots, d\}$ and $\cap_{p=1}^{P}\mathcal{G}_p = \phi$, the group lasso penalty can be defined as $\Omega(w) = \big(\sum_{p=1}^{P} \|w_{\mathcal{G}_p}\|_2\big)^2$, where $w_{\mathcal{G}_p}$ is a vector of weights corresponding to the features in $\mathcal{G}_p$. Note that the group lasso applies ℓ_1 regularization at the group level and ℓ_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 $\Omega(w)$.

$$\min_{w} \Big[\max_{z \in \mathcal{Z}_\beta,\ \pi \in \Pi_{m,j_\beta}} H(S, z, \pi; w)\Big], \quad \text{s.t.} \ \ \Omega(w) \le \lambda.$$

In the case of the ℓ_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$^{\text{tight}}_{\text{pAUC}}$ method and the earlier structural SVM method, SVM$^{\text{struct}}_{\text{pAUC}}$; comparison of the run time performances of the different optimization techniques for solving the SVM$^{\text{tight}}_{\text{pAUC}}$ and SVM$^{\text{struct}}_{\text{pAUC}}$ optimization problems; and evaluation of the different sparse extensions of SVM$^{\text{tight}}_{\text{pAUC}}$. 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[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][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;

[6]`http://pages.cs.wisc.edu/~dpage/kddcup2001/`
[7]`http://www.cs.cmu.edu/~qyj/papers_sulp/proteins05_PPI.html`

	pAUC$(0,\beta)$									
	Cheminformatics		KDD Cup 2001		PPI			Leukemia	Ovarian Cancer	
	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.01$	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.1$	$\beta = 0.1$	$\beta = 0.2$
$\text{SVM}^{\text{tight}}_{\text{pAUC}}[0,\beta]$	**57.10**	**65.30**	**62.20**	69.91	**25.49**	**43.98**	**52.95**	**30.44**	91.84	94.37
$\text{SVM}^{\text{struct}}_{\text{pAUC}}[0,\beta]$	57.03	65.28	61.53	**70.12**	23.06 **	41.70 **	51.96 **	24.64 **	91.84	94.29
SVM_{AUC}	53.98	62.78 *	51.45 **	62.23 **	13.51 **	29.57 **	39.72 **	28.83	**92.17**	**94.56**

Table 1: $\text{SVM}_{\text{pAUC}}[0,\beta]$ on different bioinformatics data sets.

	pAUC(α,β)	
	KDD Cup 2008 $[0.2s, 0.3s]$	Diabetes $[0.1, 0.2]$
$\text{SVM}^{\text{tight}}_{\text{pAUC}}[\alpha,\beta]$	**53.43**	60.94
$\text{SVM}^{\text{struct}}_{\text{pAUC}}[\alpha,\beta]$	51.89	48.72 **
SVM_{AUC}	50.66 *	**62.03**

Table 2: $\text{SVM}_{\text{pAUC}}[\alpha,\beta]$ on medical data sets.

this data set obtained from [7][8] contains 216 tissue samples, represented using 373,401 protein biomarkers, of which 121 samples are cancerous (positive) and the remaining are normal (negative). The second task we consider is the (early) breast cancer detection part of the KDD Cup 2008 challenge [23], where one needs to predict whether a given region of interest (ROI) from an X-ray image of the breast is malignant (positive) or benign (negative). This data set consists of information for 118 malignant patients and 1594 normal patients, where 4 X-ray images of the breast are available for each patient. Overall, there are 102,294 candidate ROIs from these X-ray images, with each candidate represented by 117 features. In the KDD Cup challenge, the performance measure for this task was a scaled version of partial AUC in a false positive range $[0.2s, 0.3s]$, where $s = 6848/101671$ (this is the total number of images divided by the total number of negative ROIs), deemed clinically relevant based on radiologist surveys. The third medical diagnosis data set we consider is the Pima Indian Diabetes data set (drawn from the UCI machine learning repository[9]), which consists of 768 samples corresponding to female patients of the Pima Indian heritage, with 268 having diabetes (positive); each sample here is described by 8 numerical attributes.

