MBA: Mini-Batch AUC Optimization

San Gultekin, Avishek Saha, Adwait Ratnaparkhi, and John Paisley

***Abstract*—Area under the receiver operating characteristics curve (AUC) is an important metric for a wide range of machine learning problems, and scalable methods for optimizing AUC have recently been proposed. However, handling very large datasets remains an open challenge for this problem. This paper proposes a novel approach to AUC maximization, based on sampling mini-batches of positive/negative instance pairs and computing U-statistics to approximate a global risk minimization problem. The resulting algorithm is simple, fast, and learning-rate free. We show that the number of samples required for good performance is independent of the number of pairs available, which is a quadratic function of the positive and negative instances. Extensive experiments show the practical utility of the proposed method.**

***Index Terms*—Area under curve, machine learning, U-statistics, mini batch, convex optimization, matrix concentration inequalities.**

I. Introduction

Bipartite ranking is an important problem in machine learning; given a set of positive and negative inputs, this problem is concerned with building a scoring system, such that the positive samples are ranked higher than the negative ones. Historically the problem has been studied extensively in signal processing literature [1], in particular for radars, as missing the presence of a target (miss detection) could have dire consequences. In such a setting, a received signal is assigned a score, which is then compared against a threshold to decide if the target is present. The receiver operating characteristics (ROC) curve plots the ratio of true positives (detection) to false positives (false alarm) as a function of this threshold, and provides information about the behavior of the system. The area under the ROC curve (AUC) is a threshold-independent metric which measures the fraction of times a positive instance is ranked higher than a negative one. Therefore, it is a natural measure for the bipartite ranking performance. These have a wide variety of applications such as imaging [2], dictionary learning [3], signal detection [4], and noise-enhanced detection [5]–[7], to name a few.

The bipartite ranking problem and ROC curves are not limited to traditional applications. In fact, these tools are ubiquitous in modern machine learning problems. Important subfields include cost-sensitive learning [8], [9] and imbalanced data processing [10]. In the latter, one is given a dataset where the number of negative samples is dominating. This means a classifier which predicts all incoming instances to be negative will have very high prediction accuracy. On the other hand, it will have an AUC of zero. This is worse than random guessing, which would give 0.5, and so the AUC in a sense may be a better choice for performance metric, and devising methods to achieve higher AUC is a meaningful goal. For this reason, the AUC metric is heavily used for website ad click prediction problems [11], where only a very small fraction of web history contains ads clicked by visitors. In this case, a system with high AUC is the one which can distinguish the ads that are "interesting" for a user from the rest, whereas a simple classifier that maximizes prediction accuracy may simply predict "not interesting."

Given that AUC is the primary performance measure for many problems, it is useful to devise algorithms that directly optimize this metric during the training phase. AUC optimization has been studied within the context of well-known machine learning methods, such as support vector machines [12], boosting [13], and decision trees [14]. However, most of these traditional approaches do not scale well as the size of the dataset grows, since AUC is defined over positive/negative *pairs*; this has a quadratic growth of $\mathcal{O}(N_+N_-)$. Moreover, the AUC itself is a sum of indicator functions, and its direct optimization results in NP-hard problems.

Recent research in this direction increasingly focuses on convex surrogate loss functions to represent the AUC. This enables one to use stochastic gradient methods to efficiently learn a ranking function [15]. The first work in this direction is [16], where an online AUC maximization method based on proxy hinge loss is proposed. Later, [17] use the pairwise squared loss function, which eliminates the need for buffering previous instances; [18] propose adaptive gradient/subgradient methods which can also handle sparse inputs, while [19], [20] consider the nonlinear AUC maximization problem using kernel and multiple-kernel methods. Most recently, [21] focuses on scalable kernel methods.

While these approaches can significantly increase scalability, for very large datasets their sequential nature can still be problematic. One widely used technique -particularly for training deep neural networks on large datasets [22]- is processing data in mini-batches. AUC maximization method which can utilize mini-batch processing is thus desirable. In this paper we propose a novel algorithm for fast AUC optimization. Our approach, called Mini-Batch AUC Optimization (MBA) is based on a convex relaxation of the AUC function. However instead of using stochastic gradients, it uses first and second order U-statistics of pairwise differences. A distinctive feature of our approach is it being *learning-rate free*, contrary to mini-batch gradient descent methods. This is important, as tuning the step size *a priori* is a difficult task, and generic approaches such as cross-validation are inefficient, when the dataset is large.

One of the main challenges of AUC optimization is, even if convex relaxation is applied the resulting problem is defined over pairs of positive/negative samples, and the optimization has a sample cost of $\mathcal{O}(N_+N_-)$. This grows prohibitively large even for moderate datasets. Since mini-batch optimization is based on sub-sampling it is important to understand the

behavior of MBA as a function of sample size. Our theoretical analysis reveals that, the solution returned by MBA concentrates around the batch solution that utilizes the *entire pair ensemble* provided by the data, with an exponential tail bound. Unlike previous work, our proofs are based on the more recent results on matrix concentration [23]; and they are quite straightforward. In terms of the related work, while U-processes for ranking problems have previously been explored by [24]. scalable mini-batch algorithms using U-statistics have not been developed. The nearest work of [21] uses mini-batch techniques, but for gradient descent.

We organize the paper as follows. In Section II we start with an overview of the bipartite ranking problem and the statistical learning setup. In Section III we develop the MBA algorithm and carry out the theoretical analysis. Section IV contains extensive experiments including a simulation study, fifteen datasets from UCI/LIBSVM repositories, and three large-scale commercial web click data. We conclude in Section V.

II. Background

A widely studied problem in machine learning is binary classification, in which for each given input $\boldsymbol{x} \in \mathcal{X}$ there is a a corresponding label $y \in \mathcal{Y} = \{+, -\}$. In the learning setup, a set of labeled data is provided for training, and the aim is to make accurate predictions on the unknown inputs. Similar to the PAC learning framework, we assume that the samples provided for training are iid with the following unknown distributions [1]

$$[\boldsymbol{X} \mid y = +] \sim \mathcal{P}^+ \quad , \quad [\boldsymbol{X} \mid y = -] \sim \mathcal{P}^- \tag{1}$$

Since the distributions are unknown, a model $f(\cdot)$ is fitted to the training data using a specific loss function, such as cross-entropy loss for logistic regression and hinge loss for support vector machine. Here, the loss function we focus on is the area under receiver operating characteristics (ROC) curve, which is abbreviated as AUC. Its functional form is given by

$$\text{AUC} = \mathbb{E}_{\substack{\boldsymbol{x}^+ \sim \mathcal{P}^+ \\ \boldsymbol{x}^- \sim \mathcal{P}^-}} \Big[\mathbb{1}\{f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-) > 0\} \Big], \tag{2}$$

from which we note that, it is a measure of how well $f(\cdot)$ ranks a positive sample higher than a negative one, both of which are coming from the generating distribution.

Eq. (2) shows that the AUC is defined as the expectation of an indicator function. This means the corresponding empirical risk function will be a sum of indicators, which would make the opimization NP-hard. For this reason we first discuss a relaxation of the original problem using convex surrogate loss functions.

Replacing $\mathbb{1}[f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-) > 0]$ in Eq. (2) with the pairwise convex surrogate loss $\phi(\boldsymbol{x}^+, \boldsymbol{x}^-) = \phi(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-))$, the aim now becomes to minimize the ϕ-risk [25]

$$R_\phi(f) = \mathbb{E}_{\substack{\boldsymbol{x}^+ \sim \mathcal{P}^+ \\ \boldsymbol{x}^- \sim \mathcal{P}^-}} \left[\phi(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-)) \right]. \tag{3}$$

This is the Bayes risk of the scoring function [26]. There are many possible choices for surrogate function; some common choices are the pairwise squared loss (PSL), pairwise hinge loss (PHL), pairwise exponential loss (PEL), and pairwise logistic loss (PLL) [27]:

$$\begin{aligned} &\phi_{\text{PSL}}(t) = (1-t)^2\,, \;\; \phi_{\text{PHL}}(t) = \max(0,\; 1-t)\,, \\ &\phi_{\text{PEL}}(t) = \exp(-t)\,, \;\; \phi_{\text{PLL}}(t) = \log(1 + \exp(-t))\,, \end{aligned} \tag{4}$$

where $t := f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-)$ is the pairwise scoring difference. Among the recent works on AUC optimization, [16] and [28] use PHL, whereas [17] and [18] focus on PSL. On the other hand, all these studies are focused on deriving a stochastic-gradient based algorithm. In this paper, we use the PSL function for two reasons: (i) its consistency with the original AUC loss is proven [27], and (ii) the structure of PSL allows for a mini-batch algorithm, for which theoretical guarantees can be derived. Unlike stochastic-gradient methods, this formulation is learning-rate free, which quite notably increases its practicality.

