Large-scale Optimization of Partial AUC in a Range of False Positive Rates

Anonymous Author(s)
Affiliation
Address
email

Abstract

The area under the ROC curve (AUC) is one of the most widely used performance measures for classification models in machine learning. However, it summarizes the true positive rates (TPRs) over all false positive rates (FPRs) in the ROC space, which may include the FPRs with no practical relevance in some applications. The partial AUC, as a generalization of the AUC, summarizes only the TPRs over a specific range of the FPRs and is thus a more suitable performance measure in many real-world situations. Although partial AUC optimization in a range of FPRs had been studied, existing algorithms are not scalable to big data and not applicable to deep learning. To address this challenge, we cast the problem into a non-smooth difference-of-convex (DC) program for any smooth predictive functions (e.g., deep neural networks), which allowed us to develop an efficient approximated gradient descent method based on the Moreau envelope smoothing technique, inspired by recent advances in non-smooth DC optimization. To increase the efficiency of large data processing, we used an efficient stochastic block coordinate update in our algorithm. Our proposed algorithm can also be used to minimize the sum of ranked range loss, which also lacks efficient solvers. We established a complexity of $\tilde{O}(1/\epsilon^6)$ for finding a nearly ϵ-critical solution. Finally, we numerically demonstrated the effectiveness of our proposed algorithms in training both linear models and deep neural networks for partial AUC maximization and sum of ranked range loss minimization.

1 Introduction

The area under the receiver operating characteristic (ROC) curve (AUC) is one of the most widely used performance measures for classifiers in machine learning, especially when the data is imbalanced between the classes [7, 19]. Typically, the classifier produces a score for each data point. Then a data point is classified as positive if its score is above a chosen threshold; otherwise, it is classified as negative. Varying the threshold will change the true positive rate (TPR) and the false positive rate (FPR) of the classifier. The ROC curve shows the TPR as a function of the FPR that corresponds to the same threshold. Hence, maximizing the AUC of a classifier is essentially maximizing the classifier's average TPR over all FPRs from zero to one. However, for some applications, some FPR regions have no practical relevance. So does the TPR over those regions. For example, in clinical practice, a high FPR in diagnostic tests often results in a high monetary cost, so people may only need to maximize the TPR when the FPR is low [13, 34, 64]. Moreover, since two models with the same AUC can still have different ROCs, the AUC does not always reflect the true performance of a model that is needed in a particular production environment [8].

As a generalization of the AUC, the partial AUC (pAUC) only measures the area under the ROC curve that is restricted between two FPRs. A probabilistic interpretation of the pAUC can be found in [13]. In contrast to the AUC, the pAUC represents the average TPR only over a relevant range

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

of FPRs and provides a performance measure that is more aligned with the practical needs in some applications.

In literature, the existing algorithms for training a classifier by maximizing the pAUC include the boosting method [29] and the cutting plane algorithm [41–43]. However, the former has no theoretical guarantee, and the latter applies only to linear models. More importantly, both methods require processing all the data in each iteration and thus, become computationally inefficient for large datasets.

In this paper, we proposed an approximate gradient method for maximizing the pAUC that works for nonlinear models (e.g., deep neural networks) and only needs to process randomly sampled positive and negative data points of any size in each iteration. In particular, we formulated the maximization of the pAUC as a non-smooth difference-of-convex (DC) program [30, 54]. Due to non-smoothness, most existing DC optimization algorithms cannot be applied to our formulation. Motivated by [52], we approximate the two non-smooth convex components in the DC program by their Moreau envelopes and obtained a smooth approximation of the problem, which will be solved using the gradient descent method. Since the gradient of the smooth problem cannot be calculated explicitly, we approximated the gradient by solving the two proximal-point subproblems defined by each convex component using the stochastic block coordinate descent (SBCD) method. Our method, besides its low per-iteration cost, has a rigorous theoretical guarantee, unlike the existing methods. *In fact, we show that our method finds a nearly ϵ-critical point of the pAUC optimization problem in $\tilde{O}(\epsilon^{-6})$ iterations with only small samples of positive and negative data points processed per iteration.*[1] *This is the main contribution of this paper.*

Note that, for non-convex non-smooth optimization, the existing stochastic methods [10, 11] find an nearly ϵ-critical point in $O(\epsilon^{-4})$ iterations under a weak convexity assumption. Our method needs $O(\epsilon^{-6})$ iterations because our problem is a DC problem with both convex components non-smooth which is much more challenging than a weakly non-convex minimization problem. In addition, our iteration number matches the known best iteration complexity for non-smooth non-convex min-max optimization [33, 46] and non-smooth non-convex constrained optimization [35].

In addition to pAUC optimization, our method can be also used to minimize the sum of ranked range (SoRR) loss, which can be viewed as a special case of pAUC optimization. Many machine learning models are trained by minimizing an objective function, which is defined as the sum of losses over all training samples [60]. Since the sum of losses weights all samples equally, it is insensitive to samples from minority groups. Hence, the sum of top-k losses [17, 49] is often used as an alternative objective function because it provides the model with robustness to non-typical instances. However, the sum of top-k losses can be very sensitive to outliers, especially when k is small. To address this issue, [24] proposed the SoRR loss as a new learning objective, which is defined as the sum of a consecutive sequence of losses from any range after the losses are sorted. Compared to the sum of all losses and the sum of top-k losses, the SoRR loss maintains a model's robustness to a minority group but also reduces the model's sensitivity to outliers. See Fig.1 in [24] for an illustration of the benefit of using the SoRR loss over other ways of aggregating individual losses.

To minimize the SoRR loss, [24] applied a difference-of-convex algorithm (DCA) [3, 54], which linearizes the second convex component and solves the resulting subproblem using the stochastic subgradient method. DCA has been well studied in literature; but when the both components are non-smooth, as in our problem, only asymptotic convergence results are available. To establish the total number of iterations needed to find an ϵ-critical point in a non-asymptotic sense, most existing studies had to assume that at least one of the components is differentiable, which is not the case in this paper. Using the approximate gradient method presented in this paper, one can find a nearly ϵ-critical point of the SoRR loss optimization problem in $\tilde{O}(\epsilon^{-6})$ iterations.

2 Related Works

The pAUC has been studied for decades [26, 37, 57]. However, most studies focused on its estimation [13] and application as a performance measure, while only a few studies were devoted to numerical algorithms for optimizing the pAUC. Besides the boosting method [29] and the cutting plane algorithm [41–43] mentioned in the previous section, [59] and [65] developed surrogate opti-

[1] Throughout the paper, $\tilde{O}(\cdot)$ suppresses all logarithmic factors.

mization techniques that directly maximize a smooth approximation of the pAUC or the two-way pAUC [64]. However, their approaches can only be applied when the FPR starts from exactly zero. On the contrary, our method allows the FPR to start from any value between zero and one. [61] and [47] developed algorithms that use the pAUC as a criterion for creating a linear combination of multiple existing classifiers while we consider directly train a classifier using the pAUC.

DC optimization has been studied since the 1950s [2, 20]. We refer interested readers to [30, 45, 54, 55, 58], and the references therein. The actively studied numerical methods for solving a DC program include DCA [3, 50, 54, 55], which is also known as the concave-convex procedure [32, 51, 68], the proximal DCA [5, 38, 40, 53], and the direct gradient methods [28]. However, when the two convex components are both non-smooth, the existing methods have only asymptotic convergence results except the method by [1], who considered a stopping criterion different from ours. When at least one component is smooth, non-asymptotic convergence rates have been established with and without the Kurdyka-Łojasiewicz (KL) condition [5, 6, 28, 50, 62].

