Noname manuscript No.
(will be inserted by the editor)

Maximization of AUC and Buffered AUC in Binary Classification

Matthew Norton · **Stan Uryasev**

Received: date / Accepted: date

Abstract In binary classification, performance metrics that are defined as the probability that some error exceeds a threshold are numerically difficult to optimize directly and also hide potentially important information about the magnitude of errors larger than the threshold. Defining similar metrics, instead, using Buffered Probability of Exceedance (bPOE) generates counterpart metrics that resolve both of these issues. We apply this approach to the case of AUC, the Area Under the ROC curve, and define Buffered AUC (bAUC). We show that bAUC can provide insights into classifier performance not revealed by AUC, while being closely related as the tightest concave lower bound and representable as the area under a modified ROC curve. Additionally, while AUC is numerically difficult to optimize directly, we show that bAUC optimization often reduces to convex or linear programming. Extending these results, we show that AUC and bAUC are special cases of Generalized bAUC and that popular Support Vector Machine (SVM) formulations for approximately maximizing AUC are equivalent to direct maximization of Generalized bAUC. We also prove bAUC generalization bounds for these SVM's. As a central component to these results, we provide an important, novel formula for calculating bPOE, the inverse of Conditional Value-at-Risk (CVaR). Using this formula, we show that particular bPOE minimization problems reduce to convex and linear programming.

Keywords Buffered Probability of Exceedance · Conditional-Value-at-Risk · AUC · ROC Curve · Buffered AUC · Support Vector Machine · Convex Programming · Classification Performance Metric · Generalization

1 Introduction

In binary classification, some performance metrics can be defined as the probability that some error function exceeds a particular threshold, i.e. by using Probability of Exceedance[1] (POE). For example, if one uses misclassification error, Accuracy is one minus the probability that misclassification error exceeds

M. Norton
Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, 32611
Tel.: 352-871-5031
E-mail: mdnorto@gmail.com

S. Uryasev
Risk Management and Financial Engineering Lab, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611
E-mail: uryasev@ufl.edu

[1] As later defined in Section 3, for a real valued random variable X, the Probability of Exceedance at threshold z is $P(X > z)$.

the threshold of zero. The Area Under the Receiver Operating Characteristic Curve (AUC) is a popular performance metric in classification that can also be viewed in this way, as the probability that 'ranking' error exceeds a threshold of zero. With a long history in signal detection theory [10, 35], diagnostic systems analysis [34], and medical decision making [43], AUC has found much success as a measure of a model's ability to differentiate different classes of events. In machine learning, AUC has gained popularity due to its advantages over Accuracy, particularly when one has no knowledge of misclassification costs or must deal with imbalanced classes [3, 28, 27, 19, 7]. In both of these cases, AUC has benefits over Accuracy, with Accuracy implying equal misclassification costs and heightened emphasis on correctly classifying the majority class.

Defining metrics, like AUC, as the probability that some error exceeds a threshold, although intuitive, produces metrics with some properties that may be undesirable. First, these metrics only consider the number of errors larger than the threshold and do not consider the magnitude of these errors which exceed the threshold. This information, which can sometimes be viewed as the classifier's 'confidence,' may be important when gauging classifier performance. Additionally, there is evidence that consideration of this information can lead to improved generalization, guaranteeing a classification margin when optimally considered [38]. Second, these metrics are difficult to optimize directly. When dealing with empirical observations of data, direct optimization of these metrics yields a non-convex and discontinuous optimization problem. For example, with Accuracy it is common to utilize some convex surrogate to the $0-1$ loss to attempt to optimize Accuracy (e.g., the hinge or exponential loss). With AUC defined with POE, these issues are directly applicable.

Instead of defining metrics in this way, we take the approach of defining metrics with Buffered Probability of Exceedance (bPOE). Focusing our in-depth analysis on the case of AUC, we show that this approach produces a metric that accounts for the magnitude of errors, with direct optimization of the metric reducing to convex, sometimes linear, programming. Recently introduced as a generalization of Buffered Probability of Failure, a concept introduced by [29] and explored further in [21] and [9], bPOE equals one minus the inverse of the superquantile. The superquantile is also commonly known as the Conditional Value-at-Risk (CVaR) from the financial engineering literature. In this regard, the first major contribution of this paper is a novel formula for simultaneously calculating bPOE and POE. We show that this important formula reduces many bPOE optimization problems to convex and linear programming. In the more specific context of AUC, this formula is the key to producing most of our results.

Furthermore, we apply bPOE to the case of AUC to create a new, AUC-like counterpart metric called Buffered AUC (bAUC). This new metric is indeed a counterpart to AUC. It is the tightest concave lower bound of AUC. Like AUC, it measures a classifiers ability to discriminate instances belonging to positive and negative classes. It can also be represented as the area under a modified ROC curve, which we call the bROC curve. Additionally, we compare bAUC to two other AUC counterparts in the literature (sAUC and pAUC) that attempt to account for the magnitude of errors in a similar way. We show, empirically, that as a model selection metric, bAUC produces better models in terms of AUC and is thus more similar to AUC.

With AUC defined with POE, it is extremely difficult to optimize directly, yielding a non-convex and discontinuous objective function when faced with discrete observations. We show that bAUC has substantial benefits in this regard, with direct optimization reducing to convex and linear programming. We then introduce Generalized bAUC, a natural extension of bAUC, and show that this produces a family of metrics, in which AUC and bAUC belong, all having interpretations as areas under modified ROC curves. We then provide a formulation for optimizing Generalized bAUC and show that the popular AUC maximizing RankSVM of [15, 4] is a special case of maximizing Generalized bAUC. Thus, we show that bAUC has already found its way into the AUC maximization literature, albeit not explicitly, as an easily optimizable metric alternative to AUC that leads to a classification margin. Additionally, this allows us to reinterpret the RankSVM, showing that the tradeoff parameter is related to bPOE threshold and that the optimal objective value is, in fact, equal to one minus Generalized bAUC.

The result that the RankSVM is, in fact, directly maximizing bAUC to approximately maximize AUC provides many new insights into this highly successful algorithm. First, the fact that bAUC is the tightest concave lower bound of AUC provides evidence for why RankSVM is state-of-the-art in terms of AUC maximization. Second, we can show that the objective value and trade-off parameter have interpretations in terms of bPOE and its threshold. Third, we provide new insights into generalization bounds for the RankSVM. Traditionally, generalization bounds for SVM's are provided in terms of misclassification rate, where the misclassification rate on the true distribution is bounded above by some function of empirical error rate and a measure of the complexity of the hypothesis space. With the overall goal of the SVM viewed as minimization of misclassification error, this is a sensible approach. We would like to know if our classifier generalizes w.r.t. the metric we are attempting to optimize. After showing that the RankSVM is a special case of bAUC maximization, we provide generalization bounds for bAUC. These bounds also hold for AUC. Unlike many classical generalization bounds, ours is not controlled by the complexity of the hypothesis space. It is controlled by the empirical bAUC and the threshold for Generalized bAUC, i.e. the bPOE threshold. Thus, not only do we show that the RankSVM is simply a special case of direct maximization of bAUC, we show that it generalizes w.r.t. bAUC and that bPOE threshold plays an important role in this generalization.

The remainder of this paper is organized in the following manner. Section 2 reviews the AUC performance metric and issues associated with AUC, including difficulties with direct maximization. Section 3 reviews superquantiles and bPOE. We then introduce an important calculation formula for bPOE and show that under particular circumstances, minimization of bPOE can be reduced to convex, sometimes linear, programming. Section 4 uses the bPOE concept to introduce bAUC. We discuss its value as a natural counterpart to AUC as a classifier performance metric. We show that it can be presented as the area under a modified ROC curve. We highlight that bAUC is the tightest concave lower bound of AUC and that it can be easily optimized. We also provide an accompanying case study demonstrating available software implementations for efficient calculation and optimization. Section 5 briefly illustrates the value of the bROC curve and presents an empirical comparison with other AUC counterparts. Section 6 generalizes the bAUC definition, presents it as a family of modified ROC curves with corresponding areas under these curves, and presents a formulation for maximizing this quantity. We then discuss its relation to existing SVM-based AUC maximization formulations and prove their equivalence. Finally, we show that these SVM's generalize w.r.t. to bAUC by providing theoretical guarantees for performance w.r.t. the true bAUC.

2 The AUC Performance Metric

In this paper, we consider the binary classification task where we have random vectors X^+, X^- in $\mathbb{R}^n$ that belong, respectively, to classes $(Y = +1)$ and $(Y = -1)$. We are given N samples $X_1, ..., X_N$ of the random vector $X = X^+ \cup X^-$, of which m^+ have positive label, m^- have negative label, and we must choose a scoring function $h : \mathbb{R}^n \to \mathbb{R}$ and decision threshold $t \in \mathbb{R}$ to create a classifier with decision rule

$$Y_i = \begin{cases} +1 & \text{if } h(X_i) > t \\ -1 & \text{if } h(X_i) \leq t \ . \end{cases}$$

2.1 Defining AUC: Two Perspectives

AUC is a popular performance metric that measures the ability of a scoring function, h, to differentiate between two randomly selected instances from opposite classes. As opposed to a metric such as Accuracy, which considers the threshold t, AUC does not and is a measure of separation between score distributions $h(X^+)$ and $h(X^-)$. In other words, while accuracy is a direct measure of a classifiers ability to properly

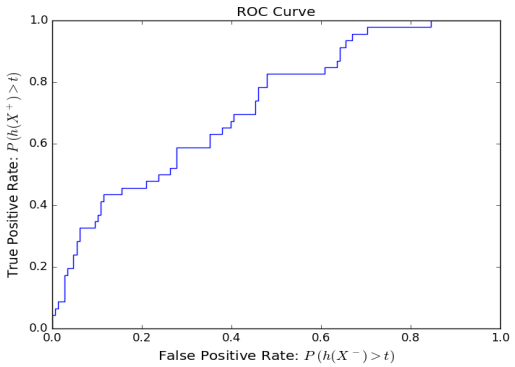

Fig. 1: An example of an Empirical ROC curve for fixed h and data set consisting of samples of X. We plot the Empirical True Positive Rate, $P(h(X^+)>t)$, on the vertical axis and the Empirical False Positive Rate, $P(h(X^-)>t)$, on the horizontal axis for all values of decision threshold $t \in \mathbb{R}$.

classify a single randomly chosen sample, AUC is concerned with a classifiers ability to properly rank two randomly selected samples that are presumed to be in different classes. This is a beneficial measure when classes are imbalanced or misclassification costs are unknown [3, 28, 27, 19, 7].

The AUC metric is defined as the Area Under the Receiver Operating Characteristic Curve (ROC curve). Figure 1 shows an example ROC curve[2], which plots the True Positive Rate, $P(h(X^+)>t)$, on the vertical axis and the False Positive Rate, $P(h(X^-)>t)$, on the horizontal axis for different values of t. The AUC is the area under the curve formed by plotting pairs $(P(h(X^-)>t), P(h(X^+)>t))$ for all thresholds $t \in \mathbb{R}$. Specifically, we can write this in integral form. If we let $P(h(X)>t) = 1 - F_{h(X)}(t)$ be one minus the cumulative density function of $h(X)$, AUC for a scoring function h can be written as,

$$AUC(h) = \int_t P(h(X^+)>t)\, dP(h(X^-)>t)\,. \tag{1}$$

This paper is focused on an equivalent probabilistic definition of AUC provided by [14]. Hanley and McNiel showed that the area under the ROC curve is equal to the probability that a randomly selected positive sample will be scored higher than a randomly selected negative sample,

$$AUC(h) = P\left(h(X^+) > h(X^-)\right). \tag{2}$$

With this paper focusing on POE and bPOE, we write AUC as one minus the probability of 'ranking error' $\xi(h) = -(h(X^+) - h(X^-))$ exceeding zero[3]. Specifically,

$$AUC(h) = 1 - P(\xi(h) \geq 0).$$

Additionally, since the true distribution of X^+ and X^- are rarely known, we often work with samples $X_1^+, ..., X_{m^+}^+$, $X_1^-, ..., X_{m^-}^-$. In this case, denote ranking errors as $\xi_{ij}(h) = -\left(h(X_i^+) - h(X_j^-)\right)$ and let I_λ denote an indicator function of the condition λ. We then have that AUC is approximated as,

$$AUC(h) = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} I_{h(X_i^+) > h(X_j^-)} = 1 - \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} I_{\xi_{ij}(h) \geq 0}\,.$$

[2] In this paper, our examples are Empirical ROC curves, where we have a fixed h and samples of the random variables X^+, X^-.

[3] Note that this is POE with a non-strict inequality.

Furthermore, the ROC curve is estimated by plotting the True Positive Rate and False Positive Rate for thresholds $t \in S = \{h(X_1^+), ..., h(X_{m^+}^+), h(X_1^-), ..., h(X_{m^-}^-)\}$ on the ROC plot and connecting these points by some means to make an ROC curve (e.g. throughout the paper, we simply use linear interpolation to connect these points in our ROC plots).

For a more thorough introduction to AUC and the use of the ROC curve, we refer readers to [11]. Additionally, for a broader view of AUC and its relation to other performance metrics, we refer readers to [16].

