A Univariate Bound of Area Under ROC

Siwei Lyu **Yiming Ying**
Computer Science Department Mathematics Department
University at Albany, State University at New York, USA
{slyu,yying}@albany.edu

Abstract

Area under ROC (AUC) is an important metric for binary classification and bipartite ranking problems. However, it is difficult to directly optimize AUC as a learning objective, so most existing algorithms are based on optimizing a surrogate loss to AUC. One significant drawback of these surrogate losses is that they require pairwise comparisons among training data, which leads to slow running time and increasing local storage for online learning. In this work, we describe a new surrogate loss based on a reformulation of AUC risk, which does not require pairwise comparison but rankings of the predictions. We further show that the ranking operation can be avoided, and the learning objective obtained based on this surrogate enjoys linear complexity in time and storage. We perform experiments to demonstrate the effectiveness of the online and batch algorithms for AUC optimization based on the proposed surrogate loss.

1 INTRODUCTION

The *area under receiver operating characteristics curves* (AUC) is a useful quantitative metric for assessing the performance of binary classification and bipartite ranking algorithms [1, 2]. However, there are two factors make AUC difficult to be used directly as a learning objective to train classification or ranking algorithms. The foremost is due to the discontinuous indicator function in the definition of the AUC (c.f. Eq.(1)), which makes direct minimization of the AUC in general an NP-hard problem [4]. As such, most existing AUC learning algorithm replace the indicator function with surrogates that are continuous and convex upper-bounds of the AUC. The second issue with the AUC is the requirement of pairwise comparison between all positive and negative examples in training data. This leads to algorithms with a running time complexity that is quadratic in the number of training data, and a space complexity that is linear of the training data. For batch algorithms, this means slow running time as we need to compare all pairs of positive/negative examples, and for online learning, this means ever increasing local storage as we need to store all previously seen data for the pairwise comparisons. Both are undesirable when applying these algorithms to large-scale datasets.

In this work, we describe a new surrogate loss to AUC that has a linear time complexity and constant space complexity. This new loss is based on an equivalent formulation of AUC based on ranking the prediction scores, which obviates pairwise comparisons. We further show that the ranking operation can be replaced with an equivalent optimization problem, and the learning objective affords a simple form that has a bounding relation with AUC. Furthermore, we show that the new loss has a close relation with the SVM learning objective, which sheds light on the previous observations of the effectiveness of the SVM on optimizing AUC [5, 6, 7, 8]. The new surrogate loss leads naturally to an online AUC optimization method with simple (projected) stochastic sub-gradient steps. Experimental evaluations on several standard benchmark datasets show that learning objective formed from this new loss achieves performance in par with other widely used AUC surrogates, with a significant reduction in running time and storage requirement.

2 DEFINITIONS

To facilitate subsequent description, we first review the definition of AUC in the context of binary classification. Assume we are given a set of data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, with $y_i \in \{-1, +1\}$ and $\mathbf{x}_i \in \mathcal{R}^d$. We denote $\mathcal{I}^+ = \{i | y_i =$

$+1\}$ and $\mathcal{I}^- = \{i|y_i = -1\}$ as the sets of indices of positive and negative examples, respectively, with $N^+ = |\mathcal{I}^+|$ and $N^- = |\mathcal{I}^-|$, and $N^+ + N^- = N$. Define $\mathbf{I}$ as the *indicator function*: $\mathbf{I}_a = 1$ if a is true and 0 otherwise. A parametric binary classifier $c_{\mathbf{w},\theta} : \mathcal{R}^d \mapsto \{-1, +1\}$, constructed as

$$c_{\mathbf{w},\theta}(\mathbf{x}) = 2\mathbf{I}_{f_{\mathbf{w}}(\mathbf{x})\geq\theta} - 1 = \text{sign}(f_{\mathbf{w}}(\mathbf{x}) - \theta),$$

maps an example to the class label, where $f_{\mathbf{w}} : \mathcal{R}^d \mapsto \mathcal{R}$ (with $\mathbf{w} \in \mathcal{R}^m$ being the parameter) is the prediction function and $\theta \in \mathcal{R}$ is the classification threshold. We denote $c_i = f_{\mathbf{w}}(\mathbf{x}_i)$ as the *prediction score* of the i$^{\text{th}}$ example ($i = 1, \cdots, N$). For simplicity, we assume there are no ties in the prediction scores, *i.e.*, $c_i \neq c_j$ for $i \neq j$, though this condition will be relaxed later.

Given a threshold θ, negative examples with prediction scores greater than θ are false positives, and the false positive rate is given by $\tau_{FP} = |\mathbf{I}_{c_i>\theta \wedge i\in\mathcal{I}^-}|/N^-$. Correspondingly, positive examples with prediction scores greater or equal to θ are true positives, and the true positive rate is given by $\tau_{TP} = |\mathbf{I}_{c_i\geq\theta \wedge i\in\mathcal{I}^+}|/N^+$. Then the receiver operation curve (ROC) is defined as the curve formed by the pair (τ_{FP}, τ_{TP}) with $\theta \in (-\infty, \infty)$. With this definition, ROC is a curve confined to $[0, 1] \times [0, 1]$ and connecting $(0, 0)$ to $(1, 1)$. AUC then corresponds to the area enclosed by the ROC curve of the classifier.

It is more conveniently computed in closed form using the Wilcoxon-Mann-Whitney (WMW) statistic [3], as $A = \frac{1}{N^+N^-}\sum_{i\in\mathcal{I}^+}\sum_{j\in\mathcal{I}^-}\mathbf{I}_{c_i>c_j}$. In this work, we use the *AUC risk*, which is defined as

$$L_{\text{AUC}} = 1 - A = \frac{1}{N^+N^-}\sum_{i\in\mathcal{I}^+}\sum_{j\in\mathcal{I}^-}\mathbf{I}_{c_i<c_j}. \quad (1)$$

Note that L_{AUC} takes values in $[0, 1]$ and corresponds to the fraction of pairs of positive and negative predictions that are ranked incorrectly, *i.e.*, a positive example with lower prediction score than a negative example, so $L_{\text{AUC}} = 0$ indicates perfect classification/ranking. In addition, L_{AUC} is independent of threshold θ, and only concerns with the overall performance of the predictor $f_{\mathbf{w}}$. Hence, we aim to learn a prediction function $f_{\mathbf{w}}$ that minimizes L_{AUC}, from which we can choose θ to construct classifier $c_{\mathbf{w},\theta}(\mathbf{x})$.