We now take one further step and assume that the scoring function is linear in the original input space; however we will discuss nonlinear extensions in Section III. In this case we have the further simplification $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$ and the ϕ-risk becomes

$$\begin{aligned} R_\phi(f) &= \mathbb{E}_{\substack{\boldsymbol{x}^+ \sim \mathcal{P}^+ \\ \boldsymbol{x}^- \sim \mathcal{P}^-}} \Big[\big(\; 1 - \boldsymbol{w}^\top (\boldsymbol{x}^+ - \boldsymbol{x}^-) \;\big)^2 \Big] \\ &= 1 - 2\boldsymbol{w}^\top \mathbb{E}\left[(\boldsymbol{x}^+ - \boldsymbol{x}^-)\right] \\ &\qquad + \boldsymbol{w}^\top \mathbb{E}\left[(\boldsymbol{x}^+ - \boldsymbol{x}^-)(\boldsymbol{x}^+ - \boldsymbol{x}^-)^\top\right] \boldsymbol{w} \\ &= 1 - 2\boldsymbol{w}^\top \boldsymbol{\mu} + \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}, \end{aligned} \tag{5}$$

where we define $\boldsymbol{x}_{\text{diff}} := (\boldsymbol{x}^+ - \boldsymbol{x}^-)$, and $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}_{\text{diff}}]$ and $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{x}_{\text{diff}}\, \boldsymbol{x}_{\text{diff}}^\top]$. Note that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ characterize the first and second order statistics of $\boldsymbol{x}_{\text{diff}}$, the pairwise difference. This is important because the quality of bipartite ranking does not directly depend on the positive and negative inputs, but the differences between them. This observation will form the basis of our mini-batch algorithm in the next section. Finally, by definition $\boldsymbol{\Sigma}$ is positive semi-definite when it is positive definite, there is a unique $\boldsymbol{w}^\star$ that achieves the minimum $R_\phi(f)$.

At this point, optimizing either one of the objectives in Eq. (2) or Eq. (5) would require knowledge of $\mathcal{P}^+$ and $\mathcal{P}^-$, which is unavailable. Instead we are given iid samples from these distributions, as we mentioned in the beginning. The task is to learn a score function which should yield high AUC on the test data, also known as the generalization performance. Here the corresponding empirical AUC metric is

$$\text{AUC} = \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} \mathbb{1}\{f(\boldsymbol{x}_i^+) - f(\boldsymbol{x}_j^-) > 0\} \tag{6}$$

which replaces the expectation in Eq. (2) with sample-base average. As mentioned before, direct optimization of the empirical AUC is NP-hard as it is a sum of indicators, furthermore the sum itself contains pairs that grow quadratically with the training data. To sidestep this difficulty, a surrogate loss function $\phi(\cdot)$ can be chosen to replace the Eq. (6), as we did for (2). In particular, the empirical ϕ-risk (c.f. Eq. (3)) has the following general form

$$\widehat{R}_\phi(f) = \frac{1}{2N_+N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} \phi(f(\boldsymbol{x}_i^+) - f(\boldsymbol{x}_j^-)) \;. \tag{7}$$

[1] Here we consider the case with no label noise, however the modification to that case is straightforward.

Based on this final equation we will derive the MBA algorithm in the next section.

III. Mini-Batch AUC Optimization

Using the pairwise squared loss function we obtained a convex optimization problem in place of the original NP-hard problem; however, Eq. (3) still cannot be used as it relies on the data generating distribution $\mathcal{P}$. In practical settings, we are only given a set of positive and negative instances sampled from $\mathcal{P}$, written $\mathcal{S}^+ = \{\boldsymbol{x}_1^+, \ldots, \boldsymbol{x}_{N_+}^+\}$, $\mathcal{S}^- = \{\boldsymbol{x}_1^-, \ldots, \boldsymbol{x}_{N_-}^-\}$. We therefore substituted the empirical risk in Eq. (7) which can be more easily optimized. The term $N := N_+ N_-$ corresponds to the total number of pairs in the data. Similar to the ϕ-risk, optimizing the empirical risk yields a convex problem, but the number of pairs grows quadratic in the number of data points. Therefore, even for moderate datasets, minimizing the empirical risk in Eq. (7) becomes intractable.

We now substitute the PSL and functional form of linear classifier into the empirical risk to obtain

$$\widehat{R}_\phi(\boldsymbol{w}) = -\boldsymbol{w}^\top \left[\frac{1}{N_+ N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \right] + \frac{1}{2} \boldsymbol{w}^\top \left[\frac{1}{N_+ N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top \right] \boldsymbol{w} \tag{8}$$

and define

$$\boldsymbol{\mu}_N = \frac{1}{N_+ N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$$

$$\boldsymbol{\Sigma}_N = \frac{1}{N_+ N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\ (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top. \tag{9}$$

The variables in Eq. (9) are sample-based approximations to the first and second moments of $\boldsymbol{x}_{\text{diff}}$, which are denoted by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in Eq. (3). Overall, the optimization problem to be solved is

$$\boldsymbol{w}_N^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{\Sigma}_N \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}_N + R_{\text{EL}}(\boldsymbol{w}), \tag{10}$$

where $R_{\text{EL}}(\boldsymbol{w}) := \lambda_1 \|\boldsymbol{w}\|_1 + (1/2)\lambda_2 \|\boldsymbol{w}\|_2^2$ is the elastic net regularizer [29], which we add to prevent overfitting. Note that, unlike Eq. (3), there is always a unique optimum since the elastic net penalty makes the objective strictly convex. In addition, this regularizer encourages solution which combines small ℓ_2 norm with sparsity. By substituting appropriate values for λ_1 and λ_2 we also recover ridge and lasso regression. In this paper, we report results for all three cases.

Since it is impractical to use all N samples, we propose to use mini-batches to obtain estimates of the moments. This is a simple process which only requires the computation of U-statistics. Note that, given a parameter θ and symmetric measurable function h which satisfies $\theta = h(X_1, \ldots, X_m)$, the corresponding U-statistic is given by

$$U_n = \binom{n}{m}^{-1} \sum_{C_{n,m}} h(X_1, \ldots, X_n), \tag{11}$$

where $C_{n,m}$ is the set of all length-m combinations with increasing indices. As the name implies the U-statistics are unbiased, so $\theta = \mathbb{E}[U_n]$, and provide best unbiased estimators [30]. On the other hand, a U-statistic of the second moment matrix $\boldsymbol{\Sigma}_N$ [2] also provides a building block to get an exponential concentration bound [23]. Our theoretical analysis will use this property.

A. The MBA algorithm

We now describe the proposed MBA algorithm. Let T be the total number of rounds. At round t we sample B positive and B negative samples from the entire population with replacement. Let $\mathcal{S}_t^+$ and $\mathcal{S}_t^-$ be the arrays of sample indices and let $\mathcal{S}_t$ be the array of pairs stored as tuples of the form $(\mathcal{S}_t^+(i), \mathcal{S}_t^-(i))$—note that we do not form the Cartesian product. The expressions for U-statistics of the first and second moments simplify from Eq. (11) as

$$\boldsymbol{\mu}_t := \frac{1}{B} \sum_{(i,j) \in \mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$$

$$\boldsymbol{\Sigma}_t := \frac{1}{B} \sum_{(i,j) \in \mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\ (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top. \tag{12}$$

Finally let $S = BT$ denote the total number of pairs sampled by our algorithm. We also introduce the notation $\mathcal{S}_{1:T}$ for the entire array of pairs sampled during all rounds. The overall moment approximations are therefore

$$\boldsymbol{\mu}_S := \frac{1}{BT} \sum_{(i,j) \in \mathcal{S}_{1:T}} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$$

$$\boldsymbol{\Sigma}_S := \frac{1}{BT} \sum_{(i,j) \in \mathcal{S}_{1:T}} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\ (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top, \tag{13}$$

and the optimization problem constructed by MBA is

$$\boldsymbol{w}_S^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{\Sigma}_S \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}_S + R_{\text{EL}}(\boldsymbol{w}). \tag{14}$$

This is the function MBA aims to construct and solve, which itself is an approximation to the global risk minimization problem in Eq. (3). On the other hand, stochastic gradient-based approaches make local gradient approximations to the global function and seek a solution that way. As we will show in the experiments, this is an important difference and MBA can find better solutions since it constructs a global problem first. We summarize the proposed MBA in Algorithm 1.

Mini-batch optimizations are heavily employed in machine learning, including training of deep neural networks [22] and scalable Bayesian inference [31]. The main benefit of using mini-batches is, it is significantly faster compared to the sequential approach. Online methods for optimizing AUC, however, require a sequential processing, as the parameters are updated per input. This is the main reason MBA offers a significant improvement in speed. In addition to this, MBA offers several other advantages. Since sampling pairs and computing U-statistics is an isolated process, MBA can easily be distributed across machines, which can work in an

[2] Note that for a *given* training set with N pairs, the quantity $\boldsymbol{\Sigma}_N$ is indeed an *unknown* constant.