2.2 Properties of AUC

As a performance metric, AUC provides insight into the ranking quality of a classifier by considering pairwise differences of scores given to samples from opposing classes. With each sample data point receiving a *score*, $h(X_i)$, the ordering of these scores (i.e. the 'ranking' induced by the scoring function) can be an important indicator of classifier performance (see e.g. [5, 33, 7, 15]). Specifically, AUC considers the distribution of ranking errors $\xi_{ij}(h)$, where a pair of samples X_i^+, X_j^- are properly ranked by h if $\xi_{ij}(h) < 0$, and equals the proportion of ranking errors $\xi_{ij} < 0$. AUC, though, does not consider the *magnitude* of ranking errors, i.e. the *confidence* with which the classifier correctly or incorrectly ranks pairs of samples. Therefore, if the magnitude of ranking errors is an important performance indicator, AUC may not be a desirable performance measure. This characteristic parallels that of Value-at-Risk (VaR) in Financial Engineering. It hides potentially important information about tail behavior by failing to consider the magnitude of tail losses.

Maximizing AUC is also a challenging task, as it is akin to probability minimization for discrete distributions, an optimization task which yields a discontinuous and non-convex objective function. Many AUC optimization approaches exist (see e.g. [4, 22, 18, 7, 17, 23]). These approaches, though, utilize approximations of the AUC objective and do not optimize AUC directly. For example, [22] optimizes an AUC approximation by replacing the indicator loss with a continuous sigmoid function. This yields a continuous optimization problem, though still non-convex.

3 bPOE and bPOE Optimization

With AUC defined as the one minus the probability that ranking error exceeds zero, we explore the use of a counterpart to POE called bPOE. Specifically, a generalization of Buffered Probability of Failure [30], bPOE is the inverse of the superquantile (CVaR) defined in [31]. In this section, after reviewing these concepts, we present a novel formula for bPOE that simultaneously calculates POE. We show that this formula allows certain bPOE minimization problems to be reduced to convex, sometimes linear, programming. This result is particularly important when we apply bPOE to create bAUC in Section 4.

3.1 bPOE and Tail Probabilities

When working with optimization of tail probabilities, one frequently works with constraints or objectives involving *probability of exceedance* (POE), $p_z(X) = P(X > z)$, or its associated quantile $q_\alpha(X) = \min\{z | P(X \leq z) \geq \alpha\}$, where $\alpha \in [0,1]$ is a probability level. The quantile is a popular measure of tail probabilities in financial engineering, called within this field Value-at-Risk by its interpretation as a measure of tail risk. The quantile, though, when included in optimization problems via constraints or objectives, is quite difficult to treat with continuous (linear or non-linear) optimization techniques.

A significant advancement was made in [31, 32] in the development of an approach to overcome the difficulties raised by the use of the quantile function in optimization. They explored a replacement for the quantile, called CVaR within the financial literature, and called the superquantile in a general context. The superquantile is a measure of uncertainty similar to the quantile, but with superior mathematical properties. Formally, the superquantile (CVaR) for a continuously distributed X is defined as

$$\bar{q}_\alpha(X) = E\left[X | X > q_\alpha(X)\right].$$

For general distributions, the superquantile can be defined by the following formula,

$$\bar{q}_\alpha(X) = \min_\gamma \gamma + \frac{E[X-\gamma]^+}{1-\alpha}, \tag{3}$$

where $[\cdot]^+ = \max\{\cdot, 0\}$.

Similar to $q_\alpha(X)$, the superquantile can be used to assess the tail of the distribution. The superquantile, though, is far easier to handle in optimization contexts. It also has the important property that it considers the magnitude of events within the tail. Therefore, in situations where a distribution may have a heavy tail, the superquantile accounts for magnitudes of low-probability large-loss tail events while the quantile does not account for this information.

Working to extend this concept, bPOE was developed as the inverse of the superquantile in the same way that POE is the inverse of the quantile. Specifically, bPOE is defined in the following way, where $\sup X$ denotes the essential supremum of random variable X.

Definition 1 ([21]). *bPOE of random variable X at threshold z equals*

$$\bar{p}_z(X) = \begin{cases} \max\{1-\alpha | \bar{q}_\alpha(X) \geq z\}, & \text{if } z \leq \sup X, \\ 0, & \text{otherwise}. \end{cases}$$

In words, bPOE calculates one minus the probability level at which the superquantile equals the threshold. Roughly speaking, bPOE calculates the proportion of worst case outcomes which average to z. We note that there exist two slightly different variants of bPOE, called Upper and Lower bPOE. For this paper, we utilize Upper bPOE. For the interested reader, details regarding the difference between Upper and Lower bPOE are contained in the appendix.

3.2 Calculation of bPOE

Using Definition 1, bPOE would seem troublesome to calculate. In fact, it is not even clear how one would go about calculating bPOE exactly. For example, using (1), one would need to calculate the superqunatile for many different probability levels until it was equal (or close) to the threshold z. Here, one would need to, for example, use binary search to find the proper probability level.

In Proposition 1, we introduce the first explicit calculation formula for bPOE. We view this new formula as a critical step in development of the bPOE concept. Not only are we the first to propose such a calculation formula for bPOE, but we show that this formula allows some bPOE minimization problems to be reduced to convex and linear programming. Additionally, calculating bPOE at threshold z with this formula allows simultaneous calculation of the threshold γ at which $P(X > \gamma) = \bar{p}_z(X)$, providing information about bPOE and POE at the same probability level.

For Proposition 1, we differentiate between the *upper* quantile $q^+_\alpha(X) = \inf\{x | P(X \leq x) > \alpha\}$ and the *lower* quantile $q_\alpha(X) = \min\{z | P(X \leq z) \geq \alpha\}$. For details regarding the subtle differences between the upper and lower quantile, see [32]. Note, though, that for continuously distributed X, we have that $q^+_\alpha(X) = q_\alpha(X)$. Furthermore, in practice, for discretely distributed X (e.g. empirical distributions based upon a finite number of observations) the difference between these values is often very small.

Proposition 1. *Given a real valued random variable X and a fixed threshold z, bPOE for random variable X at z equals*

$$\bar{p}_z(X) = \inf_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma} = \begin{cases} \lim\limits_{\gamma\to-\infty} \frac{E[X-\gamma]^+}{z-\gamma} = 1\,, & \text{if } z \le E[X]\,, \\ \min\limits_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma}\,, & \text{if } E[X] < z < \sup X\,, \\ \lim\limits_{\gamma\to z^-} \frac{E[X-\gamma]^+}{z-\gamma} = P(X=\sup X)\,, & \text{if } z = \sup X\,, \\ \min\limits_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma} = 0\,, & \text{if } \sup X < z. \end{cases} \tag{4}$$

Furthermore, if $z \in (E[X], \sup X)$ and $\min\limits_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma} = 1-\alpha$, then the set of minimizers is given by the interval $[q_\alpha(X), q^+_\alpha(X)]$, or equivalently $\{\gamma \mid \gamma \in [q_\alpha(X), q^+_\alpha(X)]\} = \operatorname{argmin}\limits_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma}$. Moreover, for the left-endpoint of this interval, $\gamma^* = q_\alpha(X)$, we have that $P(X > \gamma^*) = \bar{p}_z(X)$.

Proof. We prove four cases. Note that case 1 and 3 coincide for constant random variable X, when $z = \sup X$.

Case 1: $z \le E[X]$.
Assume $z \le E[X]$. First, note that $\bar{p}_z(X) = \max\{1-\alpha | \bar{q}_\alpha(X) \ge z\} = 1$. This follows from the fact that $\bar{q}_0(X) = E[X]$. Then, notice that

$$\inf_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma} = \inf_{0<z-\gamma} E[\frac{X}{z-\gamma} - \frac{\gamma}{z-\gamma}]^+ \,. \tag{5}$$

Letting $a = \frac{1}{z-\gamma}$, we get

$$\inf_{0<z-\gamma} E[\frac{X}{z-\gamma} - \frac{\gamma}{z-\gamma}]^+ = \inf_{a>0} E[aX + a(\frac{1}{a} - z)]^+ = \inf_{a>0} E[a(X-z)+1]^+ \,. \tag{6}$$

Now, $0 \le E[X] - z \implies$ for every $a > 0$, $E[a(X-z)+1]^+ \ge E[a(X-z)+1] \ge a(E[X]-z)+1 \ge 1$. This implies that,

$$0 \in \operatorname*{argmin}_{a\ge 0} E[a(X-z)+1]^+ \,.$$

Then, notice that since $0 \in \operatorname*{argmin}\limits_{a\ge 0} E[a(X-z)+1]^+$ and that for every $a > 0$, $E[a(X-z)+1]^+ \ge 1$ we have that

$$\inf_{a>0} E[a(X-z)+1]^+ = \min_{a\ge 0} E[a(X-z)+1]^+ = E[0(X-z)+1]^+ = 1 \,.$$

Finally, noting that if $a = \frac{1}{z-\gamma}$ then $\lim_{(z-\gamma)\to\infty} \frac{1}{z-\gamma} = 0 = a$ and

$$\inf_{0<z-\gamma} \frac{E[X-\gamma]^+}{z-\gamma} = \min_{a\ge 0} E[a(X-z)+1]^+ = E[0(X-z)+1]^+ = \lim_{(z-\gamma)\to\infty} \frac{E[X-\gamma]^+}{z-\gamma} = 1 \,.$$

Case 2: $E[X] < z < \sup X$.
Assume that $E[X] < z < \sup X$. This assumption and Definition 2 imply that

$$\bar{p}_z(X) = \max\{1-\alpha | \bar{q}_\alpha(X) \ge z\} = \min\{1-\alpha | \bar{q}_\alpha(X) \le z\} \,. \tag{7}$$

Recall the formula for the superquantile given in [31],

$$\bar{q}_\alpha(X) = \min_\gamma \left[\gamma + \frac{E[X-\gamma]^+}{1-\alpha}\right] = \min_\gamma g(X,\alpha,\gamma) \,. \tag{8}$$

Note also [32] states that if $\gamma^* \in \operatorname{argmin}_{\gamma} g(X,\alpha,\gamma)$, then
$\bar{q}_\alpha(X) = \gamma^* + \frac{E[X-\gamma^*]^+}{1-\alpha}$ and $\gamma^* \in [q_\alpha(X), q_\alpha^+(X)]$, which is the complete interval of minimizers.
Next, using (7) and (8) we get

$$\bar{p}_z(X) = \min\{1-\alpha : \min_{\gamma} g(X,\alpha,\gamma) \leq z\} \,. \tag{9}$$

Then, considering (8) we can write (9) as,

$$\begin{aligned} \bar{p}_z(X) = & \min_{\alpha,\gamma} \quad 1-\alpha \\ & s.t. \quad \gamma + \frac{E[X-\gamma]^+}{1-\alpha} \leq z \,. \end{aligned} \tag{10}$$

Let (γ^*, α^*) denote an optimal solution vector to (10). Since $z < \sup X$, the formula (8) implies that

$$\gamma^* \in [q_{\alpha^*}(X), q_{\alpha^*}^+(X)] \implies \gamma^* < \bar{q}_{\alpha^*}(X) = z \,.$$

This implies that $\gamma^* < z$. Explicitly enforcing the constraint $\gamma < z$ allows us to rearrange (10) without changing the optimal solution or objective value,

$$\begin{aligned} \bar{p}_z(X) = & \min_{\alpha,\gamma<z} \quad 1-\alpha \\ & s.t. \quad 1-\alpha \geq \frac{E[X-\gamma]^+}{z-\gamma} \,. \end{aligned} \tag{11}$$

Simplifying further, this becomes

$$\bar{p}_z(X) = \min_{\gamma<z} \quad \frac{E[X-\gamma]^+}{z-\gamma} \,. \tag{12}$$

Case 3: $z = \sup X$.
Assume $z = \sup X$. First, note that $\bar{p}_z(X) = \max\{1-\alpha | \bar{q}_\alpha(X) \geq z\} = P(X = \sup X)$. This follows from the fact that $\bar{q}_{(1-P(X=\sup X))}(X) = \sup X$. Next, recall that with (5) and (6) for $a = \frac{1}{z-\gamma}$, we get

$$\inf_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma} = \inf_{a>0} E[a(X-z)+1]^+ \,.$$

Since $\sup X - z = 0$, we have

$$\inf_{a>0} E[a(X-z)+1]^+ = \lim_{a\to\infty} E[a(X-z)+1]^+ = P(X = \sup X) \,.$$

To see this, notice that for any realization X_0 of X, where $X_0 - z < -\frac{1}{a}$, we get $[a(X_0-z)+1]^+ = 0$. Furthermore, for any realization X_1 of X where $X_1 = \sup X = z$ we have that $[a(X_1-z)+1]^+ = [0+1]^+ = 1$. Thus,

$$\lim_{a\to\infty} E[a(X-z)+1]^+ = 0 * \left(\lim_{a\to\infty} P(X-z < -\frac{1}{a}) \right) + 1 * P(X = \sup X) = P(X = \sup X) \,.$$

Case 4: $z > \sup X$.
Assume that $z > \sup X$. First, note that $\bar{p}_z(X) = 0$. This follows immediately from Definition 2 (i.e. the 'otherwise' case). Next, recall again that with (5) and (6) for $a = \frac{1}{z-\gamma}$, we get

$$\inf_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma} = \inf_{a>0} E[a(X-z)+1]^+ \,.$$

Since $\sup X - z < 0$, then for any $0 < a \leq z - \sup X$ we have that $P(\frac{X-z}{a} \leq -1) = 1$ implying that $E[\frac{X-z}{a} + 1]^+ = 0$. This gives us that

$$\inf_{a>0} E[a(X-z)+1]^+ = \min_{a>0} E[a(X-z)+1]^+ = 0 \,.$$

□

Thus, via Proposition 1 we have provided a surprisingly simple formula for calculating bPOE that is similar to formula (3). In the following section, we show that the true power of formula (4) lies in the fact that it can be utilized to reduce particular bPOE minimization problems to convex, sometimes even linear, programming. In fact, the impact of this formula has already been seen in papers following this work. In particular, our formula (4) has already been used in a number of studies, including [24, 21, 9, 20, 37]. In particular, in [21], a paper which followed this work, our formula (4) is used extensively to derive key properties of bPOE and more general convex reformulations of bPOE minimization.