3 RELATED WORKS

Most existing works for either batch or online algorithms for AUC optimization (*e.g.*, [9, 10]) minimize surrogates to the true AUC risk, which are usually in the form of convex upper-bounds to the indicator function in Eq.(1). Specifically, denoting the prediction scores for $\mathbf{x}_i$ and $\mathbf{x}_j$ as c_i and c_j, respectively, the surrogate loss function takes the form as $\frac{1}{N^+N^-}\sum_{i\in\mathcal{I}^+}\sum_{j\in\mathcal{I}^-}\ell(c_i - c_j)$, and common choices for ℓ include

1. the hinge loss [10], $\ell_h(c_i, c_j) = [1 - (c_i - c_j)]_+$, where $[a]_+ = \max\{0, a\}$ is the hinge function,
2. the squared hinge loss [11, 9], $\ell_{sh}(c_i, c_j) = [1 - (c_i - c_j)]_+^2$,
3. the logistic loss, $\ell_{lg} = \log_2(1 + e^{c_i - c_j})$,
4. and the rank-boost loss [12], $\ell_e(c_i, c_j) = e^{c_i - c_j}$.

All these surrogates are nonnegative, monotonic decreasing and satisfy $\ell(c_i, c_j) = 1$ when $c_i = c_j$. One significant problem with these surrogates is that they all rely on pairwise comparisons between positive and negative training examples, which lead to algorithms with quadratic running time complexity. For large datasets, such quadratic running time will significantly slow down the training process, and the pairwise comparisons prohibit efficient online learning algorithms for AUC optimization.

One exception is the work of [11], which shows that the squared hinge surrogate of AUC risk, $\ell_{sh}(c_i, c_j)$, affords an equivalent saddle point reformulation. An online stochastic gradient descent method is then developed based on this reformulation that has complexities $O(N)$ in time and $O(1)$ in space. However, there are two issues of this method that this work aims to improve on. First, the original surrogate loss still requires pairwise comparison, and to decouple them, one needs to introduce auxiliary variables for a saddle point reformulation. In contrast, our surrogate loss obviates pairwise comparison all together. Second, our surrogate loss reduces to a minimization problem, which is easier to analyze and implement than the saddle point reformulation of [11].

In parallel with methods directly optimizing AUC, empirical observations suggest that learning objectives not designed for AUC optimization (*e.g.*, SVM or boosting) can achieve low AUC risk [5, 6, 7, 8]. For instance, in [6], a generalized SVM approach was developed that is able to optimize multivariate non-linear performance measures in polynomial time, including AUC. However, when assessed with respect to the AUC, the superiority of the direct AUC optimization approach over standard SVMs seemed less convincing. The work of [7] many performance measures for binary classification are compared experimentally, and it was found that maximum margin methods such as boosting and SVMs yield excellent performance when measured with AUC. In [5] it was shown that optimizing standard SVMs leads to maximizing the AUC in the special (trivial) case when the given data is separable. As a perfect separation implies a zero AUC risk. The work [13] uses the rank-equivalent

definition of AUC to derive a hinge rank loss and shows that it is analogous to the SVM objective. However, no explicit relation between the SVM objective and AUC or AUC surrogates are established in previous works.

Further along this line, several studies have provided results on the consistency of the univariate losses to AUC risk, *i.e.*, in the asymptotic sense, minimizing the univariate losses under certain conditions may also lead to the minimization of AUC risk [14, 15], and a similar analysis is conducted for binary surrogate losses to AUC risk in [16]. These analyses show that univariate losses such as the ℓ_2, squared hinge and exponential losses are consistent with AUC risk, yet the widely used hinge loss in SVM are inconsistent. This seems to put in question whether minimizing AUC risk based on pairwise comparisons is really warranted. However, these studies are still of limited in practice due to several reasons. First, they can not explain the observation that the SVM algorithm which is based on the hinge loss, oftentimes leads to good performance when evaluated with AUC risk, though it is not theoretically consistent with AUC risk. In addition, these analysis does not reveal a direct relation between the univariate losses and AUC risk, and it is more illustrative if some bounding relation between them can be revealed. Furthermore, these analyses may not be as relevant in practice, as the learning objective in actual algorithms is usually combined with extra terms such as the regularizers.

4 METHOD

In this section, we start with an equivalent definition of AUC risk, which does not require pairwise comparisons of positive and negative examples. From this equivalent definition, we establish our AUC surrogate loss and its equivalent form for efficient optimization.

4.1 AUC Risk Without Pairwise Comparison

Besides the WMW statistics, Eq.(1), there exists another equivalent formulation of AUC risk (and AUC itself), which depends on the ranking of the prediction scores instead of all pairwise comparisons of the prediction scores of the positive and negative examples [4, 13]. To explain this equivalent form of AUC risk, we first introduce several additional notations. For simplicity, we assume there are no ties in the prediction scores, *i.e.*, $c_i \neq c_j$ for $i \neq j$, though this condition will be relaxed later.

We denote $(c_1^{\uparrow}, \cdots, c_N^{\uparrow})$ as the result of sorting $(c_1, \cdots, c_N)$ in ascending order, *i.e.*, $c_1^{\uparrow} < c_2^{\uparrow} < \cdots < c_N^{\uparrow}$. Moreover, let $r_i^+ \in \{1, \cdots, N\}$ $(i = 1, \cdots, N^+)$ be the *rank* of the i[th] positive example encountered in the ordered list $(c_1^{\uparrow}, \cdots, c_N^{\uparrow})$ starting from the beginning. With a slight abuse of notation, let $c_i^{\uparrow +}$ be the corresponding value of the i[th] positive example in the ordered list $(c_1^{\uparrow}, \cdots, c_N^{\uparrow})$, *i.e.*, $c_i^{\uparrow +} = c_{r_i^+}^{\uparrow}$. An example illustrating these definitions is given in Fig.1. These definitions immediately lead to the following simple result that will be important subsequently.

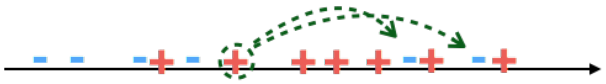

Figure 1: *An illustration of the ranking definition of the AUC. Note that in this case, we have* $N^+ = 7$, $N^- = 6$, *and* $(r_1^+, r_2^+, r_3^+, r_4^+, r_5^+, r_6^+, r_7^+) = (4, 6, 7, 8, 9, 11, 13)$. *For the positive example highlighted with circle, it is the second positive example in the ordered list, and it is outranked by two negative examples (shown by arrows). So its contribution to AUC risk is* $N^- + i - r_i^+ = 6 + 2 - 6 = 2$. *Repeating for all* 7 *positive examples the total wrong pairs is* $3+2+2+2+2+1+0 = 12$ *and AUC risk is* $\frac{12}{6\times 7} = \frac{2}{7}$, *which is the same as computed with Eq.*(1).