The algorithms mentioned above are deterministic and require processing the entire dataset per iteration. Stochastic algorithms that process only a small data sample per iteration have been studied [12, 21, 31, 36, 44]. However, they all assumed smoothness on at least one of the two convex components in the DC program. The stochastic methods of [4, 56, 63] can be applied when both components are non-smooth but their methods require an unbiased stochastic estimation of the gradient and/or value of the two components, which is not available in the DC formulation of the pAUC maximization problem in this paper.

The technique most related to our work is the smoothing method based on the Moreau envelope [16, 18, 22, 23, 39, 52]. Our work was motivated by [39, 52], but the important difference is that they studied deterministic methods and assumed either that one function is smooth or that the proximal-point subproblems can be solved exactly, which we do not assume. However, [52] developed an algorithm when there were linear equality constraints, which we do not consider in this paper.

Finally, it was brought to our attention that a recent work [69] also considered partial AUC maximization with a non-convex objective. However, the algorithms in [69] can only optimize one-way pAUC with FPR in a range $(0, \beta)$, while our algorithm is more general and can be applied to FRP in any range (α, β) with $\alpha \geq 0$, which is more challenging for $\alpha > 0$.

3 Preliminary

We consider a classical binary classification problem, where the goal is to build a predictive model that predicts a binary label $y \in \{1, -1\}$ based on a feature vector $\mathbf{x} \in \mathbb{R}^p$. Let $h_{\mathbf{w}} : \mathbb{R}^p \to \mathbb{R}$ be the predictive model parameterized by a vector $\mathbf{w} \in \mathbb{R}^d$, which produces a score $h_{\mathbf{w}}(\mathbf{x})$ for $\mathbf{x}$. Then $\mathbf{x}$ is classified as positive ($y = 1$) if $h_{\mathbf{w}}(\mathbf{x})$ is above a chosen threshold and classified as negative ($y = -1$), otherwise.

Let $\mathcal{X}_+ = \{\mathbf{x}_i^+\}_i^{N_+}$ and $\mathcal{X}_- = \{\mathbf{x}_i^-\}_i^{N_-}$ be the sets of feature vectors of positive and negative training data, respectively. The problem of learning $h_{\mathbf{w}}$ through maximizing its empirical AUC on the training data can be formulated as

$$\max_{\mathbf{w}} \frac{1}{N_+ N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} \mathbf{1}(h_{\mathbf{w}}(\mathbf{x}_i^+) > h_{\mathbf{w}}(\mathbf{x}_j^-)), \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function which equals one if the inequality inside the parentheses holds and equals zero, otherwise. According to the introduction, pAUC can be a better performance measure of $h_{\mathbf{w}}$ than AUC. Consider two FPRs α and β with $0 \leq \alpha < \beta \leq 1$. For simplicity of exposition, we assume $N_-\alpha$ and $N_-\beta$ are both integers. Let $m = N_-\alpha$ and $n = N_-\beta$. The problem of maximizing the empirical pAUC with FPR between α and β can be formulated as

$$\max_{\mathbf{w}} \frac{1}{N_+(n-m)} \sum_{i=1}^{N_+} \sum_{j=m+1}^{n} \mathbf{1}(h_{\mathbf{w}}(\mathbf{x}_i^+) > h_{\mathbf{w}}(\mathbf{x}_{[j]}^-)), \tag{2}$$

where $[j]$ denotes the index of the jth largest coordinate in vector $(h_{\mathbf{w}}(\mathbf{x}_j^-))_{j=1}^{N_-}$ with ties broken arbitrarily. Note that $N_+(n-m)$ in (2) is a normalizer that makes the objective value between

zero and one. Solving (2) is challenging due to discontinuity. Let $\ell : \mathbb{R} \to \mathbb{R}$ be a differential non-increasing loss function. Problem (2) can be approximated by the loss minimization problem

$$\min_{\mathbf{w}} \frac{1}{N_+(n-m)} \sum_{i=1}^{N_+} \sum_{j=m+1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}_i^+) - h_{\mathbf{w}}(\mathbf{x}_{[j]}^-)). \tag{3}$$

To facilitate the discussion, we first introduce a few notations. Given a vector $S = (s_i)_{i=1}^N \in \mathbb{R}^N$ and an integer l with $0 \le l \le N$, the sum of the top-l values in S is

$$\phi_l(S) := \textstyle\sum_{j=1}^{l} s_{[j]}, \tag{4}$$

where $[j]$ denotes the index of the jth largest coordinate in S with ties broken arbitrarily. For integers l_1 and l_2 with $0 \le l_1 < l_2 \le N$, $\phi_{l_2}(S) - \phi_{l_1}(S)$ is the sum from the (l_1+1)th to the l_2th (inclusive) largest coordinates of S, also called a *sum of ranked range* (SoRR). In addition, we define vectors

$$S_i(\mathbf{w}) := (s_{ij}(\mathbf{w}))_{j=1}^{N_-}$$

for $i = 1, \ldots, N_+$, where $s_{ij}(\mathbf{w}) := \ell(h_{\mathbf{w}}(\mathbf{x}_i^+) - h_{\mathbf{w}}(\mathbf{x}_j^-))$ for $i = 1, \ldots, N_+$ and $j = 1, \ldots, N_-$. Since ℓ is non-increasing, the jth largest coordinate of $S_i(\mathbf{w})$ is $\ell(h_{\mathbf{w}}(\mathbf{x}_i^+) - h_{\mathbf{w}}(\mathbf{x}_{[j]}^-))$. As a result, we have, for $i = 1, \ldots, N_+$,

$$\textstyle\sum_{j=m+1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}_i^+) - h_{\mathbf{w}}(\mathbf{x}_{[j]}^-)) = \phi_n(S_i(\mathbf{w})) - \phi_m(S_i(\mathbf{w})).$$

Hence, after dropping the normalizer, (3) can be equivalently written as

$$F^* = \min_{\mathbf{w}} \left\{ F(\mathbf{w}) := f^n(\mathbf{w}) - f^m(\mathbf{w}) \right\}, \tag{5}$$

where

$$f^l(\mathbf{w}) = \textstyle\sum_{i=1}^{N_+} \phi_l(S_i(\mathbf{w})) \quad \text{for } l = m, n. \tag{6}$$

Next, we introduce an interesting special case of (5), namely, the problem of minimizing SoRR loss. We still consider a supervised learning problem but the target $y \in \mathbb{R}$ does not need to be binary. We want to predict y based on a feature vector $\mathbf{x} \in \mathbb{R}^p$ using $h_{\mathbf{w}}(\mathbf{x})$. With a little abuse of notation, we measure the discrepancy between $h_{\mathbf{w}}(\mathbf{x})$ and y by $\ell(h_{\mathbf{w}}(\mathbf{x}), y)$, where $\ell : \mathbb{R}^2 \to \mathbb{R}_+$ is a loss function. We consider learning the model's parameter $\mathbf{w}$ from a training set $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^N$, where $\mathbf{x}_j \in \mathbb{R}^p$ and $y_j \in \mathbb{R}$ for $j = 1, \ldots, N$, by minimizing the SoRR loss. More specifically, we define vector

$$S(\mathbf{w}) = (s_j(\mathbf{w}))_{j=1}^N,$$

where $s_j(\mathbf{w}) := \ell(h_{\mathbf{w}}(\mathbf{x}_j), y_j),\ j = 1, \ldots, N$. Recall (4). For any integers m and n with $0 \le m < n \le N$, the problem of minimizing the SoRR loss with a range from $m+1$ to n is formulated as $\min_{\mathbf{w}} \{\phi_n(S(\mathbf{w})) - \phi_m(S(\mathbf{w}))\}$, which is an instance of (5) with

$$f^l = \phi_l(S(\mathbf{w})) \text{ for } l = m, n. \tag{7}$$

If we view $S_i(\mathbf{w})$ and $S(\mathbf{w})$ only as vector-value functions of $\mathbf{w}$ but ignore how they are formulated using data, (7) is a special case of (6) with $N_+ = 1$ and $N_- = N$.