3.3 bPOE Optimization

To demonstrate the ease with which bPOE can be integrated into optimization frameworks, particularly when compared to POE, consider the following optimization setup. Assume we have a real valued positive homogenous random function $f(w,X)$ determined by a vector of control variables $w \in \mathbb{R}^n$ and a random vector X. By definition, a function $f(w,X)$ is "positive homogeneous" with respect to w if it satisfies the following condition: $af(w,X) = f(aw,X)$ for any $a \geq 0, a \in \mathbb{R}$. Note that we consider only positive homogeneous functions since they are the type of error function we consider in the case of AUC.

Now, assume that we would like to find the vector of control variables, $w \in \mathbb{R}^n$, that minimize the probability of $f(w,X)$ exceeding a threshold of $z = 0$. We would like to solve the following POE optimization problem.

$$\min_{w \in \mathbb{R}^n} \quad p_0(f(w,X)) \,. \tag{13}$$

Here we have a discontinuous and non-convex objective function (for discretely distributed X) that is numerically difficult to minimize. Consider minimization of bPOE, instead of POE, at the same threshold $z = 0$. This is posed as the optimization problem

$$\min_{w \in \mathbb{R}^n} \quad \bar{p}_0(f(w,X)) \,. \tag{14}$$

Given Proposition 1, (14) can be transformed into the following.

$$\min_{w \in \mathbb{R}^n, \gamma < 0} \quad \frac{E[f(w,X) - \gamma]^+}{-\gamma} \,. \tag{15}$$

Notice, though, that the positive homogeneity of $f(w,X)$ allows us to further simplify (15) by getting rid of the γ variable. Thus, we find that bPOE minimization of $f(w,X)$ at threshold $z = 0$ can be reduced to (16).

$$\min_{w \in \mathbb{R}^n} \quad E[f(w,X)+1]^+ \,. \tag{16}$$

For convex f, (16) is a convex program. Furthermore, if f is linear, (16) can be reduced to linear programming. This is substantially easier to handle numerically than the non-convex and discontinuous POE minimization (13).

Given the attractiveness of bPOE and the superquantile within the optimization context, we are inclined to apply these concepts to define a bPOE variant of AUC. Not only would this buffered variant give way to more well behaved optimization problems, but it would provide a measure of classifier performance that considers the magnitude of ranking errors $\xi_{ij}(h)$ instead of only a discrete count of the number of ranking errors exceeding zero.

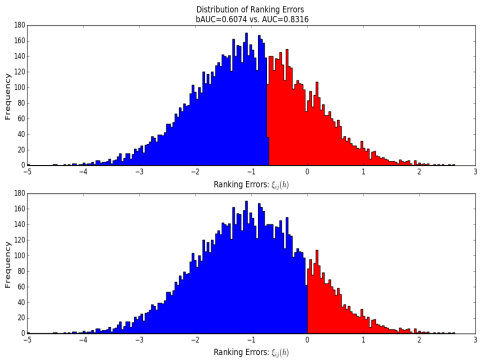

Fig. 2: In both charts, we plot the same distribution of ranking errors $\xi_{ij}(h)$ for a fixed h. In the top chart, we highlight the largest errors that have average magnitude equal to zero, i.e. the errors considered by bAUC. In the bottom chart, we highlight the errors that exceed zero, i.e. the errors considered by AUC. We have that bAUC=.6074 and AUC=.8316.

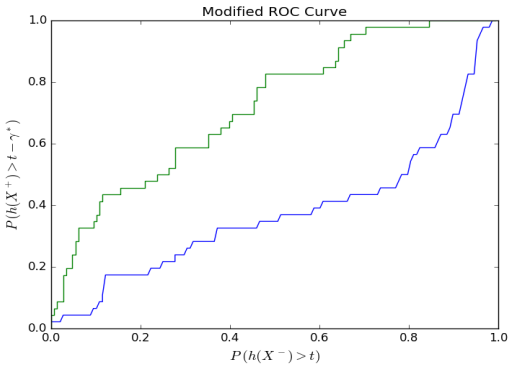

Fig. 3: We have a fixed classifier h. The area under the upper curve corresponds to $AUC(h)$, where $\gamma^* = 0$. The area under the lower curve corresponds to $bAUC(h)$, where $\gamma^* < 0$.

4 Buffered AUC: A New Performance Metric

4.1 Buffered AUC

With AUC defined as $1 - P(\xi(h) \geq 0)$, we can create a natural alternative to AUC called *Buffered AUC* (bAUC) by using bPOE instead of POE. If we assume that we have samples of our random vectors X^+, X^- and are thus working with the empirical distribution of ranking errors $\xi_{ij}(h)$, we have that bAUC equals one minus the proportion of largest ranking errors $\xi_{ij}(h)$ that have average magnitude equal to zero. Specifically, we have the following general definition.

Definition 2 (*Buffered AUC*)**.** *For a scoring function $h : \mathbb{R}^n \to \mathbb{R}$, bAUC of h is defined as*

$$bAUC(h) = 1 - \bar{p}_0(\xi(h)). \tag{17}$$

To begin, we can look at a graphical example comparing bAUC and AUC in Figure 2. Here, we plot the distribution of ranking errors $\xi_{ij}(h)$ for a fixed scoring function h for some dataset. In the bottom chart, we highlight the errors exceeding zero, i.e. the ranking errors considered by AUC. Thus, in the bottom chart, AUC equals one minus the proportion of errors larger than zero. In the top chart, we highlight the largest errors that have average magnitude equal to zero, i.e. the ranking errors considered by bAUC. Thus, in the top chart, we see that bAUC is smaller than AUC, as it considers not only errors larger than zero but also some negative errors most near to zero.

This metric, utilizing bPOE instead of POE, is similar to AUC. Both are concerned with ranking errors, measuring the tail of the error distribution $\xi(h)$. In fact, as shown below in Proposition 2, bAUC is a lower bound for AUC. Thus, classifiers with large bAUC necessarily have large AUC. Moreover, we find that bAUC is not simply a lower bound of AUC, but it is the unique maximal quasi-concave lower bound. This effectively means that there does not exist a concave function that is a tighter lower bound of AUC. Note that Proposition 2 is simply a special case of the result in [21] showing that bPOE is the minimal quasi-convex upper bound of POE.

Proposition 2. *For a scoring function $h : \mathbb{R}^n \to \mathbb{R}$,*

$$bAUC(h) \leq AUC(h).$$

Furthermore, let $\mathscr{S} = \{\xi(h) \mid h : \mathbb{R}^n \to \mathbb{R}, E[\xi(h)] < \infty]\}$ *denote the space of random variables (errors) with finite expectation. Among law-invariant*[4] *functions on* $\mathscr{S}$*, bAUC is the unique maximal quasi-concave lower bound of AUC. (i.e. If* $g : \mathscr{S} \to \mathbb{R}$ *is quasi-concave and* $g(h) \leq AUC(h)$ *for all scoring functions* $h : \mathbb{R}^n \to \mathbb{R}$*, then* $g(h) \leq bAUC(h)$ *for all scoring functions* $h : \mathbb{R}^n \to \mathbb{R}$*.)*

Proof. From [21], we know that for any threshold $z \in \mathbb{R}$ and real valued random variable X that, $P(X \geq z) \leq \bar{p}_z(X)$. Therefore, $1 - \bar{p}_0(\xi(h)) \leq 1 - P(\xi(h) \geq 0)$.
Furthermore, from [21], we know for any threshold $z \in \mathbb{R}$ that $\bar{p}_z(X)$, as a function over the space of random variables X with finite expectation, is the minimal quasi-convex upper bound of $P(X \geq z)$. Therefore, it follows that $1 - \bar{p}_z(X)$ is the minimal quasi-concave lower bound of $1 - P(X \geq z)$. □

Unlike AUC, though, bAUC is sensitive to the magnitude of ranking errors $\xi(h)$. In addition, bAUC does not only consider ranking *errors*, meaning incorrectly ranked pairs of points with $\xi_{ij}(h) > 0$. It also takes into account the confidence with which the classifier *correctly* ranked some instances, meaning the 'errors' $\xi_{ij}(h)$ that are less than, but most near to zero. These correctly ranked instances constitute the *buffer*. We discuss this concept and other differences further in the next section.

We also mention here that penalizing a classifier for some errors that are less than, but most near to zero is reminiscent of certain convex surrogates for the 0-1 loss, such as the hinge loss or log-loss. It is well known in the machine learning literature that this property is beneficial for generalization, as it promotes a classification margin (see e.g. [38]). In later sections, we will see this connection emerge further. First, we will see that bAUC maximization reduces to minimizing a form of expected hinge loss. Furthermore, we will see that a more general form of bAUC maximization shares fundamental connections with SVM's, a formulation which also minimizes a form of expected hinge loss.

4.2 The bAUC Buffer and Sensitivity to Classifier Confidence

Focusing on the benefits of bAUC's sensitivity to the magnitude of ranking errors $\xi_{ij}(h)$, we provide two examples illustrating situations where two classifiers give the same AUC, but where one of the classifiers is clearly a better ranker than the other. We show how bAUC reveals this discrepancy. The first example focuses on the importance of the bAUC buffer. The second example simply illustrates a situation where the magnitude of the ranking errors larger than zero, $\xi_{ij}(h) > 0$, would be important when selecting between classifiers.

As already mentioned, bAUC considers the magnitude of the positive errors, $\xi_{ij}(h) > 0$. Importantly, bAUC also considers the magnitude of the 'errors' that are less than, but most near to zero. This buffer may be important as illustrated in the following example. Let I_λ be an indicator function as specified in Section 2.1. Consider the task of comparing the ranking ability (on the same data set) of two imperfect classifiers[5], h_1 and h_2, that have equal AUC values, meaning that

$$(a) \quad \left(1 - \frac{1}{m^+m^-}\sum_i\sum_j I_{\xi_{ij}(h_1)\geq 0}\right) = \left(1 - \frac{1}{m^+m^-}\sum_i\sum_j I_{\xi_{ij}(h_2)\geq 0}\right) > 0\,.$$

Assume also that both classifiers produce incorrect rankings with the same magnitude (i.e. confidence), meaning that

$$(b) \quad \xi_{ij}(h_1) = 1 \;\forall\; (i,j) \text{ with } \xi_{ij}(h_1) \geq 0 \text{ and } \xi_{ij}(h_2) = 1 \;\forall\; (i,j) \text{ with } \xi_{ij}(h_2) \geq 0\,.$$

[4] A function g of a random variable is law-invariant if $g(X) = g(Y)$ for any two random variables X, Y having the same distribution w.r.t. the underlying probability measure. See e.g. [13].

[5] Although we say "classifier", we are omitting the decision thresholds t_1, t_2 since they are not necessary for AUC and bAUC.

Finally, assume that h_1 produces correct rankings more confidently than h_2, where

$$(c)\ \forall\ (i,j) \text{ such that } \xi_{ij}(h_1) < 0 \text{ we have that } \xi_{ij}(h_1) < \min_{s,k} \xi_{sk}(h_2)\ .$$

From (a), we see that both classifiers will have equal AUC. Then from (b), we see that the classifiers have identical distributions of errors greater than or equal to zero. But finally, considering (c) reveals that the distribution of negative errors (i.e. the correct rankings) for h_1 is more favorable than that of h_2. Thus, we see that h_1 is superior w.r.t. ranking ability. The AUC metric does not reveal this fact, since both classifiers have equal AUC. The bAUC metric, though, because of the *buffer*, correctly distinguishes between the ranking ability of h_1 and h_2. Specifically, we will find that $bAUC(h_1) > bAUC(h_2)$ with the *buffer* accounting for the magnitude of errors not only in (b), but also in (c).[6]

Illustrating a similar situation, not necessarily involving the buffer but instead involving bAUC's sensitivity to the magnitude of positive ranking errors, consider again two classifiers, h_1 and h_2, with equal AUC (i.e. satisfying (a)). Assume also that both classifiers produce correct rankings with the same magnitude (i.e. confidence), meaning that

$$(d)\ \ \xi_{ij}(h_1) = -1\ \forall\ (i,j) \text{ with } \xi_{ij}(h_1) < 0 \text{ and } \xi_{ij}(h_2) = -1\ \forall\ (i,j) \text{ with } \xi_{ij}(h_2) < 0\ .$$

Finally, assume that h_2 produces incorrect rankings more severe than those produced by h_1, where

$$(e)\ \forall\ (i,j) \text{ such that } \xi_{ij}(h_2) \geq 0 \text{ we have that } \xi_{ij}(h_2) > \max_{s,k} \xi_{sk}(h_1)\ .$$

From (a), we see that both classifiers will have equal AUC. Then from (d), we see that the classifiers have identical distributions of errors less than zero. But finally, considering (e) reveals that the distribution of errors larger than or equal to zero (i.e. the incorrect rankings) for h_1 are more favorable than that of h_2. Once again, AUC indicates that these classifiers perform equivalently with respect to ranking ability. The bAUC metric, though, by considering the magnitude of errors, is able to properly distinguish between the two classifiers. Specifically, because of (d) and (e), we will have that $bAUC(h_1) > bAUC(h_2)$.[7]

We note that, for both examples, it is possible to detect the difference in ranking error distributions without bAUC. For example, one could measure $P(\xi(h) \leq t)$ multiple times, over a range of different $t \in \mathbb{R}$, for each classifier h. With bAUC, though, we have a single metric that can reveal this difference and only needs to be measured once for each classifier.