Algorithm 1 Mini-Batch AUC Optimization (MBA)

Require: B, T, λ_1, λ_2
Input: $\boldsymbol{X}^+$, $\boldsymbol{X}^-$
Output: $\boldsymbol{w}^\star$
Initialize $\boldsymbol{\mu}_S = \mathbf{0}$ and $\boldsymbol{\Sigma}_S = \mathbf{0}$.
for $t = 1, \ldots, T$ **do**
  Construct index set $\mathcal{S}_t^+$ of size B sampling positive examples uniformly with replacement.
  Construct index set $\mathcal{S}_t^-$ of size B sampling negative examples uniformly with replacement.
  Construct $\mathcal{S}_t(i) = (\mathcal{S}_t^+(i), \mathcal{S}_t^-(i)),\ i = 1, \ldots, B$.
  $\boldsymbol{\mu}_S \leftarrow \boldsymbol{\mu}_S + \frac{1}{BT} \sum_{(i,j)\in\mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$
  $\boldsymbol{\Sigma}_S \leftarrow \boldsymbol{\Sigma}_S + \frac{1}{BT} \sum_{(i,j)\in\mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\ (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top$
end for
$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{\Sigma}_S \boldsymbol{w}^\top - \boldsymbol{w}^\top \boldsymbol{\mu}_S + \lambda_1 \|\boldsymbol{w}\|_1 + \frac{\lambda_2}{2} \|\boldsymbol{w}\|_2^2$
Note: Here $\boldsymbol{X}^+$ and $\boldsymbol{X}^-$ correspond to the $d \times N_+$ and $d \times N_-$ dimensional matrices of positive and negative instances. The indices in sets $\mathcal{S}$, $\mathcal{S}^+$ and $\mathcal{S}^-$ correspond to column numbers. Consequently, each vector $\boldsymbol{x}_i^+$ or $\boldsymbol{x}_j^-$ are $d \times 1$, $\boldsymbol{\mu}_S$ is $d \times 1$, $\boldsymbol{\Sigma}_S$ is $d \times d$.

asynchronous manner. Therefore MBA is suitable for cluster computing. Secondly, streaming and/or nonstationary data processing can be incorporated into the MBA framework, as it can process streams as blocks and give larger weights to more recent ones.

Remark: From an algorithmic perspective, as far as the sample size is concerned the only important parameter is S; we introduced B and T to further manage computational resources. For instance when S samples are too large to fit into memory we can instead use B samples in T rounds. Alternatively, S samples can be obtained by T machines with B samples per machine in a single round. In any case, we will have S uniform pairs sampled from the entire set.

B. Theoretical Analysis

Solving the regularized empirical risk minimization problem in Eq. (10) requires processing N pairwise samples. As this number grows quadratically with the number of positive and negative samples, it is often not possible to do this exactly. The proposed MBA addresses this problem by approximating the N-pair problem with an S-pair one, where S samples are collected in mini-batches and the total number of processed samples is much less than N. This results in the problem in Eq. (14). Clearly the success of this approach depends on how well the second problem approximates the first. In this section we derive performance guarantees.

Formally, given N samples we can solve the following regularized empirical risk minimization problem

$$\boldsymbol{w}_N^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{\Sigma}_N \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}_N + R_{\text{EL}}(\boldsymbol{w})$$

however as N can be very large we instead solve

$$\boldsymbol{w}_S^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{\Sigma}_S \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}_S + R_{\text{EL}}(\boldsymbol{w})$$

where S uniform samples are used. We would like to show that the two solutions are close to each other, while $S \ll N$. We next derive a bound for the closeness of solutions. Using the difference between final costs $|\mathcal{L}_N(\boldsymbol{w}_N) - \mathcal{L}_N(\boldsymbol{w}_S)|$ results in a regret bound similar to the ones used in comparing online/batch versions of algorithms [32], whereas the Euclidean distance $d(\boldsymbol{w}_N - \boldsymbol{w}_S) = \|\boldsymbol{w}_N - \boldsymbol{w}_S\|_2^2$ measures how similar the two solutions are, and is used by recent work on matrix sketching [33]. Here we show results for the Euclidean distance in Theorem 2, but both metrics are addressed in the process.

We define the following two functions for convenience,

$$\begin{aligned}\mathcal{L}_N &= \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{\Sigma}_N \boldsymbol{w}^\top - \boldsymbol{w}^\top \boldsymbol{\mu}_N + \lambda_1 \|\boldsymbol{w}\|_1 + \frac{\lambda_2}{2} \|\boldsymbol{w}\|_2^2 \\ \mathcal{L}_S &= \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{\Sigma}_S \boldsymbol{w}^\top - \boldsymbol{w}^\top \boldsymbol{\mu}_S + \lambda_1 \|\boldsymbol{w}\|_1 + \frac{\lambda_2}{2} \|\boldsymbol{w}\|_2^2, \quad (15)\end{aligned}$$

and let $\boldsymbol{w}_N^\star$ and $\boldsymbol{w}_S^\star$ denote their unique minimizers. Since $\mathcal{L}_N(\boldsymbol{w})$ is strictly convex, for a fixed δ there exists an ϵ such that

$$|\mathcal{L}_N(\boldsymbol{w}_N^\star) - \mathcal{L}_N(\boldsymbol{w}_S^\star)| \le \epsilon \implies \|\boldsymbol{w}_N^\star - \boldsymbol{w}_S^\star\|_2 \le \delta. \quad (16)$$

Clearly, ϵ is the infimum of $\mathcal{L}_N(\cdot)$ over the circle centered at $\boldsymbol{w}_N^\star$ with radius δ. Consequently, one can focus on bounding the objective function, and this will yield the desired bound on the solutions. We now introduce ℓ_2-norm bounds on data and weight vectors, and for any given input we assume that $\|\boldsymbol{x}\|_2^2 \le R_x$.[3] Next, define the upper bound on weights such that $\max\{\|\boldsymbol{w}_N^\star\|_2^2, \|\boldsymbol{w}_S^\star\|_2^2\} \le R_w$. Note here that $R_w < \infty$ is guaranteed by the ℓ_2 regularization of elastic net. Also define the following quantities for convenience,

$$\begin{aligned}\Delta(\boldsymbol{w}) &= \mathcal{L}_S(\boldsymbol{w}) - \mathcal{L}_N(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^\top \Delta_{\boldsymbol{\Sigma}} \boldsymbol{w} + \boldsymbol{w}^\top(\boldsymbol{\mu}_N - \boldsymbol{\mu}_S) \\ \Delta_{\boldsymbol{\Sigma}} &= \boldsymbol{\Sigma}_S - \boldsymbol{\Sigma}_N \\ \Delta_\sigma &= (\boldsymbol{w}_N^\star - \boldsymbol{w}_S^\star)^\top(\boldsymbol{\mu}_N - \boldsymbol{\mu}_S). \quad (17)\end{aligned}$$

Here it can be seen that the objective function can be bounded through $\Delta(\boldsymbol{w})$, which in turn can be bounded through $\Delta_{\boldsymbol{\Sigma}}$ and Δ_σ. The following lemma provides two useful concentration inequalities for this purpose.

Lemma 1: Let $\|\Delta_{\boldsymbol{\Sigma}}\|_2$ and $|\Delta_\sigma|$ denote the spectral and ℓ_1 norms respectively. For $\gamma > 0$ and sample size S,

(i) $P(\|\Delta_{\boldsymbol{\Sigma}}\|_2 > \gamma) \le$
$$2d \exp\left\{-S \frac{\gamma^2}{4R_x \|\boldsymbol{\Sigma}_N\|_2 + (8/3)\gamma R_x}\right\}$$
(ii) $P(|\Delta_\sigma| > \gamma) \le$
$$2 \exp\left\{-S \frac{\gamma^2}{4R_w \|\boldsymbol{\Sigma}_N\|_2 + (8/3)\gamma \sqrt{R_x R_w}}\right\}$$

Both parts of this lemma utilize the unbiased statistics of $\boldsymbol{\mu}_N$ and $\boldsymbol{\Sigma}_N$ and use the scalar and matrix Bernstein inequalities. This is made possible by the unbiased sampling procedure. We provide a full proof in Appendix B. Using Lemma 1 we can now state our main result.

[3] This is a mild assumption since training data is typically normalized.

Theorem 2: Let $\boldsymbol{w}_S^\star$ be the solution returned by MBA using S samples. For $\epsilon > 0$, if

$$S \geq \max\Bigg\{\log(4d/p)\frac{[48R_w^2\left\|\boldsymbol{\Sigma}_N\right\|_2 + 16\epsilon R_w]R_x}{3\epsilon^2}\ , \\ \log(4/p)\frac{48R_w\left\|\boldsymbol{\Sigma}_N\right\|_2 + 16\epsilon\sqrt{R_xR_w}}{3\epsilon^2}\Bigg\} \quad (18)$$

then $\|\boldsymbol{w}_N^\star - \boldsymbol{w}_S^\star\|_2 \leq \delta$ with probability at least $1-p$.