7.2 Comparison of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ with $\text{SVM}^{\text{struct}}_{\text{pAUC}}$

Our first set of experiments involved a comparison of the partial AUC performances of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ and $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ on the bioinformatics tasks mentioned above; the cutting-plane solver was used with both the methods.[10] We first compared the performances of the two methods on partial AUC in FPR intervals $[0,\beta]$ for different values of β. Five different data sets (Cheminformatics, KDD Cup 2001, PPI, Leukemia, Ovarian Cancer) were used for this purpose; the results, averaged over 10 random train-test splits (subject to preserving the proportion of positives), are shown in Table 1.[11] We also compared the two methods on the general partial AUC measure in FPR interval $[\alpha,\beta]$ on two medical diagnosis data sets (KDD Cup 2008 and Diabetes); the results, averaged over 10 random train-test splits, are shown in Table 2.[12] In each case, the AUC optimization method due to Joachims [13] (SVM_{AUC}) was also included as a baseline. A single (double) star against a baseline method indicates a statistically significant difference in performance between $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ and the baseline method using the two-sided Wilcoxon test at a 90% (95%) confidence level. As can be seen, on many of the data sets considered, the $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ method achieves statistically significant improvements in partial AUC performance over $\text{SVM}^{\text{struct}}_{\text{pAUC}}$; also, in most cases, both methods perform better than SVM_{AUC}.

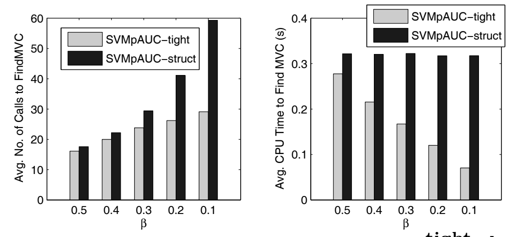

Figure 2: Timing statistics: $\text{SVM}^{\text{tight}}_{\text{pAUC}}[0,\beta]$ vs. $\text{SVM}^{\text{struct}}_{\text{pAUC}}[0,\beta]$ on synthetic data.

7.3 Run-time Comparisons

Our second set of experiments involved a comparison of the run-time performance of the different optimization algorithms.[13] 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 $\mathcal{N}(\mu,\Sigma)$ with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$; negative examples were drawn from $\mathcal{N}(-\mu,\Sigma)$. Here μ was drawn uniformly from $\{-1,1\}^d$, while Σ was drawn from a Wishart distribution.

We first compared the performances of the cutting-plane solvers used with $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ and $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ 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

[8]http://datam.i2r.a-star.edu.sg/datasets/krbd/OvarianCancer/OvarianCancer-NCI-QStar.html

[9]http://archive.ics.uci.edu/ml/

[10]Code for $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ is available at http://clweb.csa.iisc.ernet.in/harikrishna/Papers/SVMpAUC-tight

[11]The PPI data set was split into train-validation-test sets in the ratio 1:9:99; KDD Cup 2001 data set: 1/3:1/3:1/3; for the Leukemia data set, the train set contained a random set of 10 positive genes and all 157 negative genes, while the test set contained everything else; the remaining data sets were split into train-test sets as follows: Cheminformatics: 5:95; Ovarian cancer: 2:1. In each case, the parameter ϵ 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 $\{10^{-6},10^{-5},10^{-4},10^{-3},10^{-2}\}$, respectively. For the PPI data set, only a subset of 85 features with less than 25% missing values was used.

[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; ϵ 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 $\{10^{-3},10^{-2},10^{-1},1,10^2\}$, respectively.

[13]All experiments in this section were run on an Intel Xeon (2.13 GHz) machine with 12 GB RAM.