4.2.1 Remark on Use Cases

In the Section 4.1 examples, the data and error distributions were artificial. Thus, we did not prescribe any real-world meaning to the magnitudes of ranking errors. In practical applications, there are cases where these errors do have meaning which can be important when assessing the performance of a classifier. In particular, we can consider applications in which the score given to a data point $h(X)$ gives some indication of predictive confidence. For example, it is common to have $h(X)$ represent a calibrated class probability estimate, see e.g. Platt Scaling from [26] or Isotonic Regression from [41, 42]. Here, the magnitude of $h(X)$ is reflective of the degree of belief that X belongs to a particular class. Thus, the magnitude of ranking errors will serve as an indicator of the quality of these degrees of belief, with large ranking errors indicating that a distribution of scores may lead to poor class probability estimates or provide misleading degrees of belief regarding class membership.

Consider, first, the task of disease detection. This is an application in which AUC is very popular. First, AUC does not make any assumptions about misclassification costs, which can be difficult or unethical to prescribe in medical applications. Second, it does not need a classification decision threshold to assess

[6] We do make the assumption that $bAUC(h_1) \neq 0$, which is to assume that $E[\xi(h_1)] < 0$.

[7] Again, we make the assumption that $bAUC(h_1) \neq 0$, which is to assume that $E[\xi(h_1)] < 0$.

performance. In medical applications, it may be necessary to specify the decision threshold after the model (or scoring function) has been developed. For example, it may be desirable to fix the proportion of false positives or false negatives, but the desired proportion may change over time, across applications, or may not be available when devising the model (see e.g. [16] for a discussion of different threshold choice methods). Both of these beneficial characteristics are present with bAUC as well, i.e. no need for specifying misclassification costs or decision thresholds. Thus, bAUC is a justifiable compliment to AUC in this case.

For disease detection, though, the score given to a patient $h(X)$ may signify the predicted severity of the disease (e.g. the size of a tumor), or may be the predicted probability of class membership. In this case, the magnitude of a ranking error $\xi_{ij}(h)$ has significance. For example, a large ranking error $\xi_{ij}(h) > 0$, as opposed to a small one, could indicate that the model will assign a diseased patient with a very low probability of being diseased. Used in practice, this low probability estimate could lead to far fewer additional tests or preventative measures that would have otherwise been taken if the predicted probability was not so low.

This argument also applies to financial applications. For example, consider credit default prediction. This is also an application in which AUC is very popular given its lack of assumptions regarding misclassification costs and decision thresholds. Just as in disease prediction, misclassification costs and a proper decision threshold may change over time, vary from application to application, or may be unknown when devising a model. Additionally, scores given by a classifier may be predicted probabilities of default. Furthermore, these probabilities may be considered when calculating the size of the loan made available to the individual. Thus, if a ranking error $\xi_{ij}(h) > 0$ is large, the classifier may not only predict that a 'default' customer is actually worthy of a loan, but subsequent uses of the associated predicted probability may indicate that this individual is worthy of a fairly large loan. In other words, if the scores $h(X)$ are indicative of the models predictive confidence, this information may be used in future modeling tasks, and bAUC considers this aspect of performance while AUC does not.

4.3 bAUC and the ROC curve

As discussed in Section 2.1, AUC can also be defined as the area under the ROC curve. We show here that bAUC can also be represented as the area under a slightly modified ROC curve, which we call the Buffered ROC (bROC) curve.

Proposition 3. *For a fixed scoring function $h : \mathbb{R}^n \to \mathbb{R}$, assume that*

$$bAUC(h) = 1 - \alpha = 1 - \frac{E[\xi(h) - \gamma^*]^+}{-\gamma^*} \text{ where } \gamma^* \in \operatorname*{argmin}_{\gamma<0} \frac{E[\xi(h) - \gamma]^+}{-\gamma},$$

and γ^* is the smallest minimizer (i.e. the left endpoint of the argmin interval). Then,

$$bAUC(h) = \int_t P\left(h(X^+) \geq t - \gamma^*\right) dP\left(h(X^-) > t\right).$$

Proof. This follows from Proposition 1, specifically the fact that if $z \in (E[X], \sup X)$ then for any $\gamma_0 \in [q_\alpha(X), q_\alpha^+(X)]$ we have $\gamma_0 \in \operatorname*{argmin}_{\gamma<z} \frac{E[X-\gamma]^+}{z-\gamma}$ and thus for the left-endpoint of this interval, $\gamma^* = q_\alpha(X)$, we have that $P(X > \gamma^*) = \bar{p}_z(X)$. Applying this to bAUC, we get that

$$\begin{aligned} bAUC(h) &= 1 - \bar{p}_0\left(-h(X^+) + h(X^-)\right) = 1 - P(-h(X^+) + h(X^-) > \gamma^*) \\ &= P(-h(X^+) + h(X^-) \leq \gamma^*) = P(h(X^+) + \gamma^* \geq h(X^-)) \\ &= \int_t P\left(h(X^+) \geq t - \gamma^*\right) dP\left(h(X^-) > t\right). \end{aligned}$$

□

Proposition 3 is shown graphically in Figure 3 with a slightly modified ROC plot. Here, instead of plotting the pairs $(P(h(X^-) > t), P(h(X^+) > t))$ for all thresholds $t \in \mathbb{R}$ to generate an ROC curve, we plot the pairs $(P(h(X^-) > t), P(h(X^+) \geq t - \gamma^*))$ for all thresholds $t \in \mathbb{R}$ to generate the bROC curve. This yields a curve on the modified ROC plot that has area underneath it equal to $bAUC(h)$.

We can also interpret the bROC curve as the normal ROC curve of a more conservative scoring function $\hat{h}$ with positive class score distribution $\hat{h}(X^+) = h(X^+) + \gamma^*$ and negative class score distribution $\hat{h}(X^-) = h(X^-)$.[8] We have shifted the distribution of positive class scores toward the distribution of negative class scores by the amount γ^*. By Proposition 4, we know that $\gamma^* = \{z \in \mathbb{R} \mid E[\xi(h)|\xi(h) > z] = 0\} < 0$. Therefore, since γ^* is determined by the tail of the error distribution $\xi(h)$, the bROC curve is the ROC curve of a conservative variant of the original scoring function, where the magnitude of conservatism is based upon the ranking errors produced by the scoring function h.

We can also use this to interpret bAUC as a more conservative version of AUC that considers, more heavily, the extreme errors while additionally considering the worst pairs of points that were *correctly* ranked. In other words, while AUC considers only the count of errors, bAUC conservatively considers the severity of the worst incorrectly ranked pairs of points and, to possibly balance this conservatism out, it considers the confidence with which the classifier correctly ranked the pairs of points that are most poorly ranked, but still correct. For example, consider a classifier with AUC= .7, where $m^+m^- = 100$ and the ordered list of ranking errors is $\{\xi^{(1)}, \xi^{(2)}, ..., \xi^{(100)}\}$, where $\xi^{(1)} \leq \xi^{(2)} \leq ... \leq \xi^{(100)}$. This means that 30 pairs were incorrectly ranked with ranking errors larger than or equal to zero, i.e. $\xi^{(69)} < 0 \leq \xi^{(70)} \leq \xi^{(71)} \leq ... \leq \xi^{(100)}$. Now, if $\sum_{i=70}^{100} \xi^{(i)} = -\xi^{(69)}$, we will have that bAUC= .69. Thus, in this case, although bAUC is more conservative than AUC, we find that the classifier correctly ranked 69 of the 100 pairs with enough confidence to offset the magnitude of the 30 errors from the 30 incorrectly ranked pairs. If, though, the errors less than, but most near to zero, were closer to zero, we would find that bAUC would be smaller, reflecting the conservatism w.r.t. the extreme errors.

4.4 Optimizing bAUC

Direct maximization of AUC is rarely done due to the troublesome properties of probabilistic objectives, even for the simplest classifier such as the linear classifier $h(X) - t = w^T X - t$, $w \in \mathbb{R}^n$. Direct maximization of bAUC, on the other hand, reduces to convex programming and linear programming for the linear classifier. Let $\xi(w) = -w^T(X^+ - X^-)$. Maximization of AUC takes the form,

$$\max_{w \in \mathbb{R}^n} \quad 1 - P(\xi(w) \geq 0) \ , \tag{18}$$

where the probabilistic objective is discontinuous and non-convex when dealing with empirical observations of X^+ and X^-. Maximization of bAUC takes the form

$$\max_{w \in \mathbb{R}^n} 1 - \bar{p}_0(\xi(w)) = 1 - \min_{w \in \mathbb{R}^n} \bar{p}_0(\xi(w)) \ . \tag{19}$$

Applying Proposition 1, and given the positive homogeneity of $\xi(w)$, (19) becomes

$$1 - \min_{w \in \mathbb{R}^n, \gamma < 0} \frac{E[\xi(w) - \gamma]^+}{-\gamma} = \min_{w \in \mathbb{R}^n} E[\xi(w) + 1]^+ \ . \tag{20}$$

In financial optimization literature, the function $E[.]^+$ is called *Partial Moment*. It is a very popular function in various applications of stochastic programming. Here, (20) is a convex optimization problem and, moreover, can be reduced to linear programming with reduction to (21) via auxiliary variables. Thus, in

[8] The choice of score distribution to shift is arbitrary. One can shift $h(X^-)$ by $-\gamma^*$.

the case of a linear classifier, maximizing bAUC is substantially easier to handle than AUC maximization, a non-convex and discontinuous optimization problem.

$$\begin{aligned} \min_{w\in\mathbb{R}^n,\beta_{ij}\in\mathbb{R}} \quad & \frac{1}{m^+m^-}\sum_{i=1}^{m^+}\sum_{j=1}^{m^-}\beta_{ij} \\ s.t. \quad & \beta_{ij}\geq \xi_{ij}(w)+1,\ \forall\, i=1,...,m^+, j=1,...,m^- \\ & \beta_{ij}\geq 0\,. \end{aligned} \tag{21}$$

In practical applications, of course, the linear classifier found by solving (20) or (21) may not generalize to unseen samples (i.e. a testing set separate from a training set). This is why it is common in machine learning applications to include some type of regularization or non-linear classification algorithm. In Section 6.1, we cover one method for adding SVM-like regularization. We mention, though, that formulation (20) or (21) may still be useful. It is increasingly common in applied machine learning to engineer features for linear models[9] and/or to perform an iterative feature selection methodology. Thus, this simple linear formulation would be highly applicable in this context and may generalize quite well if features are engineered and selected properly for the application domain. We do not fully explore this use of (20) or (21), as the problem of feature engineering and selection is highly problem dependent.

Furthermore, we mention that, even if one is only interested in maximizing AUC, one should consider bAUC as a good surrogate. It is often desirable for the same reasons as AUC, e.g. no dependence on decision threshold or misclassification costs, but it is far easier to optimize, yielding a convex objective. Thus, it may turn out that direct optimization of bAUC is the best heuristic for optimization of AUC in particular applications. This is explored further in Section 6, as we show that existing attempts to approximately optimize AUC in the SVM context are equivalent to direct maximization of bAUC. Moreover, this would make sense in light of Proposition 2, with bAUC acting as the tightest lower bound of AUC that is concave.

4.5 Software Implementation

4.5.1 Calculating bAUC and the bROC Curve

For interested readers, Python code is available for calculating bAUC and plotting the bROC curve.[10] We mention, also, that this code contains two implementations for calculating empirical bAUC from a sample. The first is a simple method which calculates empirical bAUC *approximately*, growing more accurate as sample size grows. For this, one first creates a list of losses, $\xi_{ij}(h)$, ordered from smallest to largest. Then, beginning at the largest loss, one simply takes a running average, moving from largest to smallest loss, stopping when the average becomes less than or equal to zero. Assuming there are m^+m^- losses in the list, and that the running average contained the $\hat{m}$ largest losses, we have that bAUC is approximately equal to $1-\frac{\hat{m}}{m^+m^-}$. Specifically, if the average of the worst $\hat{m}$ losses is zero, you will get empirical bAUC exactly with no error. If the average of the worst $\hat{m}$ losses is less than zero (i.e. the average of the worst $\hat{m}-1$ losses is greater than zero), then this procedure will under-estimate empirical bAUC by at most $\frac{1}{m^+m^-}$.

The second method calculates empirical bAUC exactly by solving the following convex program for fixed h,

$$\min_{a\geq 0} \quad \frac{1}{m^+m^-}\sum_{i=1}^{m^+}\sum_{j=1}^{m^-}[a\xi_{ij}(h)+1]^+\,. \tag{22}$$

[9] A simple example would be the bag-of-words model, while an advanced example would be the linear SVM pipeline described in [39].

[10] http://www.ise.ufl.edu/uryasev/research/testproblems/advanced-statistics/case-study-bAUC-maximization/

Just as we did for (20), this convex program can be recast as a linear program with additional auxiliary variables. The benefit of this formulation is that it solves for empirical bAUC *exactly*. Using any convex or linear programming solver (e.g. we use Gurobi within our Python code), this can be quickly solved for moderate sample sizes. The drawback is that for large sample sizes, there are m^+m^- terms in the objective sum, and an additional m^+m^- constraints when reformulated as a linear program. Thus, many commercially available solvers struggle when sample size grows, specifically suffering from memory issues. However, we point out that solvers are available which exploit the problem structure of (22), avoiding issues with the $\mathscr{O}(m^+m^-)$ memory requirement. Portfolio Safeguard (PSG)[11] is one such software, for which an academic license can be obtained for free. For the interested reader, we have posted example PSG code on the same page where Python code is available.