Proof: The starting point of the proof is the equality

$$\mathcal{L}_S(\boldsymbol{w}_N^\star) - \mathcal{L}_S(\boldsymbol{w}_S^\star) = \\ \mathcal{L}_N(\boldsymbol{w}_N^\star) + \Delta(\boldsymbol{w}_N^\star) - \mathcal{L}_N(\boldsymbol{w}_S^\star) - \Delta(\boldsymbol{w}_S^\star)\ . \quad (19)$$

Here by construction

$$\mathcal{L}_N(\boldsymbol{w}_S^\star) - \mathcal{L}_N(\boldsymbol{w}_N^\star) = \mathcal{L}_N(\boldsymbol{w}_S^\star) - \arg\min_{\boldsymbol{w}} \mathcal{L}_N(\boldsymbol{w}) \geq 0 \quad (20)$$

$$\mathcal{L}_S(\boldsymbol{w}_N^\star) - \mathcal{L}_S(\boldsymbol{w}_S^\star) = \mathcal{L}_S(\boldsymbol{w}_N^\star) - \arg\min_{\boldsymbol{w}} \mathcal{L}_S(\boldsymbol{w}) \geq 0 \quad (21)$$

and we obtain

$$0 \leq \mathcal{L}_N(\boldsymbol{w}_S^\star) - \mathcal{L}_N(\boldsymbol{w}_N^\star) \leq \Delta(\boldsymbol{w}_N^\star) - \Delta(\boldsymbol{w}_S^\star)\ . \quad (22)$$

The left hand size of the inequality is a direct consequence of Eq. (20). For the right-hand side note that Eq. (19) can be manipulated as

$$\begin{aligned}\mathcal{L}_N(\boldsymbol{w}_S^\star) - \mathcal{L}_N(\boldsymbol{w}_N^\star) &= \Delta(\boldsymbol{w}_N^\star) - \Delta(\boldsymbol{w}_S^\star) \\ &\quad + [\mathcal{L}_S(\boldsymbol{w}_S^\star) - \mathcal{L}_S(\boldsymbol{w}_N^\star)] \\ &\leq \Delta(\boldsymbol{w}_N^\star) - \Delta(\boldsymbol{w}_S^\star) \end{aligned} \quad (23)$$

where the second line follows from $[\mathcal{L}_S(\boldsymbol{w}_S^\star) - \mathcal{L}_S(\boldsymbol{w}_N^\star)] < 0$ due to Eq. (21).

It is therefore sufficient to show that $\Delta(\boldsymbol{w}_N^\star) - \Delta(\boldsymbol{w}_S^\star) \leq \epsilon$ with high probability; the result then follows from the strict convexity argument. Further expand this bounding term as

$$\Delta(\boldsymbol{w}_N^\star) - \Delta(\boldsymbol{w}_S^\star) = \\ \frac{1}{2}\boldsymbol{w}_N^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_N^\star - \frac{1}{2}\boldsymbol{w}_S^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_S^\star + \boldsymbol{w}_N^{\star\top}\Delta_{\boldsymbol{\mu}} - \boldsymbol{w}_S^{\star\top}\Delta_{\boldsymbol{\mu}}\ , \quad (24)$$

and consider uniformly bounding the following two terms:

- Quadratic: $\left|\frac{1}{2}\boldsymbol{w}_N^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_N^\star - \frac{1}{2}\boldsymbol{w}_S^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_S^\star\right| < \frac{\epsilon}{2}$
- Linear: $\left|\boldsymbol{w}_S^{\star\top}\Delta_{\boldsymbol{\mu}} - \boldsymbol{w}_N^{\star\top}\Delta_{\boldsymbol{\mu}}\right| < \frac{\epsilon}{2}$

For the quadratic term we have

$$\begin{aligned}&\left|\frac{1}{2}\boldsymbol{w}_N^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_N^\star - \frac{1}{2}\boldsymbol{w}_S^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_S^\star\right| \\ &\leq \frac{1}{2}\left|\boldsymbol{w}_N^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_N^\star\right| + \frac{1}{2}\left|\boldsymbol{w}_S^{\star\top}\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w}_S^\star\right| \\ &\leq \frac{1}{2}R_w\left\|\Delta_{\boldsymbol{\Sigma}}\right\|_2 + \frac{1}{2}R_w\left\|\Delta_{\boldsymbol{\Sigma}}\right\|_2 \\ &= R_w\left\|\Delta_{\boldsymbol{\Sigma}}\right\|_2\ . \end{aligned} \quad (25)$$

Here the first line is obtained via triangle inequality, for the second line, note that for any given quadratic form the inequality $\boldsymbol{w}^\top\Delta_{\boldsymbol{\Sigma}}\boldsymbol{w} \leq \|\boldsymbol{w}\|_2^2\left\|\Delta_{\boldsymbol{\Sigma}}\right\|_2$ holds, as for any unit norm input $\boldsymbol{u}$, $\boldsymbol{u}^\top\Delta_{\boldsymbol{\Sigma}}\boldsymbol{u}$ is maximized at the largest eigenvalue of $\Delta_{\boldsymbol{\Sigma}}$. Equivalently, we want $\left\|\Delta_{\boldsymbol{\Sigma}}\right\|_2 \leq \epsilon/(2R_w)$ with high probability. Now applying Lemma 2-i with threshold $\gamma = \epsilon/(2R_w)$ and probability level $p/2$, we obtain the first term in Eq. (18).

For the linear term, note that this is already equal to Δ_σ by definition, i.e. we want to achieve $|\Delta_\sigma| \leq \epsilon/2$ with probability at least $1-p/2$. Applying Lemma 2-ii with threshold $\gamma = \epsilon/2$ and probability level $p/2$, we obtain the second term in Eq. (18). Since both terms are bounded with probability at least $1-p/2$, the theorem now follows from the union bound. ■

Theorem 4 shows that the number of samples S required to guarantee $\|\boldsymbol{w}_N^\star - \boldsymbol{w}_S^\star\|_2 < \delta$ with high probability does not depend on the total number of pairs $N = N_+N_-$ provided. Instead the sample size grows logarithmically with the feature size. This result is useful in that, even though the total number of pairs in the data is too large, randomly sampling a small fraction guarantees a solution that is close to the true solution.

Remark: It might at first seem unreasonable that S does not depend on N; but only on the input dimension d. The reason for this favorable result is that, the given samples are *fixed*, and we are uniformly subsampling from them. So in this context N is actually the support size of a discrete uniform distribution of samples. Since the concentration inequalities do not depend on the support size, we consequently do not have N in Eq. (18).

While linear models are oftentimes quite competitive, in practice we can have datasets that are not linearly separable; in such cases one is typically concerned with devising a nonlinear feature transform, to obtain separability. In fact, for finite-dimensional transforms Theorem 2 readily extends. Such transforms include, for example, polynomial features and conjunctions. random Fourier features [34], and random shallow neural networks [35]. In more abstract terms, all these transformations are mappings from d dimensions to F dimensions. Given such fixed transformation, the result of Theorem 2 still holds, where we replace d by F. Therefore, when the input space is not good for a linear ranking function, we can first apply the feature engineering, and then the MBA can be used without any modification.

IV. Experiments

In this section we conduct four types of experiments to demonstrate the performance benefits of MBA. In the first part we use simulation data from Gaussian mixtures to investigate performance of various methods; furthermore since we know the distribution generating the data here, we can also use the Neyman-Pearson decision rule, which is theoretically optimum (we discuss this in detail in Appendix A). Here we show that MBA can achieve better performance with low number of samples, corroborating the theoretical analysis. For the second part, we experiment on 15 frequently used benchmark datasets from the UCI[4] and LIBSVM[5] repositories; these datasets cover a wide range of application domains and show MBA performs better than the competing methods overall. We then use the same datasets in the third part to demonstrate how the linear framework can be extended to account for nonlinear features. Finally we consider large scale click through rate (CTR)

[4] archive.ics.uci.edu/ml/

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

prediction problem with two publicly available commercial-size datasets with tens of millions of samples.

For comparison we use the following algorithms: MBA-ℓ_2, MBA-ℓ_1, MBA-EL, which represent the three variants of our mini-batch AUC optimization using ridge regression, lasso, and elastic net respectively. OLR is simply the online logistic regression and SOLR is the sparse regression algorithm presented in [11]. OAM is the first proposed online AUC maximization algorithm using stochastic gradients [16] which uses PHL. On the other hand, AdaAUC is the adaptive gradient AUC maximization algorithm in [18]. This algorithm is proposed as an improvement to one pass AUC optimization algorithm in [17]. It is shown that AdaAUC has better performance empirically. Therefore we use this version in our comparisons. Both algorithms are based on PSL. In addition to these, we implement two mini-batch stochastic gradient algorithms for large scale CTR prediction problems: MB-PHL is a mini-batch gradient descent algorithm which uses PHL. A variant of this approach is also proposed in the recent work of [21]. MB-PSL is another mini-batch gradient method that uses PSL. MB-PHL and MB-PSL can be thought of a substitutions for OAM and AdaAUC for larger datasets. This is necessary as for large number of inputs sequential processing becomes inefficient. All implementations are done in Python with NumPy and Scikit-Learn libraries. (We will make the code available.)