4 Nearly Critical Point and Moreau Envelope Smoothing

We first develop a stochastic algorithm for (5) with f^l defined in (6). To do so, we make the following assumptions, which are satisfied by many smooth $h_{\mathbf{w}}$'s and ℓ's.

Assumption 1 *(a) $s_{ij}(\mathbf{w})$ is smooth and there exists $L \ge 0$ such that*[2] $\|\nabla s_{ij}(\mathbf{w}) - \nabla s_{ij}(\mathbf{v})\| \le L\|\mathbf{w} - \mathbf{v}\|$ *for any $\mathbf{w}, \mathbf{v} \in \mathbb{R}^d$, $i = 1, \ldots, N_+$ and $j = 1, \ldots, N_-$. (b) There exists $B \ge 0$ such that $\|\nabla s_{ij}(\mathbf{w})\| \le B$ for any $\mathbf{w} \in \mathbb{R}^d$, $i = 1, \ldots, N_+$ and $j = 1, \ldots, N_-$. (c) $F^* > -\infty$.*

[2] In this paper, $\|\cdot\|$ represents Euclidean norm.

Given $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, the subdifferential of f is

$$\partial f(\mathbf{w}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^d \,\middle|\, f(\mathbf{v}) \geq h(\mathbf{w}) + \boldsymbol{\xi}^\top(\mathbf{v} - \mathbf{w}) + o(\|\mathbf{v} - \mathbf{w}\|_2),\ \mathbf{v} \to \mathbf{w} \right\},$$

where each element in $\partial f(\mathbf{w})$ is called a subgradient of f at $\mathbf{w}$. We say f is **ρ-weakly convex** for some $\rho \geq 0$ if $f(\mathbf{v}) \geq f(\mathbf{w}) + \langle \boldsymbol{\xi}, \mathbf{v} - \mathbf{w} \rangle - \frac{\rho}{2}\|\mathbf{v} - \mathbf{w}\|^2$ for any $\mathbf{v}$ and $\mathbf{w}$ and $\boldsymbol{\xi} \in \partial f(\mathbf{w})$ and say f is **ρ-strongly convex** for some $\rho \geq 0$ if $f(\mathbf{v}) \geq f(\mathbf{w}) + \langle \boldsymbol{\xi}, \mathbf{v} - \mathbf{w} \rangle + \frac{\rho}{2}\|\mathbf{v} - \mathbf{w}\|^2$ for any $\mathbf{v}$ and $\mathbf{w}$ and $\boldsymbol{\xi} \in \partial f(\mathbf{w})$. It is known that, if f is ρ-weakly convex, then $f(\mathbf{w}) + \frac{1}{2\mu}\|\mathbf{w}\|^2$ is a $(\mu^{-1} - \rho)$-strongly convex function when $\mu^{-1} > \rho$.

Under Assumption 1, $\phi_l(S_i(\mathbf{w}))$ is a composite of the closed convex function ϕ_l and the smooth map $S_i(\mathbf{w})$. According to Lemma 4.2 in [14], we have the following lemma.

Lemma 1 *Under Assumption 1, $f^m(\mathbf{w})$ and $f^n(\mathbf{w})$ in (6) are ρ-weakly convex with $\rho := N_+ N_- L$.*

To solve (5) numerically, we need to overcome the following challenges. (i) $F(\mathbf{w})$ is non-convex even if each $s_{ij}(\mathbf{w})$ is convex. In fact, $F(\mathbf{w})$ is a DC function because, by Lemma 1, we can represent $F(\mathbf{w})$ as the difference of the convex functions $f^n(\mathbf{w}) + \frac{1}{2\mu}\|\mathbf{w}\|^2$ and $f^m(\mathbf{w}) + \frac{1}{2\mu}\|\mathbf{w}\|^2$ with $\mu^{-1} > \rho$. (ii) $F(\mathbf{w})$ is non-smooth due to ϕ_l so that finding an approximate critical point (defined below) of $F(\mathbf{w})$ is difficult. (iii) Computing the exact subgradient of $f^l(\mathbf{w})$ for $l = m, n$ requires processing $N_+ N_-$ data pairs, which is computationally expensive for a large data set.

Because of challenges (i) and (ii), we have to consider a reasonable goal when solving (5). We say $\mathbf{w} \in \mathbb{R}^d$ is a *critical point* of (5) if $\mathbf{0} \in \partial f^n(\mathbf{w}) - \partial f^m(\mathbf{w})$. Given $\epsilon > 0$, we say $\mathbf{w} \in \mathbb{R}^d$ is an *ϵ-critical point* of (5) if there exists $\boldsymbol{\xi} \in \partial f^n(\mathbf{w}) - \partial f^m(\mathbf{w})$ such that $\|\boldsymbol{\xi}\| \leq \epsilon$. A critical point can only be achieved asymptotically in general.[3] Within finitely many iterations, there also exists no algorithm that can find an ϵ-critical point unless at least one of f^m and f^n is smooth, e.g., [63]. Since f^m and f^n are both non-smooth, we have to consider a weaker but achievable target, which is a nearly ϵ-critical point defined below.

Definition 1 *Given $\epsilon > 0$, we say $\mathbf{w} \in \mathbb{R}^d$ is a nearly ϵ-critical point of (5) if there exist $\boldsymbol{\xi}$, $\mathbf{w}'$, and $\mathbf{w}'' \in \mathbb{R}^d$ such that $\boldsymbol{\xi} \in \partial f^n(\mathbf{w}') - \partial f^m(\mathbf{w}'')$ and $\max\{\|\boldsymbol{\xi}\|, \|\mathbf{w} - \mathbf{w}'\|, \|\mathbf{w} - \mathbf{w}''\|\} \leq \epsilon$.*

Definition 1 is reduced to the ϵ-stationary point defined by [39,52] when $\mathbf{w}$ equals $\mathbf{w}'$ or $\mathbf{w}''$. However, obtaining their ϵ-stationary point requires exactly solving the proximal mapping of f^m or f^n while finding a nearly ϵ-critical point requires only solving the proximal mapping inexactly. When $\mathbf{w}$ is generated by a stochastic algorithm, we also call $\mathbf{w}$ a nearly ϵ-critical point if it satisfies Definition 1 with each $\|\cdot\|$ replaced by $\mathbb{E}\|\cdot\|$.