Lemma 1. *For* $i = 1, \cdots, N^+$, *we have*

$$r_i^+ \leq N^- + i, \;\; c_{N^- + i}^{\uparrow} \geq c_i^{\uparrow +}.$$

Proof of Lemma 1 is provided in the Appendix A.

With these definitions, AUC risk can be defined using the rankings of the predictions [4], which is equivalent to the definition based on the WMW statistics as given in Eq.(1). The intuition behind this equivalent form is a different way to count the number of reverse ordered pairs of positive and negative examples, which is illustrated with the numerical example in Figure 1.

Lemma 2 ([4])**.** *When there is no ties in training data,* i.e.*,* $c_i \neq c_j$ *for* $i \neq j$, *we have*

$$\begin{aligned} L_{AUC} &= \frac{1}{N^+ N^-} \sum_{i=1}^{N^+} (N^- + i - r_i^+) \\ &= 1 + \frac{N^+ + 1}{2N^-} - \frac{1}{N^+ N^-} \sum_{i=1}^{N^+} r_i^+. \end{aligned} \tag{2}$$

Proof of Lemma 2 is provided in the Appendix A. Note that $\sum_{i=1}^{N^+}(N^- + i)$ corresponds to (trivially) the sum of the indices of the largest N^+ (top-N^+) elements in the ranked list of prediction scores, and $\sum_{i=1}^{N^+} r_i^+$ is the sum of the indices of positive examples in the ranked list of predictions. As such, AUC risk as defined in Eq.(2) is proportional to the difference between the two sums.

This gives another intuitive explanation of AUC risk: in a perfect separable case, when the prediction scores of all the positive examples rank higher than those of the negative examples, *i.e.*, all prediction scores of positive examples have ranks $N^- + 1, \cdots, N$ in the ordered list, AUC risk is zero. In the more general cases, AUC risk measures how the rankings of the prediction scores deviate from this ideal case.

4.2 Univariate Bound on AUC risk

The equivalent form of AUC risk of Eq.(2) inspires a new surrogate loss based on the values of the sorted prediction scores $(c_1^{\uparrow}, \cdots, c_N^{\uparrow})$. To be specific, let us define a new quantity

$$\tilde{L} = \frac{1}{N^+N^-} \sum_{i=N^-+1}^{N} c_i^{\uparrow} - \frac{1}{N^+N^-} \sum_{i \in \mathcal{I}^+} c_i. \quad (3)$$

Like AUC risk, $\tilde{L}$ is always nonnegative, as the second term, which is the sum of the prediction scores of all the positive examples, is less than or equal to the first term, which is the sum of the top-N^+ elements of $(c_1, \cdots, c_N)$. Equality holds only when the predictions of all positive examples rank higher than any of the negative examples. Our next result shows that we can bound AUC risk using $\tilde{L}$.

Theorem 1. *When there is no ties in training data,* i.e.*, $c_i \neq c_j$ for $i \neq j$, we have $\tilde{L} \geq 0$. Furthermore, there exist constants $\bar{\alpha} \geq \underline{\alpha} > 0$, such that $\bar{\alpha}\tilde{L} \geq L_{AUC} \geq \underline{\alpha}\tilde{L}$.*

Proof. Using Lemma 1, we have $\sum_{i=1}^{N^+} \left(c_{N^-+i}^{\uparrow} - c_i^{\uparrow +}\right) \geq 0$, therefore $\tilde{L} \geq 0$, and it is zero when $c_{N^-+i}^{\uparrow} = c_i^{\uparrow +}$ for all $i = 1, \cdots, N^+$, *i.e.*, all positive examples outrank all negative examples.

We set $\bar{\alpha}^{-1} = \min_i(c_{i+1}^{\uparrow} - c_i^{\uparrow}) > 0$, and for $i > j$, we have $c_i^{\uparrow} - c_j^{\uparrow} = (c_i^{\uparrow} - c_{i-1}^{\uparrow}) + (c_{i-1}^{\uparrow} - c_{i-2}^{\uparrow}) + \cdots + (c_{j+1}^{\uparrow} - c_j^{\uparrow}) \geq \frac{i-j}{\bar{\alpha}}$. Then we have

$$\begin{aligned}
\bar{\alpha}\tilde{L} &= \tfrac{\bar{\alpha}}{N^+N^-} \textstyle\sum_{i=N^-+1}^{N} c_i^{\uparrow} - \tfrac{\bar{\alpha}}{N^+N^-} \sum_{i \in \mathcal{I}^+} c_i \\
&= \tfrac{\bar{\alpha}}{N^+N^-} \textstyle\sum_{i=1}^{N^+} \left(c_{N^-+i}^{\uparrow} - c_i^{\uparrow +}\right) \\
&= \tfrac{\bar{\alpha}}{N^+N^-} \textstyle\sum_{i=1}^{N^+} \left(c_{N^-+i}^{\uparrow} - c_{r_i^+}^{\uparrow}\right) \\
&\geq \tfrac{1}{N^+N^-} \textstyle\sum_{i=1}^{N^+} \left(N^- + i - r_i^+\right) = L_{\text{AUC}}.
\end{aligned}$$

Next, setting $\underline{\alpha}^{-1} = \max_i(c_{i+1}^{\uparrow} - c_i^{\uparrow})$, and follow a similar derivation, we can obtain the other bound, *i.e.*, $L_{\text{AUC}} \geq \underline{\alpha}\tilde{L}$. The equalities in the bounds hold when $c_{i+1}^{\uparrow} - c_i^{\uparrow}$ is a constant for $i = 1, \cdots, N$, *i.e.*, they are equally spaced. □

4.3 Computing $\tilde{L}$ without Explicit Ranking

However, the ranking operation in $\tilde{L}$ is the main obstacle of using Eq.(3) as a learning objective. However, this can be solved based on the following result on the sum of the top k elements in a set [17, 18], from which we can derive an equivalent form of Eq.(3) that does not rely on ranking elements explicitly.

Lemma 3 ([17, 18])**.** *For N real numbers $z_1 < \ldots < z_N$, we have the equivalence of the sum-of-top-k elements with an optimization problem as*

$$\sum_{i=N-k+1}^{N} z_i = \min_{\lambda} \left\{ k\lambda + \sum_{i=1}^{N} [z_i - \lambda]_+ \right\}, \quad (4)$$

with the optimal $\lambda^\star$ satisfying $z_{N-k} \leq \lambda^\star < z_{N-k+1}$.