TABLE I
SUMMARY STATISTICS OF DATASETS USED IN EXPERIMENTS. FOR EACH DATASET WE SHOW THE TRAIN/TEST SAMPLE SIZE, FEATURE SIZE, AND THE RATIO OF NEGATIVE SAMPLES TO POSITIVE SAMPLES IN THE TRAINING SET.

Dataset	# Samp.	# Feat.	T_- / T_+
a1a	1.6K / 30.9K	123	3.06
a9a	32.5K / 16.2K	123	3.15
amazon	750 / 750	10,000	2.33
bank	20.6K / 20.6K	100	7.88
codrna	29.8K / 29.8K	8	2.00
german	500 / 500	24	2.33
ijcnn	50K / 92K	22	9.30
madelon	2,000 / 600	500	1.00
mnist	60K / 10K	780	2.30
mushrooms	4K / 4K	112	0.93
phishing	5.5K / 5.5K	68	0.79
svmguide3	642 / 642	21	2.80
usps	7.2K / 2K	256	2.61
w1a	2.5K / 47.2K	300	33.40
w7a	25K / 25K	300	32.40
avazu app	12.6M / 2M	10,000	8.33
avazu site	23.6M / 2.6M	10,000	4.06
criteo	45.8M / 6M	10,000	2.92

A. Simulation Study

For the simulations we consider the binary hypothesis testing problem of Eq. (28), which can represent, for example, a radar or communication channel setting. In particular, we employ Gaussian mixtures as data generating distributions. Namely, for hypothesis-i a K-component Gaussian mixture is given by

$$p_i(\boldsymbol{x}) = \sum_{k=1}^{K} c_{ik}\ (2\pi)^{-d/2}\ |\boldsymbol{\Sigma}_{ik}|^{-1/2} \times \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_{ik})^\top \boldsymbol{\Sigma}_{ik}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{ik})\right\}\ , \qquad (26)$$

where the weights c_{ik} are convex combination coefficients to ensure the function is a valid pdf. This distribution is completely characterized by the weights, means, and covariances. For hypothesis-i let c_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote the *set* of these parameters. For our experiments $k \in \{1, 2, 3\}$ and:

- $k = 1$: We set $c_0 = \{1\}$, $\boldsymbol{\mu}_0 = \{-\mathbf{0.1}\}$, $\boldsymbol{\Sigma}_0 = \{\boldsymbol{I}\}$ and $c_1 = \{1\}$, $\boldsymbol{\mu}_1 = \{\mathbf{0.1}\}$, $\boldsymbol{\Sigma}_1 = \{\boldsymbol{I}\}$.
- $k = 2$: We set $c_0 = \{0.9, 0.1\}$, $\boldsymbol{\mu}_0 = \{-\mathbf{0.1}, \mathbf{0.1}\}$, $\boldsymbol{\Sigma}_0 = \{\boldsymbol{I}, \boldsymbol{I}\}$ and $c_1 = \{0.1, 0.9\}$, $\boldsymbol{\mu}_1 = \{-\mathbf{0.1}, \mathbf{0.1}\}$, $\boldsymbol{\Sigma}_1 = \{\boldsymbol{I}, \boldsymbol{I}\}$.
- $k = 3$: We set $c_0 = \{0.8, 0.1, 0.1\}$, $\boldsymbol{\mu}_0 = \{-\mathbf{0.1}, \mathbf{0}, \mathbf{0.1}\}$, $\boldsymbol{\Sigma}_0 = \{\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I}\}$ and $c_1 = \{0.1, 0.1, 0.8\}$, $\boldsymbol{\mu}_1 = \{-\mathbf{0.1}, \mathbf{0}, \mathbf{0.1}\}$, $\boldsymbol{\Sigma}_1 = \{\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I}\}$

As it can be seen the distributions we choose for the two hypotheses are symmetric across the origin. As the number of components increase the distributions get less interspersed, making separation more challenging. All covariances are set to be unit-variance and isotropic; we finally note that the bold numbers for the mean sets correspond to a vector of the bold entry replicated. For our experiments, for each value of k we form 50 training sets, where for each dataset we sample 20,000 points from the generative distribution. Furthermore we create an imbalanced dataset, where roughly 90% of the data has label 0 (i.e. sampled from hypothesis 0). We also create a separate test set, with 100,000 samples and the same imbalance ratio.

Since the generative distributions are assumed known in this setting, we can analytically derive the Neyman Pearson (NP) decision rule which maximizes the AUC. For any given k, the NP rule computes the scores through $p_1(\boldsymbol{x})/p_0(\boldsymbol{x})$. Note that, this does not require any training data, as the generating distributions are already known. A particularly interesting case is $k = 1$. Here the log likelihood is [6]

$$\log \frac{p_1(\boldsymbol{x})}{p_0(\boldsymbol{x})} = \boldsymbol{x}^\top(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \frac{1}{2}(\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1^\top \boldsymbol{\mu}_1) \qquad (27)$$

Since the constant term does not affect AUC, we see that the optimum ranking rule is a linear function of $\boldsymbol{x}$. So for this specific case, the class of linear discriminants (i.e. $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$) is consistent [25].

For the comparisons in this section we use MBA-ℓ_2, ONLR, and AdaAUC, as the latter two typically have the best competitive performance against the former. We run the experiments for three different sample ratio (SR), namely SR $\in \{1\%, 10\%, 100\%\}$; this represents the percentage of available data points used for training. As mentioned above we average the generalization performance over 50 training samples, and report the average values and standard deviations. Table II shows the resulting AUC values and Figure 1 displays the corresponding ROC curves.

[6] In fact the linearity holds for arbitrary Σ as long as it is shared by both hypotheses.

TABLE II
COMPARISONS OF ALGORITHMS ON SIMULATED DATA THE PERFORMANCE OF MBA-ℓ_2, ONLR, AND ADAAUC ARE REPORTED FOR $k \in \{1, 2, 3\}$ AND SR $\in \{1\%, 10\%, 100\%\}$. THE SYMBOLS FILLED/EMPTY CIRCLE INDICATE THAT MBA IS (STATISTICALLY) SIGNIFICANTLY BETTER/WORSE.

Distribution	SR	Neyman-Pearson	MBA-ℓ_2	ONLR	AdaAUC
1-Component	1 %		87.43 ± 0.69	• 86.79 ± 0.99	• 80.94 ± 2.77
Gaussian	10 %	92.13	91.44 ± 0.10	• 90.54 ± 0.29	• 90.25 ± 0.31
Mixture	100 %		91.88 ± 0.04	• 90.69 ± 0.24	• 91.70 ± 0.07
2-Component	1 %		80.15 ± 0.68	• 77.64 ± 1.41	• 69.75 ± 3.24
Gaussian	10 %	83.71	83.15 ± 0.10	• 80.19 ± 0.69	• 80.12 ± 0.69
Mixture	100 %		83.47 ± 0.04	• 80.19 ± 0.69	• 83.01 ± 0.11
3-Component	1 %		76.39 ± 0.47	• 73.07 ± 1.06	• 66.38 ± 1.95
Gaussian	10 %	80.22	79.52 ± 0.11	• 75.94 ± 0.64	• 76.72 ± 0.33
Mixture	100 %		79.93 ± 0.11	• 75.91 ± 0.83	• 79.61 ± 0.06

Firstly note that, as k increases, the AUC value achieved by the optimal NP rule decreases; this shows that adding more mixture components progressively make the problem harder. As we increase the SR, all three learning algorithms improve, as expected. However, we can see that the starting point for MBA is significantly higher than the the other two. In particular, AdaAUC is worse for small sample sizes. This shows the difficulty of optimizing a bivariate loss function as opposed to the univariate logistic loss of ONLR. The particular difficulty comes from selecting the step size, and for smaller number of samples the stochastic gradient cannot efficiently optimize. Being learning-rate and gradient free, MBA does not suffer from these drawbacks and always performs better than ONLR.

We also use pairwise t-test to assess the statistical significance of results, using 95% confidence level, as proposed and used by [17] initially for this problem. For all cases considered we see that MBA achieves better results than its competitors with significance. This is true even when $SR = 100\%$ and the average values are close, as the standard deviations are low and a high number of experiments are performed.