Motivated by [52] and [39], we approximate non-smooth $F(\mathbf{w})$ by a smooth function using the Moreau envelopes. Given a proper, ρ-weakly convex and closed function f on $\mathbb{R}^d$, the *Moreau envelope* of f with the smoothing parameter $\mu \in (0, \rho^{-1})$ is defined as

$$f_\mu(\mathbf{w}) := \min_{\mathbf{v}} \left\{ f(\mathbf{v}) + \frac{1}{2\mu}\|\mathbf{v} - \mathbf{w}\|^2 \right\} \tag{8}$$

and the *proximal mapping* of f is defined as

$$\mathbf{v}_{\mu f}(\mathbf{w}) := \arg\min_{\mathbf{v}} \left\{ f(\mathbf{v}) + \frac{1}{2\mu}\|\mathbf{v} - \mathbf{w}\|^2 \right\}. \tag{9}$$

Note that the $\mathbf{v}_{\mu f}(\mathbf{w})$ is unique because the minimization above is strongly convex. Standard results show that $f_\mu(\mathbf{w})$ is smooth with $\nabla f_\mu(\mathbf{w}) = \mu^{-1}(\mathbf{w} - \mathbf{v}_{\mu f}(\mathbf{w}))$ and $\mathbf{v}_{\mu f}(\mathbf{w})$ is $(1 - \mu\rho)^{-1}$-Lipschitz continuous. See Proposition 13.37 in [48] and Proposition 1 in [52]. Hence, using the Moreau envelope, we can construct a smooth approximation of (5) as follows

$$\min_{\mathbf{w}} \left\{ F^\mu := f^n_\mu(\mathbf{w}) - f^m_\mu(\mathbf{w}) \right\}. \tag{10}$$

Function F^μ has the following properties. The first property is shown in [52]. We give the proof for the second in Appendix B.

[3] A stronger notion than criticality is (directional) stationarity, which can also be achieved asymptotically [45].

Lemma 2 *Suppose Assumption 1 holds and $\mu > \rho^{-1}$ with ρ defined in Lemma 1. The following claims hold*

1. $\nabla F^{\mu}(\mathbf{w}) = \mu^{-1}(\mathbf{v}_{\mu f^m}(\mathbf{w}) - \mathbf{v}_{\mu f^n}(\mathbf{w}))$ *and it is L_μ-Lipschitz continuous with* $L_\mu = \frac{2}{\mu - \mu^2\rho}$.
2. *If $\bar{\mathbf{v}}$ and $\mathbf{w}$ are two random vectors such that $\mathbb{E}\|\nabla F^{\mu}(\mathbf{w})\|^2 \leq \min\{1, \mu^{-2}\}\epsilon^2/4$ and $\mathbb{E}\|\bar{\mathbf{v}} - \mathbf{v}_{\mu f^l}(\mathbf{w})\|^2 \leq \epsilon^2/4$ for either $l = m$ or $l = n$, then $\bar{\mathbf{v}}$ is a nearly ϵ-critical points of* (5).

Since F^{μ} is smooth, we can directly apply a first-order method for smooth non-convex optimization to (10). To do so, we need to evaluate $\nabla F^{\mu}(\mathbf{w})$, which requires computing $\mathbf{v}_{\mu f^m}(\mathbf{w})$ and $\mathbf{v}_{\mu f^n}(\mathbf{w})$, i.e., exactly solving (9) with $f = f^m$ and $f = f^n$, respectively. Computing the subgradients of f^m and f^n require processing N_+N_- data pairs which is costly. Unfortunately, the standard approach of sampling over data pairs does not produce unbiased stochastic subgradients of f^m and f^n due to the composite structure $\phi_l(S_i(\mathbf{w}))$. In the next section, we will discuss a solution to overcome this challenge and approximate $\mathbf{v}_{\mu f^m}(\mathbf{w})$ and $\mathbf{v}_{\mu f^n}(\mathbf{w})$, which leads to an efficient algorithm for (10).

5 Algorithm for pAUC Optimization

Consider (10) with f^l defined in (6) for $l = m$ and n. To avoid of processing N_+N_- data points, one method is to introduce dual variables $\mathbf{p}_i = (p_{ij})_{j=1}^{N_-}$ for $i = 1, \ldots, N_+$ and formulate f^l as

$$f^l(\mathbf{w}) = \max_{\mathbf{p}_i \in \mathcal{P}^l, i=1,\ldots,N_+} \left\{ \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} p_{ij} s_{ij}(\mathbf{w}) \right\}, \tag{11}$$

where $\mathcal{P}^l = \{\mathbf{p} \in \mathbb{R}^{N_-} | \sum_{j=1}^{N_-} p_j = l,\ p_j \in [0,1]\}$. Then (10) can be reformulated as a min-max problem and solved by a primal-dual stochastic gradient method (e.g. [46]). However, the maximization in (11) involves N_+N_- decision variables and equality constraints, so the per-iteration cost is still $O(N_+N_-)$ even after using stochastic gradients.

To further reduce the per-iteration cost, we take the dual form of the maximization in (11) (see Lemma 4 in Appendix B) and formulate f^l as

$$f^l(\mathbf{w}) = \min_{\boldsymbol{\lambda}} \left\{ g^l(\mathbf{w}, \boldsymbol{\lambda}) := l\mathbf{1}^\top \boldsymbol{\lambda} + \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} [s_{ij}(\mathbf{w}) - \lambda_i]_+ \right\}, \tag{12}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{N_+})$. Hence, (9) with $f = f^l$ for $l = m$ and n can be reformulated as

$$\min_{\mathbf{v}, \boldsymbol{\lambda}} \left\{ g^l(\mathbf{v}, \boldsymbol{\lambda}) + \frac{1}{2\mu}\|\mathbf{v} - \mathbf{w}\|^2 \right\}. \tag{13}$$

Note that $g^l(\mathbf{v}, \boldsymbol{\lambda})$ is jointly convex in $\mathbf{v}$ and $\boldsymbol{\lambda}$ when $\mu^{-1} > \rho = N_+N_-L$ (see Lemma 3 in Appendix B). Thanks to formulation (13), we can construct stochastic subgradient of g^l and apply coordinate update to $\boldsymbol{\lambda}$ by sampling indexes i's and j's, which significantly reduce the computational cost when N_+ and N_- are both large. We present this standard stochastic block coordinate descent (SBCD) method for solving (13) in Algorithm 1 and present its convergence property as follows.

Proposition 1 *Suppose Assumption 1 holds and $\mu^{-1} > \rho = N_+N_-L$, $\theta_t = \frac{\text{dist}(\boldsymbol{\lambda}^{(0)}, \Lambda^*)}{\sqrt{ITN_-}}$ and $\eta_t = \frac{\|\mathbf{v}_{\mu f^l}(\mathbf{w}) - \mathbf{w}\|}{N_+N_-B\sqrt{T}}$ for any t in Algorithm 1. It holds that*

$$\left(\frac{1}{2\mu} - \frac{\rho}{2}\right)\mathbb{E}\|\bar{\mathbf{v}} - \mathbf{v}_{\mu f^l}(\mathbf{w}))\|^2 \leq \frac{N_+N_-}{\sqrt{IT}}\text{dist}(\boldsymbol{\lambda}^{(0)}, \Lambda^*) + \frac{N_+N_-B}{2\sqrt{T}}\|\mathbf{v}_{\mu f^l}(\mathbf{w}) - \mathbf{w}\| + \frac{\|\mathbf{v}_{\mu f^l}(\mathbf{w}) - \mathbf{w}\|^2}{2\mu T},$$

where $\Lambda^* = \arg\min_{\boldsymbol{\lambda}} g^l(\mathbf{v}_{\mu f^l}(\mathbf{w}), \boldsymbol{\lambda})$.