Proof of Lemma 3 is provided in the Appendix A. Using Lemma 3, we can rewrite $\tilde{L}$ by as a minimization problem over the auxiliary variable λ, as

$$N^+N^-\tilde{L} = \min_{\lambda} \left\{ N^+\lambda + \sum_{i=1}^{N} [c_i - \lambda]_+ \right\} - \sum_{i \in \mathcal{I}^+} c_i,$$

which can be further converted to

$$\min_{\lambda} \left\{ \sum_{i \in \mathcal{I}^+} \left([c_i - \lambda]_+ - (c_i - \lambda) \right) + \sum_{j \in \mathcal{I}^-} [c_j - \lambda]_+ \right\}.$$

Using the property of the hinge function that $[a]_+ - a = [-a]_+$, we can further simplify $\tilde{L}$, as

$$\begin{aligned}
\tilde{L} &= \tfrac{1}{N^+N^-} \min_{\lambda} \left\{ \textstyle\sum_{i \in \mathcal{I}^+} [\lambda - c_i]_+ + \sum_{j \in \mathcal{I}^-} [c_j - \lambda]_+ \right\} \\
&= \tfrac{1}{N^+N^-} \min_{\lambda} \textstyle\sum_{i=1}^{N} [y_i(\lambda - c_i))]_+ .
\end{aligned}$$

Bringing back the parametric model to form a learning objective based on $\tilde{L}$ as

$$\tilde{L}(\mathbf{w}) = \frac{1}{N^+N^-} \min_{\lambda} \sum_{i=1}^{N} [y_i(\lambda - f_{\mathbf{w}}(\mathbf{x}_i))]_+ . \quad (5)$$

This reformulation of $\tilde{L}$ is still a bound for AUC risk, but it does not require pairwise comparisons between predictions of positive and negative examples, and there is no need to explicitly ranking the predictions. Furthermore, in Eq.(5), the auxiliary variable λ can be understood as a threshold that separates the two classes, and $\tilde{L}(\mathbf{w})$ becomes independent of the choice of threshold by taking the overall minimum over all possible values for the threshold, as in the case of the original definition of AUC risk.

The learning objective $\tilde{L}(\mathbf{w})$ affords an intuitive interpretation in the context of binary classification. It only penalizes those positive examples with predictions less than the threshold, *i.e.*, $[\lambda - f_{\mathbf{w}}(\mathbf{x}_i)]_+$ for $i \in \mathcal{I}^+$, and negative examples with predictions greater than the threshold, *i.e.*, $[f_{\mathbf{w}}(\mathbf{x}_i) - \lambda]_+$ for $i \in \mathcal{I}^-$. All examples that are on the "correct" side of the threshold receive no penalty. According to Lemma 3, the optimal λ takes value in the range of $[c_{N^+}^{\uparrow}, c_{N^++1}^{\uparrow})$.

4.4 Relation with SVM Objective

There are some strong similarities between $\tilde{L}(\mathbf{w})$ and the SVM objective, which is particularly striking in the case of linear prediction function $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. This becomes clearer if we reformulate the SVM objective: if we regard the threshold λ as the bias term in the linear prediction function for SVM[1], $\mathbf{w}^\top \mathbf{x} - \lambda$, we can formulate the linear SVM objective [19] as

$$\tilde{L}_{\text{SVM}}(\mathbf{w}, \lambda) = \sum_{i=1}^{N} [1 + y_i(\lambda - \mathbf{w}^\top \mathbf{x}_i)]_+.$$

Now comparing with Eq.(5), the two objectives has similar forms involving the hinge function. We can further show that $\tilde{L}_{\text{SVM}}(\mathbf{w}, \lambda)$ is an upper-bound of $\tilde{L}(\mathbf{w})$. This is because we have $[1 + y_i(\lambda - \mathbf{w}^\top \mathbf{x}_i)]_+ \geq [y_i(\lambda - \mathbf{w}^\top \mathbf{x}_i)]_+$, so

$$\begin{aligned} \tilde{L}_{\text{SVM}}(\mathbf{w}, \lambda) &= \textstyle\sum_{i=1}^{N} [1 + y_i(\lambda - \mathbf{w}^\top \mathbf{x}_i)]_+ \\ &\geq \textstyle\sum_{i=1}^{N} [y_i(\lambda - \mathbf{w}^\top \mathbf{x}_i)]_+ \\ &\geq \min_\lambda \textstyle\sum_{i=1}^{N} [y_i(\lambda - \mathbf{w}^\top \mathbf{x}_i)]_+ \\ &= \tilde{L}(\mathbf{w}). \end{aligned}$$

As we have shown in Theorem 1, an upper-bound of AUC risk can be established with $\tilde{L}(\mathbf{w})$, and this relation suggests the SVM objective $\tilde{L}_{\text{SVM}}(\mathbf{w}, \lambda)$ is also an upper-bound (albeit looser than $\tilde{L}$) of AUC risk.

This helps to explain some long standing experimental observations (*e.g.*, [5, 6, 7, 8]) that when assessed with AUC, standard SVMs could not be consistently outperformed by other approaches tailored to directly maximize AUC, such as RankBoost [20], AUCsplit (local optimization of AUC) [21], or ROC-SVM [8].

The two learning objectives also differ in two important aspects. The first is the constant 1 in the SVM objective, which corresponds to the margin in constructing the binary classifier. The second difference is that the bias λ in $\tilde{L}$ is eliminated through minimization, but it is still present in the SVM objective.

5 OPTIMIZATION

In this section, we discuss batch and online learning algorithms based on learning objectives formed from Eq.(5).

5.1 Resolving Ties in Prediction Scores

However, Eq.(5) cannot be used as a learning objective due to one important issue. Note that in Eq.(5), the scale of the parameter $\mathbf{w}$ is not fixed, so the learning objective can be reduced by shrinking the scale of $\mathbf{w}$, which leads to a trivial solution with $\mathbf{w} = 0$. The underlying reason is that the formulation of $\tilde{L}$ is based on the assumption of no ties in the prediction scores, while the trivial solution corresponds to the extreme contrary, *i.e.*, the prediction function always produce the same output (zero) regardless of the data.

To resolve this problem, we augment the objective function with two other terms

$$\min_{\mathbf{w}} \tilde{L}(\mathbf{w}) + \frac{\beta}{2} \sum_{i=1}^{N} (1 - y_i f_{\mathbf{w}}(\mathbf{x}_i))^2 + \gamma \Omega(\mathbf{w}), \quad (6)$$

where the second term corresponds to a least squares term to counteract the effect of concentrating $\mathbf{w}$ to zero, the third term $\Omega(\mathbf{w})$ is a regularizer on parameter $\mathbf{w}$, and (β, γ) are weights to the two extra terms.