Interestingly, comparing the performance of NP and MBA we see that in all three cases the achieved AUC is quite close. This is the case even when $k > 1$; therefore even if the optimal scores are a nonlinear function of $\boldsymbol{x}$ for these cases, the linear approximation is still reasonable. With that said, if the data is highly nonlinear it might be difficult to find a good separating hyperplane.

B. UCI and LIBSVM Benchmark Data

In this section we experiment with 15 frequently used benchmark datasets from the UCI and LIBSVM repositories which we summarize in Table I. It can be seen that the chosen datasets cover a wide range of sample/feature sizes. The distribution of samples vary from being linearly separable to highly nonlinear. The datasets also exhibit significant differences in label imbalance. In terms of the features present, datasets fall into three categories: numerical only, categorical only, and mixed. We use the train/test splits provided in LIBSVM website; if splits are not available we use 50/50 splitting with stratification. For multiclass datasets we map the classes to binary labels.

Table III shows the AUC values obtained by six competing algorithms on benchmark datasets. Here the results are reported along with standard deviations. In addition, we once again conduct a pairwise t-test with 95% significance level. To perform this test, we compare each algorithm in the last four columns to the two MBA algorithms in the first two columns. If MBA performs significantly better/worse we represent this with a filled/empty circle. Table III shows that there is a clear benefit in using the proposed MBA, whereas the recent AdaAUC is the second best competitor. The mini-batch processing phase of MBA is learning-rate free and this brings an important advantage. AdaAUC adapts the gradient steps, while we used the learning rate $\mathcal{O}(1/\sqrt{t})$ for all other stochastic gradient algorithms, which performed well with this choice. However, though MBA does not need this parameter, it still performs significantly better than AdaAUC in 9/15 cases. It is also worth noting that MBA-ℓ_1 obtains 100% AUC for the mushrooms data, and for the svmguide3 dataset MBA is at least 9% better than the others. While logistic regression does not directly optimize AUC, it is frequently used in practice, where AUC is the main metric, as it typically has competitive performance. Here we see that logistic regression has a decent performance as well, and in fact beats MBA on the a9a data.

Another important performance measure is the ability to rank as a function of sample size. We show comparisons for this in Figure 2. We see that the stochastic gradient based methods keep improving as the sample size increases, whereas MBA has a relatively steady performance, and it converges faster. This indicates that for these benchmark datasets MBA can already construct a good approximation of the global problem at this point. This result is not surprising given Theorem 4, as good performance is independent of the number of pairs or instances, and only related to the dimensionality of the optimization problem to be solved. In the first panel of Figure 2 the best performer is logistic regression although the difference is rather small. In the second plot, MBA-ℓ_2 gives the best result, although AdaAUC is good as well. For the other two plots, both MBA methods have a clear advantage beginning to end.

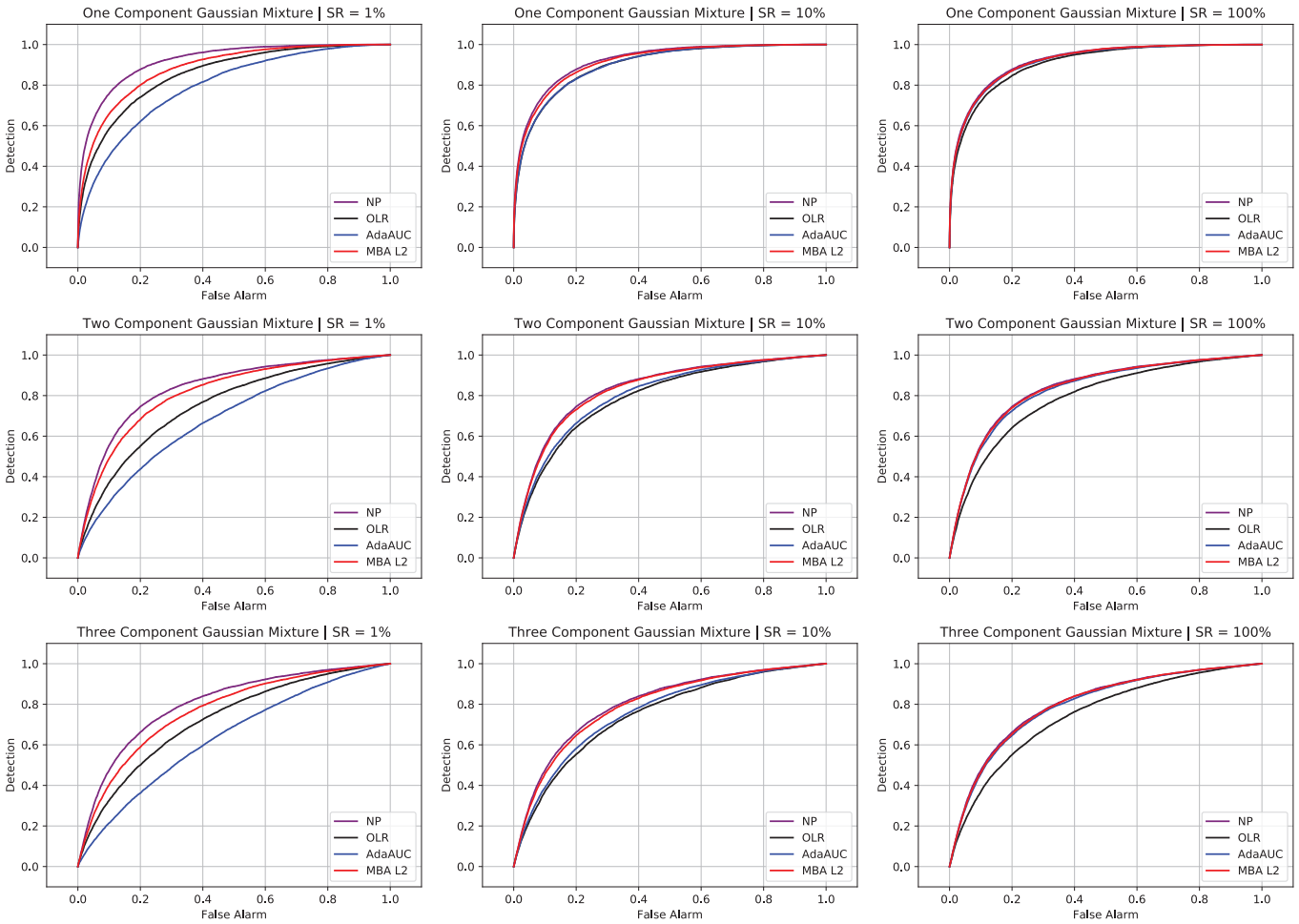

Fig. 1. The ROC curves obtained by the Neyman-Pearson detector and three learning algorithms on the simulated data. The rows are in increasing order of mixture components (k) and the columns are in increasing order of sample ratio (SR).

C. Nonlinear Features

So far we have demonstrated how MBA can achieve better performance than its competitors; however we focused on learning a linear ranking function. As mentioned in Section III, MBA can easily be extended to nonlinear feature spaces; in this case we first perform the feature transformation and apply the algorithm to this new set of features. Here we show an application using three datasets from the previous section: german, mnist, and svmguide3. Our baseline will be the best values obtained by the linear MBA (LIN) from table III. We obtain nonlinear features using a variety of methods: random thresholds to obtain binary features (RT) [35], random Fourier features (RFF) [34] which approximate kernel machines, random neural networks (RNN) where the weights are obtained via randomization, and finally a single layer binary classification network (NN) which is learned from the training data. Note that in the latter case we can also use the network itself to get scores, however we instead feed the representations in the hidden layer to MBA, for demonstration purposes.

For the german and svmguide datasets we use 500 random features, whereas mnist uses 1000 as it has higher dimensionality. For the NN the hidden layer number is half of the input dimension, and both RNN and NN use hyperbolic tangent activation function. NN is trained using Adam optimizer [36]. Given the nonlinear features all the ranking functions are learned with MBA-L2.

We report the AUC values on test set in Figure 3. For the german dataset we see that the linear model has the best performance, and applying a nonlinear feature transform actually degrades performance. This is a common theme in learning nonlinear features; in general there is no principled way of selecting the right feature transform and in our case the approaches we consider fail to find a better feature representation. In fact, for many datasets it may be difficult to find a nonlinearity that would perform better, and consequently linear models remain highly practical. For the mnist dataset, we see that while RT does not provide an improvement, RFF, RNN, and NN all yield substantially higher AUC. This dataset is a widely used benchmark for training neural networks [37] and the features learned by neural nets are highly effective. The best performance in this case is given by NN, which shows that learning representations in a data-dependent manner is the best choice here. Finally, for the svmguide3 dataset we see that the best performance is given by RT. While RFF and NN also

TABLE III
COMPARISON OF ALGORITHMS ON 15 BENCHMARK DATASETS FROM UCI AND LIBSVM REPOSITORIES. THE SYMBOLS FILLED/EMPTY CIRCLE INDICATE ONE OF THE MBA IS (STATISTICALLY) SIGNIFICANTLY BETTER/WORSE.