Using Algorithm 1 to compute an approximation of $\mathbf{v}_{\mu f^l}(\mathbf{w})$ for $l = m$ and n and thus, an approximation of $\nabla F^{\mu}(\mathbf{w})$, we can apply an approximate gradient descent (AGD) method to (10) and find a nearly ϵ-critical point of (5) according to Lemma 2. We present the AGD method in Algorithm 2 and its convergence property as follows.

Algorithm 1 Stochastic Block Coordinate Descent for (13): $(\bar{\mathbf{v}}, \bar{\boldsymbol{\lambda}})$ =SBCD$(\mathbf{w}, \boldsymbol{\lambda}, T, \mu, l)$

Input: Initial solution $(\mathbf{w}, \boldsymbol{\lambda})$, the number of iterations T, $\mu > 0$, an integer $l > 0$ and sample sizes I and J.
Set $(\mathbf{v}^{(0)}, \boldsymbol{\lambda}^{(0)}) = (\mathbf{w}, \boldsymbol{\lambda})$ and choose $(\eta_t, \theta_t)_{t=0}^{T-1}$.
for $t = 0$ **to** $T-1$ **do**
  Sample $\mathcal{I}_t \subset \{1, \ldots, N_+\}$ with $|\mathcal{I}_t| = I$ and sample $\mathcal{J}_t \subset \{1, \ldots, N_-\}$ with $|\mathcal{J}_t| = J$.
  Compute stochastic subgradient w.r.t. $\mathbf{v}$:

$$G_{\mathbf{v}}^{(t)} = \frac{N_+ N_-}{IJ} \sum_{i \in \mathcal{I}_t} \sum_{j \in \mathcal{J}_t} \nabla s_{ij}(\mathbf{v}^{(t)}) \mathbf{1}\left(s_{ij}(\mathbf{v}^{(t)}) > \lambda_i^{(t)}\right)$$

  Proximal stochastic subgradient update on $\mathbf{v}$:

$$\mathbf{v}^{(t+1)} = \arg\min_{\mathbf{v}} (G_{\mathbf{v}}^{(t)})^\top \mathbf{v} + \frac{\|\mathbf{v} - \mathbf{w}\|^2}{2\mu} + \frac{\|\mathbf{v} - \mathbf{v}^{(t)}\|^2}{2\eta_t} \tag{14}$$

  Compute stochastic subgradient w.r.t. λ_i for $i \in \mathcal{I}_t$:

$$G_{\lambda_i}^{(t)} = l - \frac{N_-}{J} \sum_{j \in \mathcal{J}_t} \mathbf{1}\left(s_{ij}(\mathbf{v}^{(t)}) > \lambda_i^{(t)}\right) \quad \text{for } i \in \mathcal{I}_t$$

  Stochastic block subgradient update on λ_i for $i \in \mathcal{I}_t$:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} - \theta_t G_{\lambda_i}^{(t)} \quad \text{for } i \in \mathcal{I}_t \quad \text{and} \quad \lambda_i^{(t+1)} = \lambda_i^{(t)} \quad \text{for } i \notin \mathcal{I}_t. \tag{15}$$

end for
Output: $(\bar{\mathbf{v}}, \bar{\boldsymbol{\lambda}}) = \frac{1}{T} \sum_{t=0}^{T-1} (\mathbf{v}^{(t)}, \boldsymbol{\lambda}^{(t)})$.

Algorithm 2 Approximate Gradient Descent (AGD) for (10)

Input: Initial solutions $(\mathbf{w}^{(0)}, \bar{\boldsymbol{\lambda}}_m^{(0)}, \bar{\boldsymbol{\lambda}}_n^{(0)})$, the number of iterations K, $\mu > \rho^{-1}$, $\gamma > 0$, $m = \alpha N_-$ and $n = \beta N_-$.
for $k = 0$ **to** $K-1$ **do**
  $(\bar{\mathbf{v}}_m^{(k)}, \bar{\boldsymbol{\lambda}}_m^{(k+1)})$ =SBMD$(\mathbf{w}^{(k)}, \bar{\boldsymbol{\lambda}}_m^{(k)}, T_k, \mu, m)$
  $(\bar{\mathbf{v}}_n^{(k)}, \bar{\boldsymbol{\lambda}}_n^{(k+1)})$ =SBMD$(\mathbf{w}^{(k)}, \bar{\boldsymbol{\lambda}}_n^{(k)}, T_k, \mu, n)$
  $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \gamma\mu^{-1}(\bar{\mathbf{v}}_m^{(k)} - \bar{\mathbf{v}}_n^{(k)})$
end for
Output: $\bar{\mathbf{v}}_n^{(\bar{k})}$ with $\bar{k}$ sampled from $\{0, \ldots, K-1\}$.

Theorem 1 *Suppose Assumption 1 holds and Algorithm 1 is called in iteration k of Algorithm 2 with parameters $\mu^{-1} > \rho = N_+ N_- L$, $\theta_t = \frac{dist(\bar{\boldsymbol{\lambda}}^{(k)}, \Lambda_k^*)}{\sqrt{I T_k N_-}}$, $\eta_t = \frac{\|\mathbf{v}_{\mu f^l}(\mathbf{w}^{(k)}) - \mathbf{w}^{(k)}\|}{N_+ N_- B \sqrt{T_k}}$ for any t, and*

$$T_k = \max\left\{\frac{144 N_+^2 N_-^2 D_l^2 (k+1)^2}{I(\mu^{-1} - \rho)^2}, \frac{4 N_+^2 N_-^2 \mu^2 l^2 B^2 (k+1)^2}{(\mu^{-1} - \rho)^2}, \frac{6\mu l^2 B^2 (k+1)}{2(\mu^{-1} - \rho)^2}\right\}$$

where $\Lambda_k^* = \arg\min_{\boldsymbol{\lambda}} g^l(\mathbf{v}_{\mu f^l}(\mathbf{w}^{(k)}), \boldsymbol{\lambda})$ and

$$D_l := \max\left\{dist(\bar{\boldsymbol{\lambda}}_l^{(0)}, \Lambda_0^*), \quad \frac{1}{2}\left(\frac{1}{\mu} - \rho\right) + \frac{\mu l^2 B^2}{2} + N_+ B + \frac{N_+ B}{1 - \mu\rho}\left(\frac{2\gamma}{\mu} + \gamma n B + \gamma m B\right)\right\}. \tag{16}$$

Then $\bar{\mathbf{v}}_n^{(\bar{k})}$ is a nearly ϵ-critical point of (5) with f^l defined in (6) with K no more than

$$K = \max\left\{\frac{16\mu^2}{\gamma \min\{1, \mu^2\}\epsilon^2}\left(F(\mathbf{v}_{\mu f^n}(\mathbf{w}^{(0)})) - F^*\right) \frac{96}{\min\{1, \mu^2\}\epsilon^2} \log\left(\frac{96}{\min\{1, \mu^2\}\epsilon^2}\right)\right\}. \tag{17}$$

According to Theorem 1. To find a nearly ϵ-critical point of (5), we need $K = \tilde{O}(\epsilon^{-2})$ iterations in Algorithm 2 and $\sum_{k=0}^{K-1} T_k = O(K^3) = \tilde{O}(\epsilon^{-6})$ iterations of Algorithm 1 in total across all calls.