4.5.2 bAUC Optimization for Linear Classifiers

For linear classifier $h(X)-t = w^TX-t$, $w \in \mathbb{R}^n$ the maximization of bAUC reduces to bPOE minimization for linear function $\xi(w) = -w^T(X^+ - X^-) = -w^TX^+ + wX^-$, see (19) and (20). Although this minimization reduces to convex programming w.r.t. decision variables, the implementation of bAUC optimization is non-trivial. Specifically, the LP representation (21) for bAUC optimization has $\mathscr{O}(m^+m^-)$ constraints. Because of this, as already mentioned, many commercial optimization packages may struggle to handle bAUC optimization efficiently.

From a practical point of view, if bAUC is to be effectively utilized in experimentation, it is critical that there exist a highly efficient implementation for bAUC calculation and optimization. Software should take into account the special structure of the optimization problem. Portfolio Safeguard (PSG) is one such software, having specialized routines for Partial Moment and bPOE minimization. The Partial Moment and bPOE function, as well as many other stochastic functions, are precoded allowing the user to include them in analytic format in optimization problem statements. Because of this precoding, PSG can efficiently invoke specially developed algorithms for these analytic expressions. For the interested reader, a PSG case study using the precoded partial moment function can be found online.[12] This case study provides data sets and PSG codes for MATLAB, R, and Run-File (text) environments.? PSG is free for academic users (data and source codes can be downloaded from the website containing the case study).

The code can handle large sample sizes that would typically cause memory issues with other commercially available solvers. For example, Problem 1 in the case study minimizes the partial moment with $m^+ = 3,990$, $m^- = 2,788$ in 0.13 second on a 3.14GHz PC. This problem is equivalent to a bPOE minimization problem with $m^+m^- = 11,124,120$ scenarios. In Table 1, we include additional runtimes for larger versions of this case study. We report the average and standard deviation, over ten trials, of the time needed to solve problem statement (19) and calculate (22) at the optimal solution for two data sets. The first data set is purely artificial, composed by generating m^+ vectors of dimension 10 with components uniformly distributed on the interval $[-.25,.75]$ and m^- vectors of dimension 10 with components uniformly distributed on the interval $[0,1]$. We chose this intersection of intervals so that bAUC calculation and optimization would not be trivial with bAUC$=1$. The second data set is generated from the Ionosphere data set from [1]. The original data set is composed of vectors with dimension 34 and $m^+ = 126$ and $m^- = 225$. We then artificially increase the sample size of the data set by adding new samples which are perturbed versions of the original samples. If $X = [X^{(1)},...,X^{(n)}]$ is an original n-dimensional sample point, its perturbed version X_p will have features $X_p^{(i)} = X^{(i)} + S_i(2U-1)$ where U is a random number from uniform distribution $\mathscr{U}(0,1)$ and S_i is the standard deviation of the i^{th} feature across the entire original data set.

[11] www.AORDA.com

[12] http://www.ise.ufl.edu/uryasev/research/testproblems/advanced-statistics/case-study-bAUC-maximization/

m^+	m^-	Avg Time to Solve (20)	Avg Time to Calculate (22)
Artificial Data			
10,000	10,000	.5794 ± .0197 seconds	.5348 ± .0158 seconds
100,000	100,000	1.667 ± .0515 seconds	.8637 ± .0118 seconds
1,000,000	1,000,000	13.5325 ± .8267 seconds	4.6658 ± .1313 seconds
Perturbed Ionosphere Data			
126	225	.2631 ± .1092 seconds	.1636 ± .0225 seconds
1,260	2,250	.2861 ± .0275 seconds	.1639 ± .0017 seconds
12,600	22,500	1.768 ± .2759 seconds	.3596 ± .0114 seconds
126,000	225,000	25.6364 ± 3.5174 seconds	3.9105 ± .2751 seconds

Table 1: Size of data sets and time needed to solve problem statement (20) and calculate (22) using Portfolio Safeguard by AORDA.com.

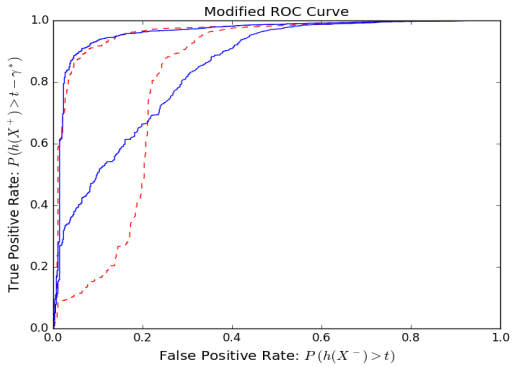

Fig. 4: Solid line corresponds to logistic regression with AUC=.962 and bAUC=.844, dashed line is SVM with AUC=.961 and bAUC=.810.

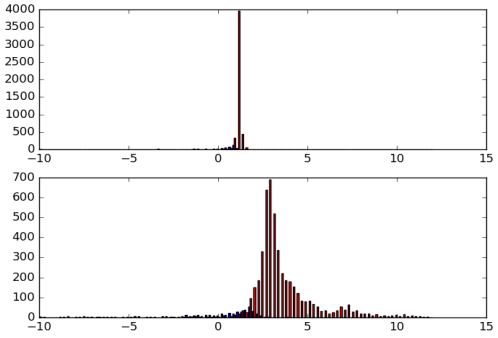

Fig. 5: Histogram of scores provided by classifiers. Upper chart is SVM scores. Lower chart is logistic regression log odds.

5 Empirical Examples

5.1 bAUC and bROC Curves

Here, much like the theoretical example provided in Section 4.2, we show that bAUC can reveal classifier properties which are not reflected by AUC but with a real data set and classifiers. We also show that while the ROC curves may be almost identical, the bROC curves can be different, offering additionally discriminatory insights to supplement the summary insight of the bAUC metric. Additional examples illustrating more cases can be found in the extended version of this paper.[13]

For the experiment, we compared two classifiers (SVM and logistic regression[14]) trained on the Page-Blocks dataset from the UCI repository. Figure 4 shows the ROC and bROC curves for these classifiers as well as their AUC and bAUC values. First, notice that for these two classifiers, AUC values and ROC curves are almost identical, providing little discriminatory insight to compare classifiers. Looking at bAUC values, we see that the logistic regression classifier has larger bAUC. Additionally, we see that the bROC curves are dramatically different. Looking at the slope of the bROC curves, we can see that the SVM classifier is quit unstable, with the large slope revealing that the score distribution is highly concentrated on a small threshold interval. We can demonstrate this fact by looking at the score distribu-

[13] http://www.ise.ufl.edu/uryasev/publications/

[14] Scores $h(X)$ of logistic regression in this example are the log odds.

tions $h(X)$ themselves in Figure 5. Clearly, the logistic regression classifier produces a more stable score distribution with respect to threshold changes, as the SVM has a score distribution that is very sensitive to changes in decision threshold because of the concentration of scores in a very small range.

5.2 Empirical comparison with similar AUC alternatives

In [40] and [12], two AUC counterparts are proposed that are similar to bAUC in that they consider the magnitude of scores $h(X)$ and ranking errors $\xi_{ij}(h)$. [40] proposes the *scored* AUC (sAUC) while [12] proposes the *probabilistic* AUC (pAUC) which are formulated as follows. Note that these are only defined in the discrete case where we assume that scores have been normalized so that $h(X_i) \in [0,1]$, $i = 1,...,N$.

$$sAUC(h) = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} I_{h(X_i^+) > h(X_j^-)} (h(X_i^+) - h(X_j^-)) + .5 I_{h(X_i^+) = h(X_j^-)}$$

$$pAUC(h) = \frac{1}{2} \left(\frac{\sum_{i=1}^{m^+} h(X_i^+)}{m^+} - \frac{\sum_{j=1}^{m^-} h(X_i^-)}{m^-} + 1 \right) .$$

To compare the performance of these different metrics, we performed the following experimental setup. For each of 13 data sets from the UCI machine learning repository, we split it into 90 percent training and 10 percent testing. We then trained multiple classifiers[15] (Adaboost, Linear SVM, Random Forest, Quadratic Discriminant Analysis, Naive Bayes, K-Nearest Neighbor, Neural Net) performing 5 fold cross-validation on the training set for each classifier and then selecting the best classifier w.r.t. each performance metric (AUC, bAUC, sAUC,and pAUC) in terms of the mean 5-fold out-of-sample performance. We then trained the best classifiers on the full training set and measured the performance on the test set. We repeated this experiment 20 times, using different random splits of the data set. The average AUC of the classifiers are listed in Table 2. Numbers in bold indicate that the classifier is statistically no different than the optimal classifier according to a paired t-test at significance level .05. Due to lack of space, we did not report results for other metrics. However, we found that the bAUC maximizing classifier frequently outperformed the sAUC and pAUC maximizing classifiers in terms of bAUC, Accuracy, F-Measure, and MCC. Therefore, at least for our particular experimental setup, using bAUC as our model selection performance metric led to superior performance. Additionally, for this experimental setup, we find that bAUC acts much more similarly to AUC than sAUC or pAUC. Often, we found that bAUC and AUC are optimal for the same classifier.

6 Generalized bAUC: Utilizing Non-zero Thresholds

Previously, we considered the definition of bAUC to be one minus the buffered probability of the error function $\xi(h)$ exceeding the threshold of $z = 0$ (i.e definition (2)). Consider now a more general definition of bAUC with thresholds $z \in \mathbb{R}$.

Definition 3. *Generalized bAUC is defined as follows,*

$$bAUC_z(h) = 1 - \bar{p}_z(\xi(h)) .$$

[15] All classifiers were programmed in Python with scikit-learn. ([25])

Data Set	bAUC	AUC	pAUC	sAUC
german credit (statlog)	**.793**	**.794**	.761	.592
spambase	**.986**	**.985**	.982	.898
pima	**.827**	**.826**	**.81**	.666
yeast	**.794**	**.795**	.734	.628
liver disorders	**.749**	**.75**	.671	.597
wpbc	**.605**	**.606**	**.594**	**.595**
ionosphere	**.973**	**.97**	**.955**	.881
wine red4	**.748**	**.783**	.691	.526
wine white4	**.855**	**.868**	.761	.614
abalone19	**.779**	**.812**	**.71**	.511
solarM	**.792**	**.793**	.777	.625
page-blocks	**.99**	**.99**	.945	.89

Table 2: Mean AUC for classifiers optimal w.r.t. each performance metric over 20 splits. Bold indicates that the model is no worse than the best model as measured by a paired t-test at significance level .05.

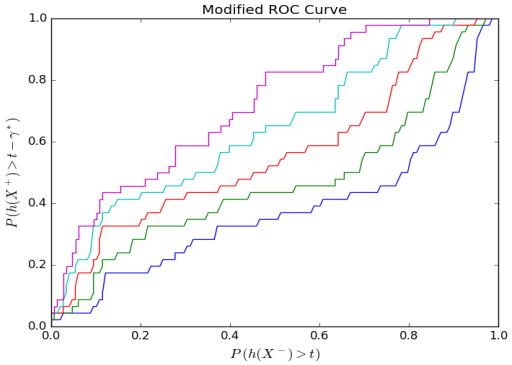

Fig. 6: A modified ROC plot for a fixed classifier h. The lower most curve corresponds to $bAUC_0(h)$ while the uppermost curve corresponds to $bAUC_{z_0}(h) = AUC(h)$. The curves in-between correspond to $bAUC_z(h)$ for values of $z \in (0, z_0)$.

Just as bAUC was shown to correspond to the area under a modified ROC curve, we have that $bAUC_z$ for any $z \in \mathbb{R}$ corresponds to the area under a curve on the same, modified ROC plot. This generates a family of ROC curves, in which AUC and bAUC are members. Specifically, we have the following proposition, the proof of which we omit since it is essentially identical to that of Proposition 3.

Proposition 4. *Assume that* $bAUC_z(h) = 1 - \alpha = 1 - \frac{E[\xi(h) - \gamma^*]^+}{z - \gamma^*}$ *where* $\gamma^* \in \underset{\gamma < z}{argmin} \frac{E[\xi(h) - \gamma]^+}{z - \gamma}$ *and* γ^* *is the smallest minimizer (i.e. the left endpoint of the argmin interval). Then,*

$$bAUC_z(h) = \int_t P\left(h(X^+) \geq t - \gamma^*\right) dP\left(h(X^-) > t\right) .$$

Notice in Proposition 4, that if we choose z_0 such that $\gamma^* = 0$, we will have $bAUC_{z_0}(h) = AUC(h)$. Thus, we see that AUC belongs to the family of curves associated with $bAUC_z$, $z \in \mathbb{R}$. Showing this on the ROC plot, we have Figure 9 which displays a family of $bAUC_z$ curves.

6.1 Maximizing Generalized bAUC

Given Generalized bAUC, it is not immediately clear how to utilize it. Here, we show that Generalized bAUC has already been utilized successfully for AUC maximization, albeit not explicitly. Specifically,

we find that the popular AUC maximizing RankSVM from [4, 15] is equivalent to a special case of direct maximization of Generalized bAUC. We first provide a formulation for maximizing $bAUC_z$ and then show that the AUC maximizing RankSVM is a special case of this formulation (specifically, for threshold range $z \leq 0$). In this context, we work with $h(X) = w^T X$ and ranking error $\xi(w) = -w^T(X^+ - X^-)$.