Dataset	MBA-ℓ_2	MBA-ℓ_1	OLR	SOLR	OAM	AdaAUC
a1a	88.98 ± 0.14	88.67 ± 0.15	• 88.64 ± 0.27	• 88.04 ± 0.14	• 87.61 ± 0.45	• 88.51 ± 0.34
a9a	89.97 ± 0.01	89.97 ± 0.02	◦ 90.17 ± 0.03	• 89.88 ± 0.03	• 89.30 ± 0.22	89.99 ± 0.04
amazon	77.12 ± 0.44	71.35 ± 2.50	• 69.90 ± 2.21	• 71.87 ± 0.74	• 60.23 ± 3.90	• 74.97 ± 0.89
bank	93.22 ± 0.06	93.22 ± 0.03	• 82.89 ± 0.21	• 80.23 ± 0.33	• 81.51 ± 0.49	• 89.46 ± 0.12
codrna	97.68 ± 0.00	97.63 ± 0.01	• 95.69 ± 0.19	• 92.36 ± 0.85	• 97.31 ± 0.12	• 94.34 ± 0.40
german	80.34 ± 0.80	80.41 ± 0.64	• 76.39 ± 1.69	• 75.07 ± 1.22	• 74.59 ± 1.79	• 77.83 ± 1.29
ijcnn	90.53 ± 0.05	90.40 ± 0.07	• 89.50 ± 0.53	• 88.93 ± 0.48	• 88.52 ± 1.76	90.59 ± 0.28
madelon	62.39 ± 0.44	62.34 ± 0.51	61.97 ± 0.69	• 61.81 ± 0.48	• 60.64 ± 0.44	61.82 ± 1.58
mnist	95.81 ± 0.02	95.77 ± 0.02	• 95.63 ± 0.27	• 95.49 ± 0.12	• 94.82 ± 0.23	• 95.47 ± 0.09
mushrooms	100.00 ± 0.00	100.00 ± 0.00	99.88 ± 0.03	• 99.73 ± 0.07	• 99.62 ± 0.28	• 99.98 ± 0.00
phishing	98.32 ± 0.01	98.32 ± 0.05	98.49 ± 0.01	98.38 ± 0.03	• 98.08 ± 0.27	98.36 ± 0.02
svmguide3	81.16 ± 0.80	82.05 ± 0.66	• 63.80 ± 0.81	• 57.65 ± 2.98	• 66.97 ± 3.45	• 69.14 ± 1.95
usps	95.89 ± 0.04	95.83 ± 0.06	• 95.82 ± 0.15	• 95.71 ± 0.07	• 94.65 ± 0.53	• 95.74 ± 0.13
w1a	92.28 ± 0.23	91.21 ± 0.36	• 84.81 ± 1.34	• 79.84 ± 1.15	• 87.83 ± 1.59	• 90.70 ± 0.67
w7a	96.27 ± 0.07	96.17 ± 0.08	• 93.05 ± 0.29	• 89.27 ± 0.82	• 93.92 ± 0.55	• 95.09 ± 0.26
Win/Tie/Loss	-	-	11/3/1	14/1/0	15/0/0	11/4/0

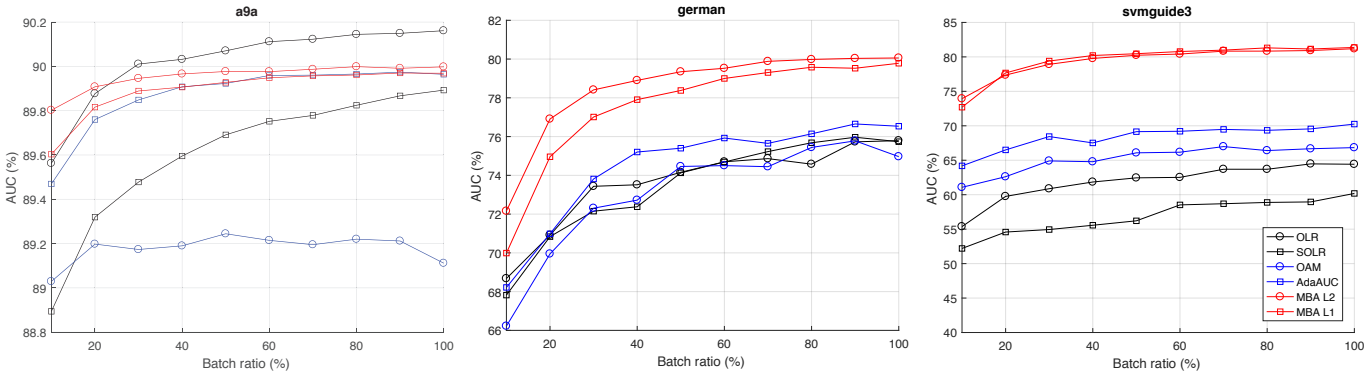

Fig. 2. AUC performance of six algorithms as a function of sample size for a9a, german, and svmguide3 selected from LIBSVM.

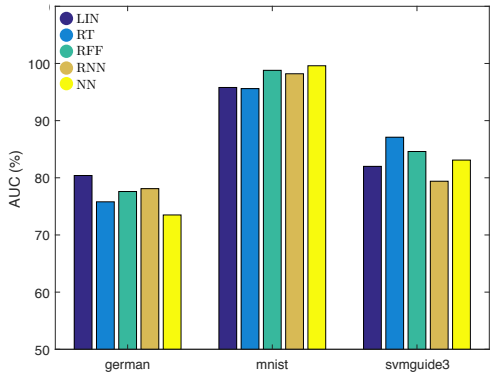

Fig. 3. AUC performance of four nonlinear feature generation methods, compared to the linear case. All of the training is done via MBA-L2.

provide improvement, RT has a clear edge here. This reinforces our previous claim that there is not a one-shot solution to the feature engineering problem and it is rather a process of trial and error.

D. Large-scale Web Click Data

For this last part of experiments we use large scale datasets where the task is click through rate (CTR) prediction. Estimating user clicks in web advertising is one of the premier areas of AUC optimization. As the number of users who click a given ad is typically low, the task naturally manifests itself as distinguishing click from non-click. The datasets used come from Avazu and Criteo, available at LIBSVM; we once again summarize them at the end of Table I. It is worthwhile to note that these datasets are an order of magnitude larger than the ones used in previous studies, showing the scaling benefit MBA brings. This time we shuffle and split the entire dataset into chunks of 100 (Avazu App) and 200 (Avazu Site and Criteo). We then make a single pass over these chunks with randomized sampling and report the results. For these datasets the variation across different runs is very small, as the inputs are very uniform. Therefore we do not show the confidence intervals in the bar charts, but note that all results are statistically significant. For the CTR problem, a 0.1% improvement in AUC is considered significant, whereas an increase of 0.5% results in noticable revenue gain.

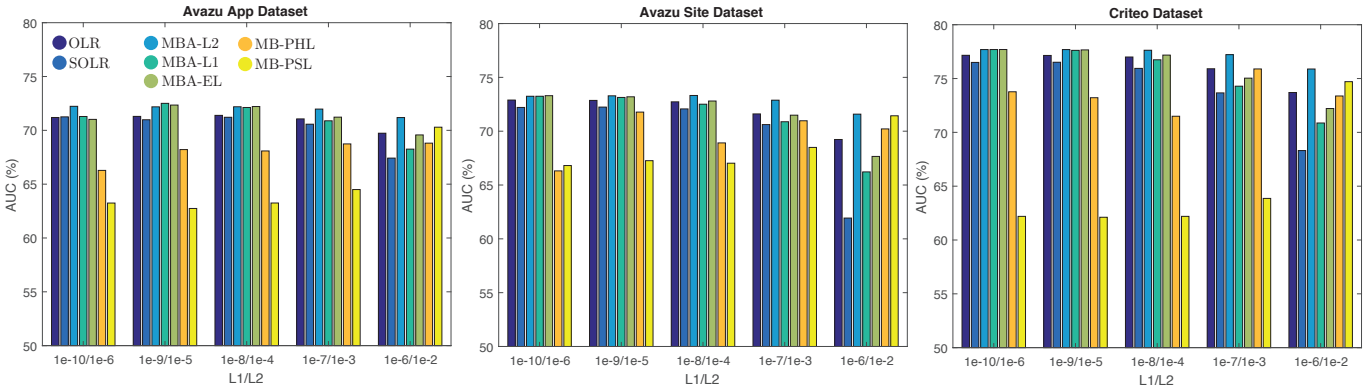

Fig. 4. AUC achieved by all algorithms on the Avazu App, Avazu Site, and Criteo datasets. Here the performance is plotted as a function of regularization parameters. The elastic net uses one half of ℓ_1-penalty for both ℓ_1 and ℓ_2 regularization.