The technique in the previous sections can be directly applied to minimize the SoRR loss, which is formulated as (5) but with f^l defined in (7). Due to the limit of space, we present the algorithm for minimizing the SoRR loss and its convergence result in Appendix A.

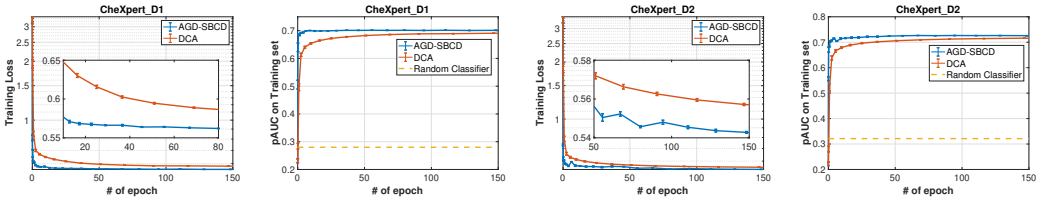

Figure 1: Results for Patial AUC Maximization of D1 and D2. (Results of D3, D4 and D5 are shown in Appendix E.2 Figure 3)

6 Numerical Experiments

In this section, we demonstrate the effectiveness of our algorithm AGD-SBCD for pAUC maximization and SoRR loss minimization problems. All experiments are conducted in Python and Matlab on a computer with the CPU 2GHz Quad-Core Intel Core i5 and the GPU NVIDIA GeForce RTX 2080 Ti. All datasets we used are publicly available and contain no personally identifiable information and offensive contents.

6.1 Partial AUC Maximization

For maximizing pAUC, we focus on large-scale imbalanced medical dataset CheXpert [25], which is licensed under CC-BY-SA and has 224,316 images. We construct five binary classification tasks with the logistic loss $\ell(z) = \log(1 + \exp(-z))$ for predicting five popular diseases, Cardiomegaly (D1), Edema (D2), Consolidation (D3), Atelectasis (D4), and P. Effusion (D5).

For comparison of training convergence, we consider different methods for optimizing the partial AUC. We compare with two baselines, DCA [24] (see Appendix E.3 for details) and SVM$_{pAUC}$-tight [42]. Since DCA and SVM$_{pAUC}$-tight cannot be applied to deep neural networks, we focus on linear model and use a pre-trained deep neural network to extract a fixed dimensional feature vectors of 1024. The deep neural network was trained by optimizing the cross-entropy loss following the same setting as in [66].

For AGD-SBCD, we fix the $I = J = 100$, $m = 10^4$ and $n = 10^5$. In the spirit of Theorem 1, within Algorithm 3 called in the kth main iteration, η_t is set to $\frac{c}{(k+1)N^+N^-}$, θ_t is set to $\frac{c}{(k+1)N^-}$ for any t with c tuned in $\{0.1, 1, 5, 10, 20, 30\}$ and T_k is set to $50(k+1)^2$. Parameters μ is chosen from $\{10^2, 10^3\}/(N_+N_-)$ and γ is tuned from $\{2 \times 10^3, 5 \times 10^3\}/(N_+N_-)$. For DCA, we only implement the setting of DCA.Changing.lr (see Appendix E.1 for descriptions). In Algorithm 4 called in the kth main iteration, η_t and θ_t are selected following the same process as in AGD-SBCD and T is set to $C(k+1)^2$ with C chosen from $\{100, 200, 500\}$. For the SVM$_{pAUC}$-tight method, we use their MATLAB code in [42] and tune hyper-parameter C from $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$. In Figure 1 and Appendix E.2 Figure 3, we show how the training loss (the objective value of (3)) and normalized partial AUC on the training data change with the number of epochs. We observe that for all of these five diseases, our algorithm converges much faster than DCA and we get a better partial AUC than DCA.

The comparison between our AGD-SBCD and SVM$_{pAUC}$-tight on training data are shown in Table 1. As shown from the second to the fifth column of Table 1, our algorithm needs only a few seconds to exceed the pAUCs that SVM$_{pAUC}$-tight takes more than one minute to return. As shown from sixth to eighth column, our algorithm eventually improves the pAUC by at least 12% compared with SVM$_{pAUC}$-tight. DCA is not included in the tables because it computes deterministic subgradients, which leads to a runtime significantly longer than the other two methods.

To compare the testing performances, we consider the deep neural networks besides the linear model. For linear model, we still compare with DCA and SVM$_{pAUC}$-tight. For deep neural networks, we compare with the naive mini-batch based method (MB) [27] and methods based on different optimization objectives, including the cross-entropy loss (CE) and the AUC min-max margin loss (AUC-M) [67]. We learn the model DenseNet121 from scratch with the CheXpert training data split in train/val=9:1 and the CheXpert validation dataset as the testing set, which has 234 samples. The

Table 1: Comparison on the CheXpert training data. From left to right, the columns are the tasks, the pAUCs returned by SVM$_{pAUC}$-tight, the CPU time (in seconds) SVM$_{pAUC}$-tight takes, the CPU and GPU time AGD-SBCD uses to exceed SVM$_{pAUC}$-tight's pAUCs, the final pAUCs returned by AGD-SBCD, and the CPU and GPU time (in seconds) AGD-SBCD takes to return the final pAUCs.

Methods	SVM$_{pAUC}$-tight		AGD-SBCD				
Tasks	pAUC	CPU time	CPU time to outperform	GPU time to outperform	pAUC	CPU time	GPU time
D1	0.6259	95.14	2.91	1.85	0.7005±0.0003	118.32	82.13
D2	0.5860	90.83	3.36	1.93	0.7214±0.0024	415.66	247.29
D3	0.3745	90.56	3.26	1.84	0.4910±0.0006	181.70	104.55
D4	0.3895	89.64	10.09	8.38	0.4616±0.0006	187.36	158.14
D5	0.7267	90.86	3.97	1.89	0.8272±0.0001	238.10	142.91

range of FPRs in pAUC is [0.05, 0.5]. For optimizing CE, we use the standard Adam optimizer. For optimizing AUC-M, we use the PESG optimizer in [67]. We run each method 10 epochs and the learning rate (c in AGD-SBCD) of all methods is tuned from $\{10^{-5} \sim 10^{0}\}$. The mini-batch size is 32. For AGD-SBCD, T_k is set to $50(k+1)^2$, μ is set to $\frac{10^3}{N_+N_-}$ and γ is tuned from $\{0.1, 1, 2\} \times 10^3/(N_+N_-)$. For MB, the learning rate decays in the same way as in [27]. For CE and AUC-M, the learning rate decays 10-fold after every 5 epochs. For AUC-M, we tune the hyperparameter γ in {100, 500, 1000}. For each method, the validation set is used to tune the hyperparameters and select the best model across all iterations. The results of the pAUCs on the testing set are reported in Table 2, which shows that our algorithm performs the best for all diseases. The complete ROC curves on the testing set are shown in Appendix E.2.

Table 2: The pAUCs with FPRs between 0.05 and 0.5 on the testing sets from the CheXpert data.