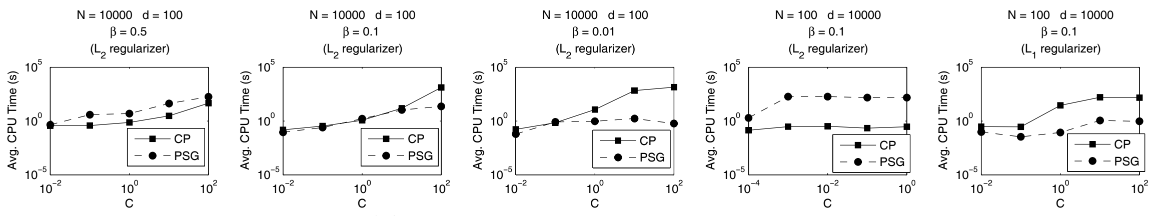

Figure 3: Timing statistics for $\mathbf{SVM^{tight}_{pAUC}}[0,\beta]$: Cutting-plane Method vs. Projected Subgradient Method on synthetic data.

	pAUC(0, 0.1)			
	Cheminformatics		KDD Cup 2001	
$\text{SVM}^{\ell_2}_{\text{pAUC}}[0,0.1]$	**63.25**	(100)	77.20	(100)
$\text{SVM}^{\text{elastic-net}(0.001)}_{\text{pAUC}}[0,0.1]$	63.11	(41.5)	**77.52**	(41.6)
$\text{SVM}^{\text{elastic-net}(0.1)}_{\text{pAUC}}[0,0.1]$	56.93	(32.24)	71.93	(27.6)
$\text{SVM}^{\ell_1}_{\text{pAUC}}[0,0.1]$	53.63	(**11.36**)	66.22	(**10.0**)

Table 3: $\mathbf{SVM^{tight}_{pAUC}}$ with sparsity-inducing regularizers on Cheminformatics and KDD Cup 2001 data sets; % of features selected is reported in brackets.

intervals of the form $[0,\beta]$. The data sets used for these experiments were of size $N = 10^5, d = 100$; the results, averaged over 10 such randomly generated data sets, are shown in Figure 2 (C and ϵ were set to 1 and 0.01 respectively for both the methods). As expected, the cutting-plane solver for $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ was better off than that for $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ in both aspects. Indeed, consistent with our observations in Section 5, for the $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ cutting-plane solver, the time taken to find the most-violated constraint was found to decrease with β, while that for the $\text{SVM}^{\text{struct}}_{\text{pAUC}}$ cutting-plane solver remained constant. For both solvers, the number of calls to the routine for finding the most-violated constraint increased with decrease in β, though this increase was slower for $\text{SVM}^{\text{tight}}_{\text{pAUC}}$.

We also performed a run-time comparison between the cutting-plane method and the projected subgradient method in solving the $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ optimization problem (considering both ℓ_1 and ℓ_2 regularizations), again focusing on FPR intervals $[0,\beta]$. For a fair comparison between the two methods, similar to [11], we ran the cutting-plane algorithm for a particular value of C and used the (appropriate) norm of the learnt weight vector as the value of λ in the projected subgradient method; the projected subgradient method was run until it reached a primal objective value equal to or lesser than that attained by the cutting-plane method.[14] The average run times (over 10 random data sets) for different values of C are shown in Figure 3. With the ℓ_2 regularizer, the projected subgradient method was found to run faster than the cutting-plane method on low dimensional data for small values of β and large values of C; this is because (on low dimensional data) a single iteration of this method requires lesser running time than a single cutting-plane iteration (which involves solving a quadratic program). With the ℓ_1 regularizer, however, the projected subgradient method was found to run faster than the cutting-plane method even on high dimensional data[15], motivating us to use the projected subgradient method for the sparse extensions of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$.

[14]The parameter ϵ was set to 0.01 and 0.1 respectively for the experiments involving the ℓ_2 and ℓ_1 regularizer, while η_0 was set to $\lambda/10$ and $\lambda/100$ respectively for these experiments.

[15]With the ℓ_1 regularizer, each cutting-plane iteration now involves solving a linear program (LP), which slows down the algorithm; there has been effort though to reduce the run time of this method by using specialized LP solvers [31].

	pAUC(0, 0.1)	# of groups selected
$\text{SVM}^{\ell_2}_{\text{pAUC}}[0,0.1]$	**67.09**	17
$\text{SVM}^{\ell_1/\ell_2}_{\text{pAUC}}[0,0.1]$	65.67	**11.3**

Table 4: Group Sparsity: $\mathbf{SVM^{tight}_{pAUC}}$ (with ℓ_2-regularizer and ℓ_1/ℓ_2-regularizer) on PPI data set.