Consider the problem of finding the vector $w \in \mathbb{R}^n$ which maximizes $bAUC_z(w)$. In other words, we would like to solve the following optimization problem.

$$\min_{w \in \mathbb{R}^n, \gamma < z} \quad \frac{E[\xi(w) - \gamma]^+}{z - \gamma} \equiv \quad \min_{w \in \mathbb{R}^n} \bar{p}_z(\xi(w)) \ . \tag{23}$$

However, this problem is ill-posed. As was shown in [24], this formulation yields trivial solutions for thresholds $z \neq 0$ due to the positive homogeneity of the error function $\xi(w)$ (see appendix of [24] for details). This issue, though, can be alleviated by fixing the scale of the vector w. This can be accomplished by fixing any general norm on w, effectively minimizing bPOE of the *normalized* error distribution $\frac{\xi(w)}{\|w\|}$. Thus, we can consider the following optimization problem which maximizes $bAUC_z$ for non-zero thresholds, where $\|\cdot\|$ is any general norm,

$$\begin{aligned} \min_{w \in \mathbb{R}^n, \gamma < z} \quad & \frac{E[\xi(w) - \gamma]^+}{z - \gamma} \equiv \quad \min_{w \in \mathbb{R}^n} \bar{p}_z \left(\frac{\xi(w)}{\|w\|} \right) \ . \\ s.t. \quad & \|w\| = 1 \end{aligned} \tag{24}$$

Furthermore, using the result from [24] we know that to maximize $bAUC_z$, we can alternatively solve the following equivalent problem, which is convex for thresholds $z \leq 0$,

$$\min_{w \in \mathbb{R}^n} \bar{p}_z \left(\frac{\xi(w)}{\|w\|} \right) \quad \equiv \quad \min_{w \in \mathbb{R}^n} E[\xi(w) - z\|w\| + 1]^+ \ . \tag{25}$$

The last formula is easy to interpret. Specifically, adapting a result from [24], we have the following proposition.

Proposition 5. *For $z \in \mathbb{R}$, assume that*

$$1 - \alpha^* = \min_{w \in \mathbb{R}^n} E[\xi(w) - z\|w\| + 1]^+ = E[\xi(w^*) - z\|w^*\| + 1]^+ \ .$$

Then for the normalized error, $F := \xi\left(\frac{w^*}{\|w^*\|}\right)$, at the optimal point w^*:

$$\bar{p}_z(F) = 1 - \alpha^*, \quad \bar{q}_{\alpha^*}(F) = z, \quad q_{\alpha^*}(F) = z - \frac{1}{\|w^*\|} \quad .$$

In the next section, after showing that (25) and the RankSVM are equivalent over the parameter range $z \leq 0$, we find that Proposition 5 provides us with a novel interpretation for the optimal objective value and free parameter of the RankSVM.

6.2 RankSVM Maximizes Generalized bAUC

In [4, 15], the AUC maximizing RankSVM (also called the AUC-SVM) is derived and shown to maximize AUC better than the traditional max-margin SVM's proposed by [8]. More recently, [36] similarly showed that the RankSVM outperforms many variants of traditional SVM's, as well as a k-nearest neighbor classifier, particularly when class sizes are imbalanced. In addition, building upon the efficient implementation of [6], they also show that the RankSVM can be scaled to large data sets without a decrease

in performance by using a rare-class kernel representation. Thus, the RankSVM has shown continued success in optimizing AUC, backed up by considerable empirical evidence along the way.

Utilizing a result from [24], we can show that RankSVM is equivalent to direct maximization of Generalized bAUC for thresholds $z \leq 0$. This serves to show in a more exact manner that the AUC maximizing SVM is, in fact, maximizing a lower bound on AUC, specifically Generalized bAUC. This equivalence also suggests a novel interpretation for the optimal objective value of the RankSVM and the free parameter. Furthermore, because of this equivalence, the considerable empirical evidence in the literature confirming that the RankSVM is highly effective for maximizing AUC also confirms that maximizing bAUC is an efficient heuristic method for maximizing AUC.

The RankSVM is formulated as follows, where $\lambda \geq 0$ is typically introduced as a parameter specifying the tradeoff between ranking error and regularization. Traditionally, the squared L_2 norm is used, but we use any general norm.

$$\min_{w} \quad \lambda \|w\| + \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} [\xi_{ij}(w) + 1]^+ . \tag{26}$$

This is a reformulation of the well known C-SVM of [8], reformulated for AUC maximization. Let $Y_i \in \{-1,+1\}, i = 1,..,N$ indicate the class of samples $X_1,...,X_N$, let $\lambda \geq 0$, and let $(w,b) \in \mathbb{R}^{n+1}$. The C-SVM is formulated as follows,

$$\min_{w,b} \quad \lambda \|w\| + \frac{1}{N} \sum_{i=1}^{N} [-Y_i(w^T X_i + b) + 1]^+ . \tag{27}$$

Relating the C-SVM to bPOE minimization, [24] introduced the EC-SVM formulation, which is identical to bPOE minimization problem (25) but with error function $\xi(w,b) = -Y(w^T X + b)$. The EC-SVM is formulated as follows, where $z \in \mathbb{R}$.

$$\min_{w,b} \quad E[-Y(w^T X + b) - z\|w\| + 1]^+ . \tag{28}$$

Specifically, the EC-SVM and C-SVM were related through the following proposition which shows that the traditional soft margin SVM of [8] is equivalent to minimizing bPOE.

Proposition 6. *Consider (27) and (28) formulated with the same norm and assume that we have N equally probable samples (X_i, Y_i), $i = 1,...,N$. Then, over the parameter range $\lambda \geq 0$, $z \leq 0$, (27) and (28) achieve the same set of optimal solutions.*

Using Proposition 6, we can prove that RankSVM is simply maximizing Generalized bAUC.

Proposition 7. *Consider (26) and (25) formulated with the same norm and assume that we have $m^+ m^-$ equally probable realizations of the random error $\xi_{ij}(w)$, $i = 1,...,m^+$, $j = 1,...,m^-$. Then, over the parameter range $\lambda \geq 0$, $z \leq 0$, (26) and (25) achieve the same set of optimal solutions.*

Proof. Note that (26) is exactly formulation (27) with $m^+ m^-$ samples $(X_i^+ - X_j^-)$, $i = 1,...,m^+$, $j = 1,...,m^-$ all having class $Y_{ij} = +1$ and with the classifier intercept $b = 0$. Thus, applying Proposition 6, we have that (26) and (25) produce the same set of optimal solutions over the parameter range $\lambda \geq 0$, $z \leq 0$. □

With this equivalence, we can draw a novel interpretation of the free parameter of the RankSVM and its optimal objective from Property 5. Specifically, we can now interpret the trade off parameter $\lambda \geq 0$ as Generalized bAUC threshold (i.e. bPOE threshold). Additionally, we can conclude that one minus the optimal objective of the RankSVM equals a probability level, specifically Generalized bAUC for some z. We do not present this formally here, as it follows directly from [24] and the analysis of the dual formulations of (27) and (28).

As we will see in the next section, the insight that the RankSVM λ parameter can be interpreted as bPOE threshold has consequences from a theoretical perspective. We provide generalization bounds for the true AUC and bAUC of RankSVM classifiers and show that these bounds are controlled by empirical bAUC, sample size, and λ. Thus, our bounds are not controlled by a measure of the complexity of the hypothesis space, which is typical in statistical learning theory, but are controlled instead by bPOE threshold. In other words, we see that consideration of non-zero bPOE thresholds leads to generalization.

6.3 Rank SVM Generalization Bounds For bAUC

With the new insight that the RankSVM directly maximizes bAUC, we show here that the classifier produced by the RankSVM generalizes w.r.t. bAUC when trained on a finite sample. Specifically, we use the stability arguments of [2] to provide generalization bounds on bAUC for optimal solutions of the RankSVM. This provides theoretical guarantees on the performance of RankSVM classifiers w.r.t. bAUC. Thus, RankSVM is shown to directly maximize bAUC and generalize w.r.t. this performance metric as well. It is important to note that these performance guarantees also hold for AUC, since bAUC is a lower bound on AUC.

These bounds diverge from traditional generalization bounds given in the classification literature in two ways. First, generalization bounds are typically focused on bounding the true misclassification rate. This intuitively makes sense. With classification algorithms, like the SVM, often viewed as approximate methods for minimizing misclassification rate, we would like theoretical guarantees that our classifier will perform well on unseen samples w.r.t. the misclassification rate performance metric. Having shown that the RankSVM is directly maximizing bAUC, we instead provide a bound on the true bAUC, showing that RankSVM classifiers perform well on unseen samples in terms of the bAUC performance metric.

Second, generalization bounds are often composed of a sum of empirical error and an additional uncertainty term. This uncertainty term is often controlled by the sample size and a measure of the complexity of the hypothesis space from which the classifier was selected, e.g. the VC dimension. Our bAUC generalization bound is similar in that it is a sum of empirical bAUC and an additional uncertainty term. However, our uncertainty term is not controlled by a measure of the complexity of the hypothesis space. It is controlled by sample size and bPOE threshold.

Specifically, assume we have a finite training set $S = S^+ \bigcup S^- = \{X_1^+, ..., X_{m^+}^+\} \bigcup \{X_1^-, ..., X_{m^-}^-\}$ and a training algorithm that produces the scoring function h_S. The *true* bAUC is given by

$$bAUC(h_S) = 1 - \min_{a \geq 0} E[a\xi(h_S) + 1]^+,$$

which is unknown since we do not assume to know the true distribution of X. However, given access to a training set, we can calculate empirical bAUC,

$$\widehat{bAUC(h_S)} = 1 - \min_{a \geq 0} \frac{1}{m^+ m^-} \sum_i \sum_j [a\xi_{ij}(h_S) + 1]^+.$$

We show in Proposition 8 that for the scoring function produced by the RankSVM algorithm, the true bAUC is bounded from below by the sum of empirical bAUC and an uncertainty term. Although this uncertainty term looks complex, it essentially relies on two parts. First, it relies on the size of the sample of the minority class, i.e. $\hat{m} = \min\{m^+, m^-\}$., and the overall sample size $N = m^+ + m^-$. Second, it relies on the RankSVM parameter λ. As was discussed in the previous section, this parameter plays the role of bPOE threshold. Therefore, we see that our generalization bound is not reliant upon a measure of the complexity of the hypothesis space, but instead bPOE threshold. Thus, this shows that bPOE threshold plays an important role in the generalization capabilities of the RankSVM algorithm.

For brevity, we avoid a full introduction to the ideas of uniform stability and the associated generalization theorems and refer readers to [2]. Thus, we confine most details to the proof. However, we note

that this bound, like many of those from [2], can be very tight. Also, we note that Proposition 8 considers the RankSVM, but with squared norm. Although similar bounds hold for RankSVM formulated with non-squared norm and for formulation (25), we present the RankSVM with squared norm in the main proposition because the bound expression is much more straight forward.[16]

For the proposition, let us denote the support of X, X^+, X^- respectively as $\mathscr{X}, \mathscr{X}^+, \mathscr{X}^-$. Additionally, for a norm $\|\cdot\|$ let $\|\cdot\|^*$ denote the dual norm. We also assume that the squared norm is differentiable and r-strongly convex w.r.t. itself.[17] For example, note that the squared L_2 norm is differentiable and strongly convex w.r.t. itself with $r = 2$.

Proposition 8. *Assume that the squared norm $\|\cdot\|^2$ is differentiable and r-strongly convex w.r.t. itself and that $\sup_{x^+\in\mathscr{X}^+}\|x^+\|^* \le R^+$ and $\sup_{x^-\in\mathscr{X}^-}\|x^-\|^* \le R^-$, meaning that the support of X^+, X^- is contained in a dual-norm ball of radius R^+, R^- respectively. Assume we solve the RankSVM with squared norm on training set $S = S^+ \bigcup S^- = \{X_1^+, ..., X_{m^+}^+\} \bigcup \{X_1^-, ..., X_{m^-}^-\}$, yielding*

$$w_S \in \underset{w}{argmin}\ \frac{1}{m^+m^-}\sum_i\sum_j [\xi_{ij}(w)+1]^+ + \lambda\|w\|^2 .$$

Let $\hat{m} = \min\{m^+, m^-\}$, $N = m^+ + m^-$. For any $a \ge 0$, with probability $1-\rho$,

$$bAUC(w_S) \ge 1 - \frac{\sum_i\sum_j [a\xi_{ij}(w_S)+1]^+}{m^+m^-} - \left(\frac{2a(R^-+R^+)^2}{r\lambda\hat{m}} + \left(\frac{4Na(R^-+R^+)^2}{r\lambda\hat{m}} + 1 + \frac{a(R^-+R^+)}{\sqrt{\lambda}}\right)\sqrt{\frac{\ln\frac{1}{\rho}}{2N}}\right) .$$

In particular, if $\bar{a} \in \underset{a\ge 0}{argmin}\ \frac{1}{m^+m^-}\sum_i\sum_j [a\xi_{ij}(w_S)+1]^+$, with probability $1-\rho$,

$$bAUC(w_S) \ge \widehat{bAUC(w_S)} - \left(\frac{2\bar{a}(R^-+R^+)^2}{r\lambda\hat{m}} + \left(\frac{4N\bar{a}(R^-+R^+)^2}{r\lambda\hat{m}} + 1 + \frac{\bar{a}(R^-+R^+)}{\sqrt{\lambda}}\right)\sqrt{\frac{\ln\frac{1}{\rho}}{2N}}\right) .$$

Proof. See Appendix B. □

Although we do not present it here, considering only the case of linear functions $w \in \mathbb{R}^n$, this result can easily be generalized to consider functions in a Reproducing Kernel Hilbert Space such that the kernel function $K(\cdot,\cdot)$ is bounded with $\sup_{X\in\mathscr{X}} K(X,X) \le R$, as is done in [2].