		D1	D2	D3	D4	D5
Linear Model	SVM$_{pAUC}$-tight	0.6538±0.0042	0.6038±0.0009	0.6946±0.0020	**0.6521±0.0006**	0.7994±0.0004
	DCA	0.6636±0.0093	0.8078±0.0030	0.7427±0.0257	0.6169±0.0208	0.8371±0.0022
	AGD-SBCD	**0.6721±0.0081**	**0.8257±0.0025**	**0.8016±0.0075**	0.6340±0.0165	**0.8500±0.0017**
Deep Model	MB	0.7510±0.0248	0.8197±0.0127	0.6339±0.0328	0.5698±0.0343	0.8461±0.0188
	CE	0.6994±0.0453	0.8075±0.0244	0.7673±0.0266	0.6499±0.0184	0.7884±0.0080
	AUC-M	0.7403±0.0339	0.8002±0.0274	0.8533±0.0469	0.7420±0.0277	0.8504±0.0065
	AGD-SBCD	**0.7535±0.0255**	**0.8345±0.0130**	**0.8689±0.0184**	**0.7520±0.0079**	**0.8513±0.0107**

6.2 SoRR Loss Minimization

For SoRR loss minimization, we compare with the baseline DCA. We focus on learning a linear model $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with four benchmark datasets from the UCI [15] data repository preprocessed by Libsvm [9]: a9a, w8a, ijcnn1 and covtype. The statistics of these datasets are summarized in Table 3 in Appendix E.1. We use logistic loss $l(h_{\mathbf{w}}(\mathbf{x}), y) = -y\log(h_{\mathbf{w}}(\mathbf{x})) - (1-y)\log(1-h_{\mathbf{w}}(\mathbf{x}))$ where label $y \in \{0, 1\}$. Due to the limit space, we present the process of tuning hyperparameters and the convergence curves (Figure2) in Appendix E.1. From the curves, we notice that our algorithm reduce SoRR loss faster than DCAs for all of these four datasets.

7 Conclusion

Most existing methods for optimizing pAUC are deterministic and only have an asymptotic convergence property. We formulate pAUC optimization as a non-smooth DC program and develop a stochastic subgradient method based on the Moreau envelope smoothing technique. We show that our method finds a nearly ϵ-critical point in $\tilde{O}(\epsilon^{-6})$ iterations and demonstrate its performance numerically. A limitation of this paper is the smoothness assumption on $s_{ij}(\mathbf{w})$, which does not hold for some models, e.g., neural networks using ReLU activation functions. It is a future work to extend our results for non-smooth models.

References

[1] Hadi Abbaszadehpeivasti, Etienne de Klerk, and Moslem Zamani. On the rate of convergence of the difference-of-convex algorithm (dca). *arXiv preprint arXiv:2109.13566*, 2021. 3

[2] AD Alexandroff. Surfaces represented by the difference of convex functions. In *Doklady Akademii Nauk SSSR (NS)*, volume 72, pages 613–616, 1950. 3

[3] Le Thi Hoai An and Pham Dinh Tao. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of operations research*, 133(1):23–46, 2005. 2, 3

[4] Le Thi Hoai An, Huynh Van Ngai, Pham Dinh Tao, and Luu Hoang Phuc Hau. Stochastic difference-of-convex algorithms for solving nonconvex optimization problems. *arXiv preprint arXiv:1911.04334*, 2019. 3

[5] Nguyen Thai An and Nguyen Mau Nam. Convergence analysis of a proximal point algorithm for minimizing differences of functions. *Optimization*, 66(1):129–147, 2017. 3

[6] Francisco J Aragón Artacho, Ronan MT Fleming, and Phan T Vuong. Accelerating the dc algorithm for smooth functions. *Mathematical Programming*, 169(1):95–118, 2018. 3

[7] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997. 1

[8] Andrew P Bradley. Half-auc for the evaluation of sensitive or specific classifiers. *Pattern Recognition Letters*, 38:93–98, 2014. 1

[9] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011. 9, 13, 21

[10] Damek Davis and Dmitriy Drusvyatskiy. Stochastic subgradient method converges at the rate $o(k^{-1/4})$ on weakly convex functions. *arXiv preprint arXiv:1802.02988*, 2018. 2

[11] Damek Davis and Benjamin Grimmer. Proximally guided stochastic subgradient method for nonsmooth, nonconvex problems. *SIAM Journal on Optimization*, 29(3):1908–1930, 2019. 2

[12] Qi Deng and Chenghao Lan. Efficiency of coordinate descent methods for structured nonconvex optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 74–89. Springer, 2020. 3

[13] Lori E Dodd and Margaret S Pepe. Partial auc estimation and regression. *Biometrics*, 59(3):614–623, 2003. 1, 2

[14] Dmitriy Drusvyatskiy and Courtney Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178(1):503–558, 2019. 5, 15

[15] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. 9, 13, 21

[16] Rachid Ellaia. *Contribution à l'analyse et l'optimisation de différence de fonctions convexes*. PhD thesis, Université Paul Sabatier, 1984. 3

[17] Yanbo Fan, Siwei Lyu, Yiming Ying, and Bao-Gang Hu. Learning with average top-k loss. *arXiv preprint arXiv:1705.08826*, 2017. 2

[18] D. Gabay. Minimizing the difference of two convex functions. I. Algorithms based on exact regularization. 1982. 3

[19] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. 1

[20] Philip Hartman. On functions representable as a difference of convex functions. *Pacific Journal of Mathematics*, 9(3):707–713, 1959. 3

[21] Lulu He, Jimin Ye, et al. Accelerated proximal stochastic variance reduction for dc optimization. *Neural Computing and Applications*, 33(20):13163–13181, 2021. 3

[22] J-B Hiriart-Urruty. Generalized differentiability/duality and optimization for problems dealing with differences of convex functions. In *Convexity and duality in optimization*, pages 37–70. Springer, 1985. 3

[23] J-B Hiriart-Urruty. How to regularize a difference of convex functions. *Journal of mathematical analysis and applications*, 162(1):196–209, 1991. 3

[24] Shu Hu, Yiming Ying, Siwei Lyu, et al. Learning by minimizing the sum of ranked range. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 8, 21, 22

[25] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison, 2019. 8, 13

[26] Yulei Jiang, Charles E Metz, and Robert M Nishikawa. A receiver operating characteristic partial area index for highly sensitive diagnostic tests. *Radiology*, 201(3):745–750, 1996. 2

[27] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Online and stochastic gradient methods for non-decomposable loss functions. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 8, 9

[28] Koulik Khamaru and Martin Wainwright. Convergence guarantees for a class of non-convex and non-smooth optimization problems. In *International Conference on Machine Learning*, pages 2601–2610. PMLR, 2018. 3

[29] Osamu Komori and Shinto Eguchi. A boosting method for maximizing the partial area under the roc curve. *BMC bioinformatics*, 11(1):1–17, 2010. 2

[30] Hoai An Le Thi and Tao Pham Dinh. Dc programming and dca: thirty years of developments. *Mathematical Programming*, 169(1):5–68, 2018. 2, 3

[31] Hoai An Le Thi, Hoai Minh Le, Duy Nhat Phan, and Bach Tran. Stochastic dca for the large-sum of non-convex functions problem and its application to group variable selection in classification. In *International Conference on Machine Learning*, pages 3394–3403. PMLR, 2017. 3

[32] Thomas Lipp and Stephen Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016. 3