7.4 Evaluation of Sparse Extensions of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$

Our final set of experiments involved evaluating the different sparse versions of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ on real-world data sets; all sparse methods were implemented using the projected subgradient method. We evaluated the ℓ_1 and elastic net regularized versions of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ 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 ℓ_2 regularized $\text{SVM}^{\text{tight}}_{\text{pAUC}}$. The results, averaged over 10 random train-test splits, are shown in Table 3.[16] On both data sets, the ℓ_2 regularizer does not give sparse models, while the elastic net regularizer for small values of κ (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 ℓ_2 regularizer. The ℓ_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) ℓ_1/ℓ_2 regularized version of $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ 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 ℓ_1/ℓ_2 regularized version picked 11.3 groups on average, yielding partial AUC values close to the ℓ_2 regularized version which picked all 17 groups.[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

[16]We note the partial AUC value reported for the Cheminformatics data set using ℓ_2 regularized $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ with the projected subgradient method in Table 3 is slightly different from that reported using ℓ_2 regularized $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ with the cutting-plane method in Table 1; this difference is due to different parameter ranges used for the two experiments.

[17]We note the partial AUC reported on the PPI data set for ℓ_2 regularized $\text{SVM}^{\text{tight}}_{\text{pAUC}}$ in Table 4 is higher than that in Table 1; this is due to the different number of features used.

[18]For these experiments, we used the same splits as before. The value of parameter λ (or $\sqrt{2\lambda}$ in the case of the ℓ_2 regularizer) was chosen from the range $\{10^{-4}, 10^{-2}, 1, 10^2, 10^4\}$ via cross-validation (or using the validation set); the initial step-size η_0 was set to λ/u, with the value of u chosen from a range to yield minimum objective value on the train set.

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

[1] S. Agarwal. The Infinite Push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *SDM*, 2011.

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

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

[4] P. Balamurugan, S. Shevade, and T. Babu. Sequential alternating proximal method for scalable sparse structural SVMs. In *ICDM*, 2012.

[5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[6] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. `http://www.stanford.edu/class/ee392o/subgrad_method.pdf`, 2003.

[7] T. P. Conrads, M. Zhou, E. F. I. Petricoin, L. Liotta, and T. D. Veenstra. Cancer diagnosis using proteomic patterns. *Expert Rev. Mol. Diagn.*, (3):411–420, 2003.

[8] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *NIPS*, 2004.

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

[10] J. Duchi. Elastic net projections, `http://www.cs.berkeley.edu/~jduchi/projects/proj_elastic_net.pdf`, 2009.

[11] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*, 2008.

[12] T. R. Golub. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[13] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005.

[14] T. Joachims. Training linear SVMs in linear time. In *KDD*, 2006.

[15] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

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

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

[18] H. Narasimhan and S. Agarwal. A structural SVM based approach for optimizing partial AUC. In *ICML*, 2013.

[19] H. Narasimhan and S. Agarwal. $\mathrm{SVM}_{\mathrm{pAUC}}^{\mathrm{tight}}$: A new support vector method for optimizing partial AUC based on a tight convex upper bound (full version). `http://clweb.csa.iisc.ernet.in/harikrishna/Papers/SVMpAUC-tight/kdd-full.pdf`, 2013.

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

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

[22] A. Rakotomamonjy. Sparse support vector infinite push. In *ICML*, 2012.

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

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

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

[26] Y. Singer and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML*, 2007.

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

[28] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[29] E. van den Berg, M. Schmidt, M. P. Friedlander, and K. Murphy. Group sparsity via linear-time projection. Technical Report TR-2008-09, UBC, 2008.

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

[31] Z. Wang and J. Shawe-Taylor. Large-margin structured prediction via linear programming. In *AISTATS*, 2009.

[32] S.-H. Wu, K.-P. Lin, C.-M. Chen, and M.-S. Chen. Asymmetric support vector machines: low false-positive learning under the user tolerance. In *KDD*, 2008.

[33] M. Yuan, M. Yuan, Y. Lin, and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[34] J. Zhu, E. P. Xing, and B. Zhang. Primal sparse max-margin markov networks. In *KDD*, 2009.

[35] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.