5.2 Linear Predictor

In general, the learning objective of Eq.(6) is not a convex function of $\mathbf{w}$, but if we choose $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ and $\Omega(\mathbf{w})$ is convex with respect to $\mathbf{w}$ (*i.e.*, $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$), then we can show it is a convex function of $\mathbf{w}$. We first show that $[\mathbf{x}^\top \mathbf{w} - \lambda]_+$ is a convex function. For $\alpha \in [0, 1]$, $\mathbf{w}$, $\mathbf{w}'$, λ, and λ', we have

$$\begin{aligned} &[\mathbf{x}^\top(\alpha \mathbf{w} + (1-\alpha)\mathbf{w}') - (\alpha\lambda + (1-\alpha)\lambda')]_+ = \\ &[\alpha(\mathbf{x}^\top \mathbf{w} - \lambda) + (1-\alpha)(\mathbf{x}^\top \mathbf{w}' - \lambda')]_+ \leq \\ &\alpha[\mathbf{x}^\top \mathbf{w} - \lambda]_+ + (1-\alpha)[\mathbf{x}^\top \mathbf{w}' - \lambda']_+. \end{aligned} \quad (7)$$

Therefore, $\sum_{i=1}^{N} \left[\mathbf{x}^\top \mathbf{w} - \lambda\right]_+ + N^+\lambda$ is a convex function jointly for $(\mathbf{w}, \lambda)$. As the minimization of one variable in a joint convex function, $\min_\lambda \sum_{i=1}^{N} [c_i - \lambda]_+ + N^+\lambda$ is also a convex function of $\mathbf{w}$.

In summary, for the linear case, we can obtain the following convex learning objective with regards to $\mathbf{w}$ and λ jointly,

$$\begin{aligned} &(\mathbf{w}^\star, \lambda^\star) \leftarrow \operatorname{argmin}_{\mathbf{w},\lambda} \tfrac{\gamma}{2}\|\mathbf{w}\|^2 + \\ &\textstyle\sum_{i=1}^{N} \left\{ \left[y_i(\lambda - \mathbf{x}_i^\top \mathbf{w})\right]_+ + \frac{\beta}{2}(1 - y_i \mathbf{x}_i^\top \mathbf{w})^2 \right\} \end{aligned} \quad (8)$$

In the following, we discuss the batch and online optimization of Eq.(8), for which the convergence to global minimum is guaranteed.

5.2.1 Batch Learning

In the batch setting, where we have access to all training examples, we can use block coordinate descent algorithm to optimize Eq.(8). We initialize $\mathbf{w}$ and λ, then iterate between

[1]Typically in SVM we define the linear prediction function as $\mathbf{w}^\top \mathbf{x} + b$, but here we flip the sign of the bias so to better compare with $\tilde{L}(\mathbf{w})$.

$$\begin{aligned}\mathbf{w}^{(t+1)} &\leftarrow \operatorname{argmin}_{\mathbf{w}} \sum_{i=1}^{N} \left[y_i(\lambda^{(t)} - \mathbf{w}^\top \mathbf{x}_i)\right]_+ + \\ &\quad \tfrac{\beta}{2}\sum_{i=1}^{N}(1 - y_i \mathbf{x}_i^\top \mathbf{w})^2 + \tfrac{\gamma}{2}\|\mathbf{w}\|^2; \\ \lambda^{(t+1)} &\leftarrow \tfrac{1}{2}(c^{\uparrow}_{N^+} + c^{\uparrow}_{N^+ + 1}),\end{aligned}$$

where $c_i^{\uparrow}$ is the rerank of $\{\mathbf{x}_i^\top \mathbf{w}^{(t+1)}\}_{i=1}^N$ in the ascending order. The $\mathbf{w}$ sub-problem can be converted to a constrained optimization problem as

$$\begin{aligned}&\min_{\mathbf{w},\mathbf{t}} \sum_{i=1}^{N} t_i + \tfrac{\beta}{2}\sum_{i=1}^{N}(1 - y_i \mathbf{x}_i^\top \mathbf{w})^2 + \tfrac{\gamma}{2}\|\mathbf{w}\|^2; \\ &\text{s.t. } \mathbf{y}_i(\lambda^{(t)} - \mathbf{w}^\top \mathbf{x}_i) \geq t_i, t_i \geq 0.\end{aligned}$$

This is a quadratic convex optimization problem and can be solved with interior point method when the dimensionality of $\mathbf{w}$ is low to medium. For high dimensional $\mathbf{w}$, the online learning algorithm is more effective as it avoids building the Hessian matrix.

5.2.2 Online Learning

Because Eq.(8) does not involve pairwise comparison, we can also derive an online learning algorithm based on stochastic gradient descent [22, 23]. The runtime of the online algorithm does not depend on the number of training examples and thus this algorithm is especially suited for large datasets. Specifically, with initial choice for the value of $\mathbf{w}^{(0)}$, at the t^{th} iteration, a single training example $(\mathbf{x}_{i_t}, y_{i_t})$ is chosen at random from the training set and used to estimate a sub-gradient of the objective, and a step with pre-determined step-size is taken in the opposite direction, as

$$\begin{aligned}\mathbf{w}^{(t+1)} &\leftarrow \mathbf{w}^{(t)} - \eta_t \left((\gamma I + \beta \mathbf{x}_{i_t}^\top \mathbf{x}_{i_t})\mathbf{w}^{(t)} - \right. \\ &\quad \left. (\beta + \mathbf{I}_{y_i(\lambda^{(t)} - \mathbf{w}^\top \mathbf{x}_i) > 0}) y_{i_t} \mathbf{x}_{i_t}\right) \\ \lambda^{(t+1)} &\leftarrow \lambda^{(t)} - \eta_t y_{i_t} \mathbf{I}_{y_i(\lambda^{(t)} - \mathbf{w}^\top \mathbf{x}_i) > 0},\end{aligned} \tag{9}$$

where we can choose the step-size $\eta_t \sim \frac{1}{\sqrt{t}}$, then the SGD algorithm will converge in $O(1/\epsilon)$ steps to the ϵ-accuracy of the global optimal value of Eq.(5) [22, 23]. Note that each step of our online iterative algorithm has space and time complexity of $O(d)$ and $O(1)$, and obviates the need to store or buffer data in previous online AUC optimization methods [10, 9].

6 EXPERIMENTS

We perform several experiments of learning binary classifiers to evaluate the batch and online algorithms optimizing learning objectives given in Eq.(8) (subsequently denoted as `ba-UBAUC` and `ol-UBAUC`, respectively), and compare their performance with existing learning algorithms for AUC optimization.