7 Conclusion

AUC is a useful and popular metric for measuring the ranking quality of scores given by a classifier. As a metric defined with POE, though, it does not consider the magnitude of ranking errors and is numerically difficult to optimize. We utilize bPOE to create an informative counterpart metric called bAUC. We show that bAUC is indeed a counterpart. It is a readily optimizable, tight, concave lower bound of AUC that can also be viewed as the area under a modified ROC curve. We also show that bAUC is a more

[16] Intuitively, similar bounds will hold for any equivalent formulation, but with different constants.

[17] A differentiable function f is r-strongly convex w.r.t. some norm $\|\cdot\|$ if for any w_1, w_2 we have that $\frac{r}{2}\|w_1 - w_2\|^2 \le f(w_1) - f(w_2) - \langle\nabla f(w_2), w_1 - w_2\rangle$.

appealing counterpart, theoretically and empirically, than sAUC or pAUC when the magnitude of ranking errors yields important discriminatory information. Additionally, the bROC curve can provide additional discriminatory insights when comparing classifiers, particularly when ROC curves are very similar and cross multiple times.

To facilitate the creation of bAUC, we focused our attention on deriving a novel formula for calculating bPOE, the inverse of the superquantile (CVaR). We show that this formula is significant, allowing certain bPOE minimization problems to be reduced to convex and linear programming. Applying this to bAUC, we show that this reduction applies to bAUC allowing for efficient maximization of the metric.

By considering non-zero bPOE thresholds in the definition of bAUC, we also introduce Generalized bAUC. We show that Generalized bAUC generates a family of metrics, in which AUC and bAUC belong. Furthermore, we show that Generalized bAUC has already found its way into the AUC maximization literature. Specifically, we show that the popular AUC maximizing RankSVM is equivalent to maximization of Generalized bAUC, providing also theoretical performance guarantees for bAUC for classifiers produced by this algorithm. Thus, bAUC has already, in some sense, been used as a metric counterpart to AUC that is much simpler to optimize.

In general, we see that defining metrics with POE is highly intuitive, but produces metrics that are numerically difficult to optimize. Utilizing bPOE, one can create a complimentary counterpart to the intuitive POE metric that reveals information about the magnitude of errors while proving to be efficiently optimizable with convex or linear programming.

Acknowledgements

Authors would like to thank Prof. R.T. Rockafellar and Dr. Alexander Mafusalov for their valuable comments and suggestions. This work was partially supported by USA Air Force Office of Scientific Research grant: "Design and Redesign of Engineering Systems", FA9550-12-1-0427, and "New Developments in Uncertainty: Linking Risk Management, Reliability, Statistics and Stochastic Optimization", FA9550-11-1-0258 as well as the DARPA grant "Risk-Averse Optimization of Large-Scale Multiphysics Systems."

8 Appendix A

Here, we discuss the slight differences between Upper and Lower bPOE. First, Lower bPOE is defined as follows.

Definition 4. *Let X denote a real valued random variable and $z \in \mathbb{R}$ a fixed threshold parameter. bPOE of random variable X at threshold z equals*

$$\bar{p}_z^L(X) = \begin{cases} 0, & \text{if } z \geq \sup X \,, \\ \{1-\alpha | \bar{q}_\alpha(X) = z\}, & \text{if } E[X] < z < \sup X \,, \\ 1, & \text{otherwise.} \end{cases}$$

Upper bPOE is defined as follows.

Definition 5. *Upper bPOE of random variable X at threshold z equals*

$$\bar{p}_z^U(X) = \begin{cases} \max\{1-\alpha | \bar{q}_\alpha(X) \geq z\}, & \text{if } z \leq \sup X \,, \\ 0, & \text{otherwise.} \end{cases}$$

Upper and Lower bPOE do not differ dramatically. This is shown by the following proposition.

Proposition 9.

$$\bar{p}_z^U(X) = \begin{cases} \bar{p}_z^L(X), & \text{if } z \neq \sup X\,, \\ P(X = \sup X), & \text{if } z = \sup X. \end{cases}$$

Proof. We prove four cases.
Case 1: Assume $z > \sup X$. By Definition 1, $\bar{p}_z^L(X) = 0$. By Definition 2, $\bar{p}_z^U(X) = 0$.
Case 2: Assume $E[X] < z < \sup X$. By Definition 1, $\bar{p}_z^L(X) = \{1-\alpha | \bar{q}_\alpha(X) = z\}$. By Definition 2, $\bar{p}_z^U(X) = \max\{1-\alpha | \bar{q}_\alpha(X) \geq z\}$. Since $\bar{q}_\alpha(X)$ is a strictly increasing function of α on $\alpha \in [0, 1-P(X = \sup X)]$, $\bar{q}_\alpha(X) = z$ has a unique solution. Therefore, we have that $\bar{p}_z^U(X) = \max\{1-\alpha | \bar{q}_\alpha(X) \geq z\} = \{1-\alpha | \bar{q}_\alpha(X) = z\} = \bar{p}_z^L(X)$.
Case 3: Assume $z \leq E[X]$, $z \neq \sup X$. By Definition 1, $\bar{p}_z^L(X) = 1$. Since $\bar{q}_0(X) = E[X]$, $\max\{1-\alpha | \bar{q}_\alpha(X) \geq z\} = 1$ implying that $\bar{p}_z^U(X) = 1$.
Case 4: Assume $z = \sup X$. Following from the fact that $\bar{q}_{(1-P(X=\sup X))}(X) = \sup X$, we have that $\bar{p}_x^U(X) = \max\{1-\alpha | \bar{q}_\alpha(X) \geq z\} = P(X = \sup X)$. □

Thus, one will notice that Upper and Lower bPOE are equivalent when $z \neq \sup X$. The difference between the two definitions arises when threshold $z = \sup X$. In this case, we have that $\bar{p}_z^L(X) = 0$ while $\bar{p}_z^U(X) = P(X = \sup X)$. Thus, for a threshold $z \in (E[X], \sup X)$, both Upper and Lower bPOE of X at z can be interpreted as one minus the probability level at which the superquantile equals z. Roughly speaking, Upper bPOE can be compared with $P(X \geq z)$ while Lower bPOE can be compared with $P(X > z)$.

The importance of using Upper bPOE instead of Lower bPOE in the definition of bAUC should be noted here. To illustrate, consider a trivial classifier with $w = 0$. Clearly this is not a very good classifier. Using Upper bPOE, we find that $1 - \bar{p}_0^U(\xi(w)) = 1 - P(\xi(w) = \sup \xi(w)) = 1 - 1 = 0$. Using this number as our ranking ability performance metric intuitively makes sense, i.e. assigning the trivial classifier the lowest possible bAUC, reflecting its poor ranking ability. What if we use Lower bPOE instead? Using Lower bPOE, we find that $1 - \bar{p}_0^L(\xi(w)) = 1 - 0 = 1$. Using this as our measure of ranking ability does not make much sense. Thus, we find that Upper bPOE treats losses at the supremum in a manner more fitting to our application, i.e. measuring the ranking ability of a classifier.

9 Appendix B: Proof of Proposition 8

Let $V(w, X^-, X^+, a) = [-aw^T(X^- - X^+) + 1]^+$. First, note that,

$$\begin{aligned} bAUC(w_S) &= 1 - \min_{a \geq 0} E[-aw_S^T(X^+ - X^-) + 1]^+ \\ &= 1 - \min_{a \geq 0} E[V(w_S, X^-, X^+, a)] \\ &\geq 1 - E[V(w_S, X^-, X^+, a)]\,, \forall\, a \geq 0\,. \end{aligned}$$

Given any $a \geq 0$, we can upper bound $E[V(w_S, X^-, X^+, a)]$ by using stability arguments of [2].

The first step is to prove that the RankSVM is uniformly stable, as defined in [2], w.r.t. loss function V. The second step is to prove that for any training set S and sample pair $(x^-, x^+) \in \mathscr{X}^- \times \mathscr{X}^+$, that $V(w_S, x^-, x^+, a)$ is bounded. We can then apply Theorem 12 from [2].

To prove uniform stability, we show that there exists θ such that given any training set $S = S^+ \bigcup S^-$,

$$\sup_{X^-, X^+} |V(w_S, X^-, X^+, a) - V(w_{S^k}, X^-, X^+, a)| \leq \theta$$

where

$$w_S \in \operatorname*{argmin}_w \frac{1}{m^+ m^-} \sum_i \sum_j [-w^T(X_i^+ - X_j^-) + 1]^+ + \lambda \|w\|^2\,,$$

and

$$w_{S^k} \in \underset{w}{\operatorname{argmin}} \frac{1}{m^+m^-}\sum_i\sum_{j\neq k}[-w^T(X_i^+ - X_j^-)+1]^+ + \lambda\|w\|^2 ,$$

or

$$w_{S^k} \in \underset{w}{\operatorname{argmin}} \frac{1}{m^+m^-}\sum_{i\neq k}\sum_j[-w^T(X_i^+ - X_j^-)+1]^+ + \lambda\|w\|^2 .$$

In the context of typical uniform stability proofs, the set S^k is meant to denote the training set S with the k^{th} point removed. In our context, the removed point will be removed from S^-, hence the sum over indices $j \neq k$, or it will be removed from S^+, hence the sum over indices $i \neq k$. We first prove the case where the point has been removed from S^-. As we will see, the main argument of the proof is symmetric and thus the proof for the case where the point is removed from S^+ follows immediately from the proof of the prior case.

Now, note that for any $w_1, w_2 \in \mathbb{R}^n$, and sample pair $(x^-, x^+) \in \mathscr{X}^- \times \mathscr{X}^+$, we have that

$$\begin{aligned}
\left|V(w_1,x^-,x^+,a) - V(w_2,x^-,x^+,a)\right| &= \left|[-aw_1^T(x^+-x^-)+1]^+ - [-aw_2^T(x^+-x^-)+1]^+\right| \\
&\leq \left|-aw_1^T(x^+-x^-)+1-\left(-aw_2^T(x^+-x^-)+1\right)\right| \\
&= a\left|(w_1-w_2)^T(x^- - x^+)\right| \\
&\leq a\sup_{x^-\in\mathscr{X}^-}\left|(w_1-w_2)^Tx^-\right| + a\sup_{x^+\in\mathscr{X}^+}\left|(w_1-w_2)^Tx^+\right| \\
&\leq a\left(\sup_{\|x\|^*\leq R^-, x\in\mathbb{R}^n}\left|(w_1-w_2)^Tx\right|\right) + a\left(\sup_{\|x\|^*\leq R^+, x\in\mathbb{R}^n}\left|(w_1-w_2)^Tx\right|\right) \\
&= aR^-\|w_1-w_2\| + aR^+\|w_1-w_2\| \\
&= a(R^-+R^+)\|w_1-w_2\| .
\end{aligned}$$

We show now that $\|w_S - w_{S^k}\| \leq \frac{(R^-+R^+)}{rm^-\lambda}$. Let d_f,$\bar{d}_f$ denote the Bregman divergence and generalized Bregman divergence of a convex function f with subderivatives at any w denoted as $\nabla f(w) \in \partial f(w)$. (See Appendix C for definition and properties of Bregman divergence) Let $g(w) = \|w\|^2$, $h_S(w) = \frac{1}{m^+m^-}\sum_i\sum_j[-w^T(X_i^+ - X_j^-)+1]^+$, and $h_{S^k}(w) = \frac{1}{m^+m^-}\sum_i\sum_{j\neq k}[-w^T(X_i^+ - X_j^-)+1]^+$. First, since $g(w)$ is assumed to be r-strongly convex w.r.t norm $\|\cdot\|$, by definition we have $\frac{r}{2}\|w_S - w_{S^k}\|^2 \leq d_g(w_S, w_{S^k})$. Next, we have that

$$\begin{aligned}
\lambda r\|w_S - w_{S^k}\|^2 &\leq \lambda\left(d_g(w_{S^k}, w_S) + d_g(w_S, w_{S^k})\right) \\
&= d_{\lambda g}(w_{S^k}, w_S) + d_{\lambda g}(w_S, w_{S^k}) \\
\text{(following from (31))} &= \bar{d}_{\lambda g}(w_{S^k}, \nabla\lambda g(w_S)) + \bar{d}_{\lambda g}(w_S, \nabla\lambda g(w_{S^k})) .
\end{aligned}$$

Since generalized divergence is linear and non-negative (see Appendix C and (32)), we have for any $\nabla(h_S(w_S)+\lambda g(w_S)) \in \partial(h_S(w_S)+\lambda g(w_S))$ and any $\nabla(h_{S^k}(w_{S^k})+\lambda g(w_{S^k})) \in \partial(h_{S^k}(w_{S^k})+\lambda g(w_{S^k}))$, that

$$\begin{aligned}
\bar{d}_{\lambda g}(w_{S^k}, \nabla\lambda g(w_S)) + \bar{d}_{\lambda g}(w_S, \nabla\lambda g(w_{S^k})) \leq\ & \bar{d}_{(h_S+\lambda g)}(w_{S^k}, \nabla(h_S(w_S)+\lambda g(w_S))) \\
& + \bar{d}_{(h_{S^k}+\lambda g)}(w_S, \nabla(h_{S^k}(w_{S^k})+\lambda g(w_{S^k}))) .
\end{aligned}$$

Now, since $0 \in \partial(h_S(w_S) + \lambda g(w_S))$ and $0 \in \partial(h_{S^k}(w_{S^k}) + \lambda g(w_{S^k}))$, we have

$$\begin{aligned}
\bar{d}_{\lambda g}(w_{S^k}, \nabla\lambda g(w_S)) + \bar{d}_{\lambda g}(w_S, \nabla\lambda g(w_{S^k})) &\leq \bar{d}_{(h_S+\lambda g)}(w_{S^k}, 0) + \bar{d}_{(h_{S^k}+\lambda g)}(w_S, 0) \\
\text{(following from (33))} &= h_S(w_{S^k}) + \lambda g(w_{S^k}) - h_S(w_S) - \lambda g(w_S) \\
&\quad + h_{S^k}(w_S) + \lambda g(w_S) - h_{S^k}(w_{S^k}) - \lambda g(w_{S^k}) \\
&= h_S(w_{S^k}) - h_S(w_S) + h_{S^k}(w_S) - h_{S^k}(w_{S^k}) \\
&= \frac{1}{m^+m^-}\left(\sum_i V(w_{S^k}, X_i^+, X_k^-, 1) - V(w_S, X_i^+, X_k^-, 1)\right) \\
&\leq \frac{m^+(R^- + R^+)\|w_{S^k} - w_S\|}{m^+m^-} .
\end{aligned}$$

The last inequality follows from the prior result bounding the absolute difference in V for any w_1, w_2. This then implies that $\|w_S - w_{S^k}\|^2 \leq \|w_S - w_{S^k}\| \frac{(R^- + R^+)}{rm^-\lambda}$ which further implies that $\|w_S - w_{S^k}\| \leq \frac{(R^- + R^+)}{rm^-\lambda}$. Together with the previous result bounding the absolute difference in V, we have that for any sample pair $(x^-, x^+) \in \mathscr{X}^- \times \mathscr{X}^+$, $|V(w_S, x^-, x^+, a) - V(w_{S^k}, x^-, x^+, a)| \leq \|w_S - w_{S^k}\| a(R^- + R^+) \leq \frac{a(R^- + R^+)^2}{rm^-\lambda}$.