[33] Mingrui Liu, Hassan Rafique, Qihang Lin, and Tianbao Yang. First-order convergence theory for weakly-convex-weakly-concave min-max problems. *Journal of Machine Learning Research*, 22(169):1–34, 2021. 2

[34] Hua Ma, Andriy I Bandos, Howard E Rockette, and David Gur. On use of partial area under the roc curve for evaluation of diagnostic performance. *Statistics in medicine*, 32(20):3449–3458, 2013. 1

[35] Runchao Ma, Qihang Lin, and Tianbao Yang. Quadratically regularized subgradient methods for weakly convex optimization with weakly convex constraints. In *International Conference on Machine Learning*, pages 6554–6564. PMLR, 2020. 2

[36] Julien Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. *arXiv preprint arXiv:1306.4650*, 2013. 3

[37] Donna Katzman McClish. Analyzing a portion of the roc curve. *Medical decision making*, 9(3):190–195, 1989. 2

[38] Abdellatif Moudafi. On the difference of two maximal monotone operators: Regularization and algorithmic approaches. *Applied mathematics and computation*, 202(2):446–452, 2008. 3

[39] Abdellatif Moudafi. A complete smooth regularization of dc optimization problems. 2021. 3, 5

[40] Abdellatif Moudafi and Paul-Emile Maingé. On the convergence of an approximate proximal method for dc functions. *Journal of computational Mathematics*, pages 475–480, 2006. 3

[41] Harikrishna Narasimhan and Shivani Agarwal. A structural svm based approach for optimizing partial auc. In *International Conference on Machine Learning*, pages 516–524. PMLR, 2013. 2

[42] Harikrishna Narasimhan and Shivani Agarwal. $\text{SVM}_{\text{pAUC}}^{\text{tight}}$: a new support vector method for optimizing partial auc based on a tight convex upper bound. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 167–175, 2013. 2, 8

[43] Harikrishna Narasimhan and Shivani Agarwal. Support vector algorithms for optimizing the partial area under the roc curve. *Neural computation*, 29(7):1919–1963, 2017. 2

[44] Atsushi Nitanda and Taiji Suzuki. Stochastic difference of convex algorithm and its application to training deep boltzmann machines. In *Artificial intelligence and statistics*, pages 470–478. PMLR, 2017. 3

[45] Jong-Shi Pang, Meisam Razaviyayn, and Alberth Alvarado. Computing b-stationary points of nonsmooth dc programs. *Mathematics of Operations Research*, 42(1):95–118, 2017. 3, 5

[46] Hassan Rafique, Mingrui Liu, Qihang Lin, and Tianbao Yang. Weakly-convex–concave min–max optimization: provable algorithms and applications in machine learning. *Optimization Methods and Software*, pages 1–35, 2021. 2, 6

[47] Maria Teresa Ricamato and Francesco Tortorella. Partial auc maximization in a linear combination of dichotomizers. *Pattern Recognition*, 44(10-11):2669–2677, 2011. 3

[48] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009. 5

[49] Shai Shalev-Shwartz and Yonatan Wexler. Minimizing the maximal loss: How and why. In *International Conference on Machine Learning*, pages 793–801. PMLR, 2016. 2

[50] João Carlos O Souza, Paulo Roberto Oliveira, and Antoine Soubeyran. Global convergence of a proximal linearized algorithm for difference of convex functions. *Optimization Letters*, 10(7):1529–1539, 2016. 3

[51] Bharath K Sriperumbudur and Gert RG Lanckriet. On the convergence of the concave-convex procedure. In *Nips*, volume 9, pages 1759–1767. Citeseer, 2009. 3

[52] Kaizhao Sun and Xu Andy Sun. Algorithms for difference-of-convex (dc) programs based on difference-of-moreau-envelopes smoothing. *arXiv preprint arXiv:2104.01470*, 2021. 2, 3, 5, 14, 21

[53] Wen-yu Sun, Raimundo JB Sampaio, and MAB Candido. Proximal point algorithm for minimization of dc function. *Journal of computational Mathematics*, pages 451–462, 2003. 3

[54] Pham Dinh Tao and Le Thi Hoai An. Convex analysis approach to dc programming: theory, algorithms and applications. *Acta mathematica vietnamica*, 22(1):289–355, 1997. 2, 3

[55] Pham Dinh Tao and Le Thi Hoai An. A dc optimization algorithm for solving the trust-region subproblem. *SIAM Journal on Optimization*, 8(2):476–505, 1998. 3

[56] Hoai An Le Thi, Hoang Phuc Hau Luu, and Tao Pham Dinh. Online stochastic dca with applications to principal component analysis. *arXiv preprint arXiv:2108.02300*, 2021. 3

[57] Mary Lou Thompson and Walter Zucchini. On the statistical analysis of roc curves. *Statistics in medicine*, 8(10):1277–1290, 1989. 2

[58] Hoang Tuy. Dc optimization: theory, methods and algorithms. In *Handbook of global optimization*, pages 149–216. Springer, 1995. 3

[59] Naonori Ueda and Akinori Fujino. Partial auc maximization via nonlinear scoring functions. *arXiv preprint arXiv:1806.04838*, 2018. 2

[60] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992. 2

[61] Zhanfeng Wang and Yuan-Chin Ivan Chang. Marker selection via maximizing the partial area under the roc curve of linear risk scores. *Biostatistics*, 12(2):369–385, 2011. 3

[62] Bo Wen, Xiaojun Chen, and Ting Kei Pong. A proximal difference-of-convex algorithm with extrapolation. *Computational optimization and applications*, 69(2):297–324, 2018. 3

[63] Yi Xu, Qi Qi, Qihang Lin, Rong Jin, and Tianbao Yang. Stochastic optimization for dc functions and non-smooth non-convex regularizers with non-asymptotic convergence. In *International Conference on Machine Learning*, pages 6942–6951. PMLR, 2019. 3, 5

[64] Hanfang Yang, Kun Lu, Xiang Lyu, and Feifang Hu. Two-way partial auc and its properties. *Statistical methods in medical research*, 28(1):184–195, 2019. 1, 3

[65] Zhiyong Yang, Qianqian Xu, Shilong Bao, Yuan He, Xiaochun Cao, and Qingming Huang. When all we need is a piece of the pie: A generic framework for optimizing two-way partial auc. In *International Conference on Machine Learning*, pages 11820–11829. PMLR, 2021. 2

[66] Zhuoning Yuan, Yan Yan, Milan Sonka, and Tianbao Yang. Robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification. *arXiv preprint arXiv:2012.03173*, 2020. 8

[67] Zhuoning Yuan, Yan Yan, Milan Sonka, and Tianbao Yang. Large-scale robust deep AUC maximization: A new surrogate loss and empirical studies on medical image classification. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 3020–3029. IEEE, 2021. 8, 9

[68] Alan L Yuille and Anand Rangarajan. The concave-convex procedure. *Neural computation*, 15(4):915–936, 2003. 3

[69] Dixian Zhu, Gang Li, Bokun Wang, Xiaodong Wu, and Tianbao Yang. When auc meets dro: Optimizing partial auc for deep learning with non-convex convergence guarantee. *arXiv preprint*, 2022. 3

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Algorithm 1 and 2 and Section 6.
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We are not aware of any negative societal impacts of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 6.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 6.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 6.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 6.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite them as [9, 15, 25]
   (b) Did you mention the license of the assets? [Yes] See Section 6.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We only use publicly available data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 6.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]