As in previous works [10, 11], we perform experiments on 12 benchmark datasets that have been used in previous studies. A summary of the data in these datasets is given in Table 1, with the training/testing split obtained from the original dataset. For datasets that are for data with more than 2 class labels (*i.e.*, `news20` and `sector`), following the convention of previous work [10, 11], we convert them to binary classification problems by randomly partitioning the data into two groups, each with equal number of classes. Then the binary class labels are determined from the group to which the original class label belongs. Following the evaluation protocol of [10, 11], the performance of reported is obtained by averaging the AUC scores on the test set for 25 models learned from subsets of the same training set, each is chosen as a random 80% of the original training data.

	train	test	data dim.
diabetes	389	389	8
fourclass	431	431	2
german	500	500	24
splice	1,000	2,175	60
usps	7,291	2,007	256
a9a	32,561	16,281	123
w8a	49,749	14,951	300
mnist	60,000	10,000	780
acoustic	78,823	19,705	50
ijcnn1	49,990	91,701	22
sector	6,412	3,207	55,197
news20	15,935	3,993	62,061

Table 1: *Summary of the* 12 *benchmark datasets used in our experiments. The training/testing splitting is from the original datasets.*

On these datasets, we evaluate and compare UBAUC-based algorithms with four state-of-the-art online and two batch learning algorithms for learning linear binary classifiers that minimizes various pairwise surrogates to the original AUC risk L_{AUC}. The hyper parameters (β, γ) for UBAUC are determined by a grid search on the validation set. The initial learning rate for the online learning algorithm is also set for different dataset by a grid search. We compare the following algorithms with UBAUC-based algorithms.

- SOLAM [11], an online AUC optimization algorithm based on a saddle point reformulation of the pairwise ℓ_2 surrogate loss of AUC risk;
- OPAUC [9], an online AUC optimization algorithm that uses the pairwise ℓ_2 loss surrogate of the AUC objective function;
- OAM [10], an online AUC optimization algorithm that uses the pairwise hinge loss surrogate of the AUC objective function with two variants, one with sequential update (OAMseq) and the other using gradient update (OAMgra);
- B-SVM-OR [6], a batch learning algorithm using the pairwise hinge loss surrogate of the AUC objective function;

	ol-UBAUC	ba-UBAUC	SOLAM	OPAUC	OAM_{seq}	OAM_{gra}	B-SVM-OR	SVM
diabetes	.8326±.0299	.8328±.0352	.8253±.0314	.8309±.0350	.8264±.0367	.8262±.0338	.8326±.0328	.7821±.0145
fourclass	.8301±.0318	.8310±.0296	.8226±.0240	.8310±.0251	.8306±.0247	.8295±.0251	.8305±.0311	.7717±.0294
german	.7928±.0371	.7933±.0324	.7882±.0243	.7978±.0347	.7747±.0411	.7723±.0358	.7935±.0348	.7641±.0283
splice	.9231±.0224	.9269±.0094	.9253±.0097	.9232±.0099	.8594±.0194	.8864±.0166	.9239±.0089	.8439±.0096
usps	.9728±.0051	.9730±.0066	.9766±.0032	.9620±.0040	.9310±.0159	.9348±.0122	.9630±.0047	.8930±.0075
a9a	.9005±.0019	.9009±.0041	.9001±.0042	.9002±.0047	.8420±.0174	.8571±.0173	.9009±.0036	.8213±.0064
w8a	.9673±.0993	.9695±.0079	.9114±.0075	.9633±.0035	.9304±.0074	.9418±.0070	.9495±.0082	.8964±.0029
mnist	.9327±.0239	.9340±.0024	.9324±.0020	.9242±.0021	.8615±.0087	.8643±.0112	.9340±.0020	.8406±.0072
acoustic	.8871±.0035	.8962±.0046	.8898±.0026	.8192±.0032	.7113±.0590	.7711±.0217	.8262±.0032	.7629±.0045
ijcnn1	.9264±.0039	.9337±.0038	.9215±.0045	.9269±.0021	.9209±.0079	.9100±.0092	.9337±.0024	.8793±.0094
sector	.9845±.0033	-	.9834±.0023	.9292±.0081	.9163±.0087	.9043±.0100	-	.8815±.0062
news20	.9468±.0045	-	.9467±.0039	.8871±.0083	.8543±.0099	.8346±.0094	-	.8431±.0127

Table 2: *Comparison of the AUC scores (mean±std.) on test sets of the evaluated datasets.*

- UNI-SVM, a linear SVM algorithm implemented using LIBSVM with SMO minimization [24].

Classification performances measured by the AUC score on the testing dataset of all compared methods for all 12 benchmark datasets are given in Table 2. For fair comparison, we implement all algorithms using MATLAB, and following the default parameter settings in the original papers. Note that the simple implementation of the two batch algorithms cannot handle datasets with high dimensional datasets, *i.e.*, `sector` and `news20`, due to the memory requirement. However, for those datasets that it is feasible to run, `ba-UBAUC`, the batch version optimizing the proposed learning objective, performs best. On the other hand, the results of u`UNI-SVM`, though optimizing a different objective, still achieves reasonable performance when evaluated with AUC. The online algorithm based on the proposed learning objective, `ol-UBAUC`, achieves comparable performance as other state-of-the-art online algorithms based on pairwise surrogate losses to AUC risk, although the improvements of performance on some of the datasets are not conspicuous due to the nature of the data.

On the other hand, the main advantage of `ol-UBAUC` in comparison with other online algorithms is the running efficiency – its per-iteration running time and space complexity is linear in data dimension and do not depend on the iteration number. Furthermore, each iteration of `ol-UBAUC` Eq. (9) corresponds to a simpler update step than the saddle point solve in SOLAM [11]. In Table 3, we show the per-iteration running time and the total running time for the learning objective function to converge to have smaller than 10^{-7} relative changes[2] of the five online algorithms we compared. Note that the online version of the UBAUC-based algorithms has more efficient running time with comparable performances in comparison to existing AUC optimization methods.

	a9a	usps	sector
ol-UBAUC	0.48	0.15	11.24
SOLAM	0.50	0.19	19.90
OPAUC	6.24	4.62	120.30
OAMseq	34.31	13.98	1350.41
OAMgra	34.35	12.54	1350.50

	a9a	usps	sector
ol-UBAUC	0.83	0.15/0.58	276.41
SOLAM	0.99	0.19/0.81	721.52
OPAUC	14.21	4.62/11.23	5540.24
OAMseq	78.42	13.98/32.71	6730.75
OAMgra	69.23	12.54/39.54	6324.64

Table 3: **(top)***The average running time (in seconds) per pass over training data for each online algorithm, and* **(bottom)** *the average running time (in seconds) for the learning objective function to converge to have smaller than* 10^{-7} *relative changes for each online algorithm.*

7 POPULATION FORM

So far, we have described the proposed learning objective over a set of finite training data. In this section, we discuss the population form of the surrogate loss using probability distributions of data. This analysis will shed light on the formal connection of the new surrogate loss with existing methods and can lead to deeper theoretical studies.