Recall that, up until now, we let S^k signify that we removed the k^{th} point from the set S^-. Notice now that if we had instead started with

$$w_{S^k} \in \operatorname*{argmin}_w \frac{1}{m^+m^-} \sum_{i \neq k} \sum_j [-w^T(X_i^+ - X_j^-) + 1]^+ + \lambda \|w\|^2 ,$$

where S^k signified that we removed the k^{th} point from the set S^+, we could repeat the same arguments to show that for any sample pair $(x^-, x^+) \in \mathscr{X}^- \times \mathscr{X}^+$, $|V(w_S, x^-, x^+, a) - V(w_{S^k}, x^-, x^+, a)| \leq \|w_S - w_{S^k}\| a(R^- + R^+) \leq \frac{a(R^- + R^+)^2}{rm^+\lambda}$. Therefore, combining the two arguments, we know that the learning algorithm is uniformly θ-stable w.r.t. loss function V with $\theta = \frac{a(R^- + R^+)^2}{r\min\{m^+, m^-\}\lambda}$.

To apply Theorem 12 of [2], which gives us the final generalization bounds, we need to show that for any training set S and any sample pair (x^-, x^+), $V(w_S, x^-, x^+, a)$ is bounded. To see this, first notice that

$$\lambda \|w_S\|^2 \leq h_S(w_S) + \lambda \|w_S\|^2 \leq h_S(0) + \lambda \|0\|^2 = \frac{1}{m^+m^-} \sum_i \sum_j V(0, X_j^-, X_i^+, a) = 1 .$$

Second, combining this with the first result of the proof yields,

$$\begin{aligned}
V(w_S, X_j^-, X_i^+, a) &\leq \Big|V(w_S, X_j^-, X_i^+, a) - V(0, X_j^-, X_i^+, a)\Big| + V(0, X_j^-, X_i^+, a) \\
&\leq \|w_S - 0\| a(R^- + R^+) + 1 \\
&\leq \frac{a(R^- + R^+)}{\sqrt{\lambda}} + 1 .
\end{aligned}$$

Thus, applying Theorem 12 of [2], for any $a \geq 0$ yields the first generalization bound. The second generalization bound follows from the fact that if

$$\bar{a} \in \operatorname*{argmin}_{a \geq 0} \frac{1}{m^+m^-} \sum_i \sum_j [-aw^T(X_i^+ - X_j^-) + 1]^+ ,$$

then

$$\widehat{bAUC}(w) = 1 - \frac{1}{m^+m^-} \sum_i \sum_j [-\bar{a}w^T(X_i^+ - X_j^-) + 1]^+ .$$

10 Appendix C: Bregman Divergence Background

For a convex function $f:\mathbb{R}^n \to \mathbb{R}$, let $\partial f(w)$ denote the set of subderivatives $\nabla f(w) \in \partial f(w)$ at a point $w \in \mathbb{R}^n$. For a differentiable, convex f the Bregman Divergence at $w_1, w_2 \in \mathbb{R}^n$ is defined as

$$d_f(w_1,w_2) = f(w_1) - f(w_2) - \langle \nabla f(w_2), w_1 - w_2 \rangle \; . \tag{29}$$

Since f is differentiable, $\nabla f(w)$ is the unique subderivative, and thus the divergence is well defined. If f is convex, but not differentiable, $\nabla f(w)$ may not be unique. In this case, however, following Appendix C of [2], we can define a generalized Bregman Divergence. Letting the function $f^*(a) = \sup_w \langle w,a \rangle - f(w)$ denote the conjugate of f, we define the generalized Bregman Divergence at $w_1, a \in \mathbb{R}^n$ as,

$$\bar{d}_f(w_1,a) = f(w_1) + f^*(a) - \langle w_1, a \rangle \; . \tag{30}$$

As it relates to the normal Bregman Divergence, it is easy to check that for convex differentiable f,

$$d_f(w_1,w_2) = \bar{d}_f(w_1, \nabla f(w_2)) \; . \tag{31}$$

which follows from the property that

$$a \in \partial f(w) \iff f^*(a) = \langle a, w \rangle - f(w) \; .$$

As for its other properties, first note that the generalized divergence is non-negative. Second, notice that we have linearity, such that if $f = g + h$, for convex functions g,h, and we choose subderivatives of f,g,h satisfying $\nabla f(w_2) = \nabla g(w_2) + \nabla h(w_2)$, then

$$\bar{d}_f(w_1, \nabla f(w_2)) = \bar{d}_g(w_1, \nabla g(w_2)) + \bar{d}_h(w_1, \nabla h(w_2) \; . \tag{32}$$

To see this, we simply need to expand the right hand side in the following manner, where we use the fact that $a \in \partial f(w) \iff f^*(a) = \langle a, w \rangle - f(w)$ in the third and fifth equality.

$$\begin{aligned}
\bar{d}_f(w_1, \nabla f(w_2)) &= \bar{d}_{(g+h)}(w_1, \nabla (g+h)(w_2)) \\
&= (g+h)(w_1) + (g+h)^*(\nabla (g+h)(w_2)) - \langle w_1, \nabla (g+h)(w_2) \rangle \\
&= [(g+h)(w_1) - \langle w_1, \nabla (g+h)(w_2) \rangle] + \langle w_2, \nabla (g+h)(w_2) \rangle - (g+h)(w_2) \\
&= [(g+h)(w_1) - \langle w_1, \nabla (g+h)(w_2) \rangle] + (\langle w_2, \nabla g(w_2) \rangle - g(w_2)) + (\langle w_2, \nabla h(w_2) \rangle - h(w_2)) \\
&= [(g+h)(w_1) - \langle w_1, \nabla (g+h)(w_2) \rangle] + g^*(\nabla g(w_2)) + h^*(\nabla h(w_2)) \\
&= (g(w_1) + g^*(\nabla g(w_2)) - \langle w_1, \nabla g(w_2) \rangle) + (h(w_1) + h^*(\nabla h(w_2)) - \langle w_1, \nabla h(w_2) \rangle) \\
&= \bar{d}_g(w_1, \nabla g(w_2)) + \bar{d}_h(w_1, \nabla h(w_2) \; .
\end{aligned}$$

Finally, assume that w_f is a minimizer of f, meaning that $0 \in \partial f(w_f)$. Then, we have for any w_1,

$$\bar{d}_f(w_1, 0) = f(w_1) - f(w_f) \; . \tag{33}$$

References

1. Bache K, Lichman M (2013) UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science
2. Bousquet O, Elisseeff A (2002) Stability and generalization. Journal of Machine Learning Research 2(Mar):499–526
3. Bradley AP (1997) The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition 30(7):1145–1159
4. Brefeld U, Scheffer T (2005) AUC maximizing support vector learning. Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning
5. Caruana R, Baluja S, Mitchell T, et al (1996) Using the future to" sort out" the present: Rankprop and multitask learning for medical risk evaluation. Advances in neural information processing systems pp 959–965
6. Chapelle O, Keerthi SS (2010) Efficient algorithms for ranking with svms. Information Retrieval 13(3):201–215
7. Cortes C, Mohri M (2004) AUC optimization vs. error rate minimization. Advances in Neural Information Processing Systems, 16(16), 313-320
8. Cortes C, Vapnik V (1995) Support-vector networks. Machine learning 20(3):273–297
9. Davis JR, Uryasev S (2015) Analysis of tropical storm damage using buffered probability of exceedance. Natural Hazards pp 1–19
10. Egan JP (1975) Signal detection theory and {ROC} analysis
11. Fawcett T (2006) An introduction to roc analysis. Pattern recognition letters 27(8):861–874
12. Ferri C, Flach P, Hernández-Orallo J, Senad A (2005) Modifying roc curves to incorporate predicted probabilities. In: Proceedings of the second workshop on ROC analysis in machine learning, pp 33–40
13. Frittelli M, Gianin ER (2005) Law invariant convex risk measures. In: Advances in mathematical economics, Springer, pp 33–46
14. Hanley JA, McNeil B (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 143(1), 29-36
15. Herbrich R, Graepel T, Obermayer K (1999) Large margin rank boundaries for ordinal regression. Advances in neural information processing systems pp 115–132
16. Hernández-Orallo J, Flach P, Ferri C (2012) A unified view of performance metrics: Translating threshold choice into expected classification loss. Journal of Machine Learning Research 13(Oct):2813–2869
17. Herschtal A, Raskutti B (2004) Optimising area under the roc curve using gradient descent. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 49
18. Krm E, Yildirak K, Weber G (2012) A classification problem of credit risk rating investigated and solved by optimization of the ROC curve. CEJOR 20, 3 (2012) 529-557; in the special issue at the occasion of EURO XXIV 2010 in Lisbon
19. Ling CX, Huang J, Zhang H (2003) Auc: a statistically consistent and more discriminating measure than accuracy. In: IJCAI, vol 3, pp 519–524
20. Mafusalov A, Shapiro A (2015) Estimation and asymptotics for buffered probability of exceedance. Research Report 2015-5, ISE Dept, University of Florida
21. Mafusalov A, Uryasev S (2015) Buffered probability of exceedance: Mathematical properties and optimization algorithms. Research Report 2014-1, ISE Dept, University of Florida
22. Miura K, Yamashita S, Eguchi S (2010) Area under the curve maximization method in credit scoring. The Journal of Risk Model Validation, 4(2), 3–25
23. Mozer MC (2003) Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic
24. Norton M, Mafusalov A, Uryasev S (2015) Soft margin support vector classification as buffered probability minimization. Research Report 2015-2, ISE Dept, University of Florida
25. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830
26. Platt J, et al (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers 10(3):61–74
27. Provost FJ, Fawcett T, et al (1997) Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: KDD, vol 97, pp 43–48
28. Provost FJ, Fawcett T, Kohavi R (1998) The case against accuracy estimation for comparing induction algorithms. In: ICML, vol 98, pp 445–453

29. Rockafellar R (2009) Safeguarding strategies in risky optimization. Presentation at the International Workshop on Engineering Risk Control and Optimization, Gainesville, FL, February, 2009
30. Rockafellar R, Royset J (2010) On buffered failure probability in design and optimization of structures. Reliability Engineering & System Safety, Vol 95, 499-510
31. Rockafellar R, Uryasev S (2000) Optimization of conditional value-at-risk. The Journal of Risk, Vol 2, No 3, 2000, 21-41
32. Rockafellar RT, Uryasev S (2002) Conditional value-at-risk for general loss distributions. Journal of banking & finance 26(7):1443–1471
33. Schapire WWCRE, Singer Y (1998) Learning to order things. Advances in Neural Information Processing Systems 10:451
34. Swets JA (1988) Measuring the accuracy of diagnostic systems. Science 240(4857):1285–1293
35. Swets JA, Dawes RM, Monahan J (2000) Better decisions through. Scientific American 283:82–87
36. Tayal A, Coleman TF, Li Y (2015) Rankrc: Large-scale nonlinear rare class ranking. IEEE Transactions on Knowledge and Data Engineering 27(12):3347–3359
37. Uryasev S (2014) Buffered probability of exceedance and buffered service level: Definitions and properties. Department of Industrial and Systems Engineering, University of Florida, Research Report 2014-3
38. Vapnik V (1998) Statistical learning theory, vol 1. Wiley New York
39. Vinyals O, Jia Y, Deng L, Darrell T (2012) Learning with recursive perceptual representations. In: Advances in Neural Information Processing Systems, pp 2825–2833
40. Wu S, Flach P (2005) A scored auc metric for classifier evaluation and selection. In: Second Workshop on ROC Analysis in ML, Bonn, Germany
41. Zadrozny B, Elkan C (2001) Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: ICML, Citeseer, vol 1, pp 609–616
42. Zadrozny B, Elkan C (2002) Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 694–699
43. Zou KH (2002) Receiver operating characteristic (roc) literature research. On-line bibliography available from:¡ http://splweb bwh harvard edu 8000