We start with the population form of the equivalent definition of AUC risk in Eq.(2). We assume that the input data and label are from a joint model $p(\mathbf{x}, y)$, which induces density models for the predictions $c = f(\mathbf{x})$. As such, we denote $\rho^+(c) = p(c|y = 1)$ and $\rho^-(c) = p(c|y = -1)$ as the (conditional) probability density functions (PDFs) for positive and negative class, respectively. For simplicity, we assume both PDFs have infinite support, *i.e.*, is non-zero for the whole $\mathcal{R}$. Also, we denote $p = Pr(y = 1)$ as the class prior probability.

The joint probability density function of the classification output c is then given by $\rho(c) = p\rho^+(c) + (1 -$

[2]Experiments were performed with running time reported based on a cluster with 12 nodes, each with an Intel Xeon E5-2620 2.0GHz CPU and 64GB RAM. All algorithms are implemented using MATLAB, with available code obtained from the authors of the corresponding publications.

$p)\rho^-(c)$. We also denote $F^+(c) = \int_{-\infty}^{c} \rho^+(c')dc'$, $F^-(c) = \int_{-\infty}^{c} \rho^-(c')dc'$, and $F(c) = \int_{-\infty}^{c} \rho(c')dc'$ as the cumulative distribution functions (CDFs) for ρ^+, ρ^- and ρ, respectively, with $F(c) = pF^+(c)+(1-p)F^-(c)$. $F^+(c)$ is the false negative rate (FNR) and $1 - F^-(c)$ is the false positive rate (FPR).

AUC risk is defined as the area under the whole curve of FNR vs. FPR, as $L_{\text{AUC}} = \int_{-\infty}^{\infty}(1 - F^-(c))dF^+(c)$ [4]. Using relation $F^-(c) = \frac{1}{1-p}(F(c) - pF^+(c))$ yields

$$L_{\text{AUC}} = \frac{1}{1-p}\int_{-\infty}^{\infty}(1 - p + pF^+(c) - F(c))dF^+(c).$$

Because F is a CDF is a continuous monotonic function and $F(c) \leq 1 - p + pF^+(c) \leq 1$, using the mean value theorem, there exists $c' \geq c_0 = \max\{c|F(c) = 1 - p\}$, such that $1-p = F(c_0) \leq F(c') = 1-p+pF^+(c) \leq 1$, and

$$L_{\text{AUC}} = \frac{1}{1-p}\int_{-\infty}^{\infty}(F(c') - F(c))dF^+(c).$$

Next, note that $F(c)$ is Lipschitz with constant $\alpha' \geq \max_c |\rho(c)|$, *i.e.*, $|F(c') - F(c)| \leq \alpha'|c' - c|$, we have

$$L_{\text{AUC}} \leq \frac{\alpha'}{1-p}\int_{-\infty}^{\infty}(c' - c)dF^+(c). \tag{10}$$

Next, we use the following result

Lemma 4. *For $F(c') = 1 - p + pF^+(c)$, we have*

$$\int_{-\infty}^{\infty} c'dF^+(c) = \min_{\lambda}\int_{-\infty}^{\infty}\frac{(c-\lambda)_+}{p}dF(c) + \lambda.$$

Proof of Lemma 4 is provided in the Appendix A. Using Lemma 4, we can rewrite the integral of the right hand side of Eq.(10) as

$$\min_{\lambda}\int_{-\infty}^{\infty}\frac{(c-\lambda)_+}{p}dF(c) + \lambda - \int_{-\infty}^{\infty} cdF^+(c),$$

where the terms being minimized can be further simplified as

$$\int_{-\infty}^{\infty}\frac{(c-\lambda)_+}{p}dF(c) + (\lambda - c)dF^+(c).$$

This can be further expanded using the relation $dF(c) = (1-p)dF^-(c) + pdF^+(c)$ to have

$$\begin{array}{l}\int_{-\infty}^{\infty}(c-\lambda)_+(1-p)dF^-(c)+\\ \int_{-\infty}^{\infty}\left[(\lambda - c) + (c-\lambda)_+\right]pdF^+(c).\end{array}$$

Putting all terms together and using the relation $(c - \lambda)_+ + (\lambda - c) = (\lambda - c)_+$ we have

$$L_{\text{AUC}} \leq \frac{\alpha'}{p(1-p)}\min_{\lambda} E_{c,y}[y(c-\lambda)]_+, \tag{11}$$

where $E_{c,y}$ represents the expectation over c and y.

8 CONCLUSION

In this work, we describe a new surrogate loss to the AUC metric based on a formulation of AUC, which does not require pairwise comparison but rankings of the prediction scores. We further show that the ranking operation can be avoided and the learning objective obtained based on this surrogate affords complexity in time and storage that is linear in the number of training data. We perform experiments to demonstrate the effectiveness of the online and batch algorithms for AUC optimization based on the proposed surrogate.

There are several directions we would like to further explore for this work. First, form the theoretical point of view, we would like to establish the consistency of the proposed learning objective with regards to AUC risk, *i.e.*, the question if the surrogate loss will also lead to the convergence to the optimal AUC risk. The form of our surrogate loss (Eq.(5)) as an optimization problems makes it difficult to apply the techniques used in previous works [14, 15] to this case. We would also like to establish the generalization error between the data form of the loss Eq.(5) and its population form counterpart Eq.(11). From the algorithm perspective, we would like to extend this learning objective to substitute multi-class AUC [4], where multi-class AUC risk is computed as the average of binary class AUC between each pairs of classes. Last, we are interested in applying the online algorithm based on the proposed surrogate loss to non-convex learning objectives such as those used for training deep neural networks.

Acknowledgement. Siwei Lyu is supported by the National Science Foundation (NSF, Grant IIS-1537257) and Yiming Ying is supported by the Simons Foundation (#422504) and the 2016-2017 Presidential Innovation Fund for Research and Scholarship (PIFRS) program from SUNY Albany.

A Appendix: Proofs

Proof of Lemma 1. Being the i$^{\text{th}}$ positive example encountered in the ordered list $(c_1^{\uparrow}, \cdots, c_N^{\uparrow})$, $c_i^{\uparrow +}$ can outrank no more than $N^- + i$ elements in the list, *i.e.*, i positive examples and at most N^- negative examples. Therefore, we have $r_i^+ \leq N^- + i$. By the ranking order we also have $c_{N^-+i}^{\uparrow} \geq c_{r_i^+}^{\uparrow} = c_i^{\uparrow +}$. □

Proof of Lemma 2. Consider the i^{th} positive example encountered in $(c_1^{\uparrow}, \cdots, c_N^{\uparrow})$ starting from the beginning, which has rank r_i^+. The number of negative examples that rank lower than it is $r_i^+ - i$, *i.e.*, there will be $N^- - (r_i^+ - i) = N^- + i - r_i^+$ negative examples with

ranks higher than this positive example, *i.e.*, forming a reversed ordered pair with it. This corresponds to the sum over reversed ordered pairs in the definition of AUC risks of Eq.(1). Summing over all such reverse ordered pairs divided by the number of all such positive-negative pairs (N^+N^-) proves the result. □

Proof of Lemma 3. First, we note that $\sum_{i=N-k+1}^{N} z_i$ is the solution of the following linear programming problem

$$\max_{\mathbf{p}\in R^{n\times 1}} \ \mathbf{p}^\top \mathbf{z}, \ \text{s.t.} \ \mathbf{p}^\top \mathbf{1} = k, \ p_i \in [0,1], \qquad (12)$$

We form its Lagrangian as

$$L = -\mathbf{p}^\top \mathbf{z} - \mathbf{a}^\top \mathbf{p} + \mathbf{b}^\top (\mathbf{p} - \mathbf{1}) + \lambda(\mathbf{p}^\top \mathbf{1} - k), \ (13)$$

where $\mathbf{a} \geq \mathbf{0}$, $\mathbf{b} \geq \mathbf{0}$ and λ are Lagrangian multipliers. Setting the derivative of L with respect to $\mathbf{p}$ to be $\mathbf{0}$, we obtain $\mathbf{a} = \mathbf{b} - \mathbf{z} + \lambda\mathbf{1}$. Substituting this into Eq (13) , we get the dual problem of (12) as

$$\min_{\mathbf{b},\lambda} \ \mathbf{b}^\top \mathbf{1} + k\lambda, \ \text{s.t.} \ \mathbf{b} \geq \mathbf{0}, \mathbf{b} + \lambda\mathbf{1} - \mathbf{z} \geq \mathbf{0}, \qquad (14)$$

The constraints of Eq. (14) suggest that we should have $\mathbf{b}^\top \mathbf{1} \geq \sum_{i=1}^{n} [z_i - \lambda]_+$. As such, the objective function achieves its minimum when the equality holds. Reorganizing terms leads to Eq.(4). Further, when we choose $\lambda^\star$ satisfying $z_{N-k} \leq \lambda^\star < z_{N-k+1}$, we have $k\lambda^\star + \sum_{i=1}^{N} [z_i - \lambda^\star]_+ = k\lambda^\star + \sum_{i=N-k+1}^{N} (z_i - \lambda^\star) = \sum_{i=N-k+1}^{N} z_i$. Thus proves the lemma. □

Proof of Lemma 4. First, we have $dF(c') = pdF^+(c)$, then $\int_{-\infty}^{\infty} c' dF^+(c) = \frac{1}{p} \int_{c_0}^{\infty} c' dF(c')$, where the lower limit of the integral, $c_0 = \max\{c | F(c) = 1 - p\}$, originates from the range of value c'. Next, we compute $\min_\lambda \int_{-\infty}^{\infty} (c - \lambda)_+ dF(c) + p\lambda = \min_\lambda \int_{\lambda}^{\infty} c dF(c) - \lambda \int_{\lambda}^{\infty} dF(c) + p\lambda$. Differentiating the inner terms with regards to λ, we obtain $\int_{\lambda}^{\infty} dF(c) = p$, or $\int_{-\infty}^{\lambda} dF(c) = 1 - p$, so we have at optimum, $\lambda = c_0$. Therefore we have $\min_\lambda \int_{-\infty}^{\infty} (c - \lambda)_+ dF(c) + p\lambda = \int_{c_0}^{\infty} c' dF(c')$. Further rearranging terms proves the result. □

References

[1] C. Cortes and M. Mohri, "AUC optimization vs. error rate minimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[2] W. Kotlowski, K. Dembczynski, and E. Hüllermeier., "Bipartite ranking through minimization of univariate loss," in *International Conference on Machine Learning (ICML)*, 2011.

[3] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under of receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[4] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Machine learning*, vol. 45, pp. 171–186, 2001.

[5] U. Brefeld and T. Scheffer, "AUC maximizing support vector learning.," in *Workshop ROC Analysis in AI in conjunction with European Conference on Artificial Intelligence*, 2005.

[6] T. Joachims, "A support vector method for multivariate performance measures," in *International Conference on Machine Learning (ICML)*, 2005.

[7] R. Caruana and A. Niculescu-Mizil, "Data mining in metric space: an empirical anal- ysis of supervised learning performance criteria.," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

[8] A. Rakotomamonjy, "Optimizing area ROC curve with SVMs," in *Workshop ROC Analysis in AI in conjunction with European Conference on Artificial Intelligence*, 2004.

[9] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang, "Online AUC maximization," in *International Conference on Machine Learning (ICML)*, 2011.

[10] W. Gao, R. Jin, S. Zhu, and Z. H. Zhou, "One-pass AUC optimization," in *International Conference on Machine Learning (ICML)*, 2013.

[11] Y. Ying, L. Wen, and S. Lyu, "Stochastic online AUC maximization," in *Advances in Neural Information Processing Systems (NIPS)*, (Barcelona, Spain), December 2016.

[12] C. Rudin, C. Cortes, M. Mohri, and R. E. Schapire, "Margin-based ranking meets boosting in the middle," in *Learning Theory* (P. Auer and R. Meir, eds.), (Berlin, Heidelberg), pp. 63–78, Springer Berlin Heidelberg, 2005.

[13] H. Steck, "Hinge rank loss and the area under the ROC curve," in *European Conference on Machine Learning (ECML)*, 2007.

[14] S. Clemencon and S. Robbiano, "Minimax learning rates for bipartite ranking and plug-in rules," in *International Conference on Machine Learning (ICML)*, 2011.

[15] S. Agarwal, "Surrogate regret bounds for the area under the roc curve via strongly proper losses," *Journal of Machine Learning Research*, 2013.

[16] W. Gao and Z. Zhou, "On the consistency of auc pairwise optimization," *Artificial Intelligence Journal*, 2014.

[17] W. Ogryczak and A. Tamir, "Minimizing the sum of the k largest functions in linear time," *Information Processing Letters*, vol. 85, no. 3, pp. 117–122, 2003.

[18] Y. Fan, S. Lyu, Y. Ying, and B. Hu, "Learning with average top-k loss," in *Advances in Neural Information Processing Systems (NIPS)*, (Long Beach, CA), 2017.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] Y. Freund, R. Iyer, R. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of Machine Learning Research*, vol. 4, pp. 933–969, 2003.

[21] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal re- gression," in *International Conference on Neural Networks*, 1999.

[22] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *NIPS*, pp. 161–168, 2008.

[23] N. Srebro and A. Tewari, "Stochastic optimization for machine learning," *ICML Tutorial*, 2010.

[24] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.