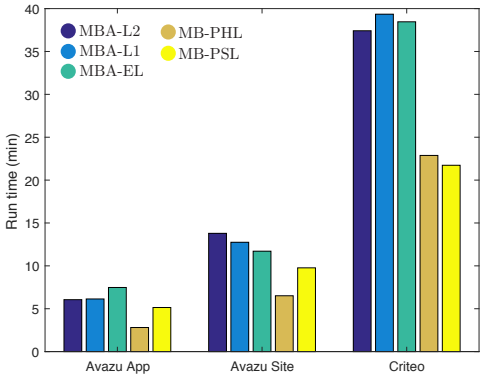

Fig. 5. Runtime comparison of MBA with MB-PSL and MB-PHL. As the latter two only require a gradient computation they are faster than MBA, but with significantly reduced performance. On the other hand MBA can process ten million samples under an hour, which shows the scalability of this approach.

In Figure 4 we show the AUC performance of seven algorithms. For the Avazu App data, MBA-ℓ_2 gives the best results while for Avazu Site and Criteo all MBA algorithms give similar results. Comparing the proposed MBA with the best non-MBA algorithm, the performance improvements are 1.20%, 0.43% and 0.54%. The mini-batch gradient descent algorithms do not perform as well, especially when the regularization parameter is small, and they get better as this parameter increases. For these experiments the step size is $\mathcal{O}(1/\sqrt{t})$, and while logistic regression has good performance with this choice, optimizing pairwise losses seems less robust. As the regularization increases the variation in the gradients decreases, which helps improve the AUC scores. We also experiment with a small constant step size, and this yields similar results. On the other hand MBA does not require this parameter at all. For this reason our algorithm is quite suitable for large-scale problems.

Another important concern is the running time. Here, we do not make a relative comparison, instead we state how much time it takes to find the result. This is because, comparing the running time to logistic regression is not very informative; if a sequential logistic regression is implemented in Python script, then the mini-batch algorithm is roughly 10 times faster, as sequential processing is slow. However, if an optimized package is used, then it can be 100 times faster than MBA, as the underlying code is optimized. For this reason we show the running time of the vanilla implementation of MBA in Figure 5. As it can be seen, even for the Criteo dataset, which contains the largest number of instances, the runtime is under an hour. As we briefly mentioned in Section III, the mini-batch portion of MBA can be distributed without loss of accuracy, therefore using cluster computing, MBA can easily scale to billion-sample datasets, which are several orders of magnitude larger than the datasets that can be handled by sequential methods.

V. Conclusion

This paper has introduced a fast algorithm to optimize the AUC metric. Our proposed approach, called MBA, uses the specific structure of the squared pairwise surrogate loss function. In particular, it is shown that one can approximate the global risk minimization problem simply by approximating the first and second moments of pairwise differences of positive and negative inputs. This suggests an efficient mini-batch scheme, where the moments are estimated by U-statistics. MBA comes with theoretical guarantees, and importantly the number of samples required for good performance is independent of the number of pairs present, which is typically a very large number. Our experiments demonstrate the advantages of MBA in terms of speed and performance. We think MBA would be particularly useful for applications where AUC is the prime metric, and the data size is massive and parallel processing is necessary.

Appendix A
AUC Optimization and Signal Detection

In this section we discuss the connection of AUC optimization to the signal detection framework. This framework is concerned with a probabilistic setup, where the optimal solution with maximum AUC can be obtained in analytical form. This is in contrast to the statistical learning setup which assumes

that the probability distributions generating the observations are unknown to the modeler.

In binary signal detection we have two hypotheses

$$\mathcal{H}^+ \ : \ \boldsymbol{X} \sim \mathcal{P}^+ \quad , \quad \mathcal{H}^- \ : \ \boldsymbol{X} \sim \mathcal{P}^- \tag{28}$$

This setting arises frequently in many different applications, such as radar systems and communication channels [1]. In this setup, it is commonly assumed that the generating distributions $\mathcal{P}^+$ and $\mathcal{P}^-$ are known (c.f. Eq. (1)). For our purposes we consider the *Neyman-Pearson (NP)* hypothesis testing scenario, where neither the priors for hypotheses are known, nor the costs of making a wrong decision are directly available. In this case the optimal detector is designed based on the following two metrics: detection and false alarm, defined for a fixed decision rule as

$$\begin{aligned} \text{Detection: } P_D(\Gamma) &= \int \Gamma(\boldsymbol{x}) \ p^+(\boldsymbol{x}) \ d\boldsymbol{x} \\ \text{False Alarm: } P_F(\Gamma) &= \int \Gamma(\boldsymbol{x}) \ p^-(\boldsymbol{x}) \ d\boldsymbol{x} \end{aligned} \tag{29}$$

Two immediate observations follow: (i) The metrics measure the performance of the rule itself, so they are a function of Γ. (ii) The detector $\Gamma(\boldsymbol{x})$, in turn, is a mapping from the observed signal $\boldsymbol{x}$ to the hypotheses or labels $set+, -$. As the names imply, detection is the probability of correctly choosing the positive hypothesis, whereas false alarm is choosing the positive whereas the correct hypothesis was the negative. In general, the positive hypothesis corresponds to the presence of a target/message, whereas the negative one indicates absence, hence the names.

In NP hypothesis testing, the optimal detector is the solution to the optimization

$$\Gamma'(\boldsymbol{x}) := \arg\max_{\Gamma} P_D(\Gamma) \quad s.t. \quad P_F(\Gamma) \leq \alpha \ . \tag{30}$$

We therefore seek the detector with highest detection probability while setting a limit on the false alarm rate ($0 \leq \alpha \leq 1$). Note that, without this limit (i.e. $\alpha = 1$) we can use a trivial decision rule that maps all observations to positive hypothesis and obtain $P_D(\Gamma) = 1$. The solution of this optimization is given by the following.

Lemma 1 (Neyman-Pearson, [1]): For α, let Γ be any decision rule with $P_D(\Gamma) \leq \alpha$ and let Γ' be the decision rule of form

$$\Gamma'(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad p_1(\boldsymbol{x}) > \eta \ p_0(\boldsymbol{x}) \\ \gamma(x) & \text{if} \quad p_1(\boldsymbol{x}) = \eta \ p_0(\boldsymbol{x}) \\ 0 & \text{if} \quad p_1(\boldsymbol{x}) < \eta \ p_0(\boldsymbol{x}) \end{cases} \tag{31}$$

where $\eta \geq 0$ and $0 \leq \gamma(\boldsymbol{x}) \leq 1$ are chosen such that $P_F(\Gamma') = \alpha$. Then $P_D(\Gamma') \geq P_D(\Gamma)$.

We note that a decision rule that is optimal in the NP sense satisfies the false alarm inequality on the boundary. The structure in Eq. (31) reveals that, for any given input $\boldsymbol{x}$ this rule computes a score based on the likelihood ratio $p_1(x)/p_0(x)$ and compares it to a threshold. Since the ROC curve is the plot of detection vs false alarm, when $\mathcal{P}^+$ and $\mathcal{P}^-$ are known, the likelihood ratio function can be used to obtain scores with maximum AUC.

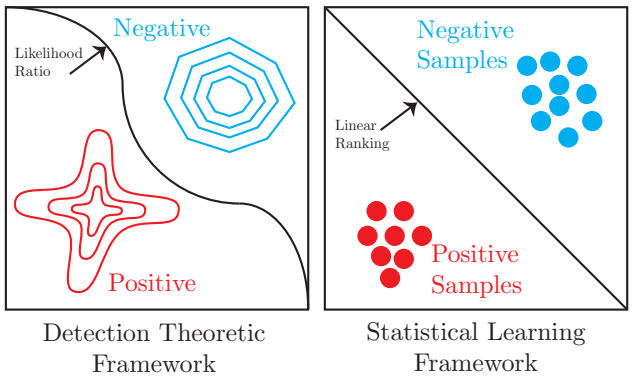

Fig. 6. A cartoon illustration of the signal detection (left) and statistical learning (right) frameworks for AUC maximization.

The framework outlined in this section is displayed in Figure 6, left panel. In general, the likelihood ratio test would yield non-linear decision boundaries. On the other hand, the right panel of Figure 6 displays the statistical learning approach to AUC optimization, where the ϕ-risk is minimized under the linear classifier assumption, i.e. this is what the MBA does.

In contrast to the signal detection problem of left panel, here we only have access to samples, instead of the class-conditional densities. As the figure suggests, the linear scoring function assumption is meaningful when the two classes are linearly separable, which is measured by the signal-to-noise ratio (SNR) in electronic systems.

Appendix B
Proof of Lemma 1

We will use the shorthand $\Delta_{\boldsymbol{w}} = \boldsymbol{w}_N - \boldsymbol{w}_S$, $\Delta_{\boldsymbol{\mu}} = \boldsymbol{\mu}_N - \boldsymbol{\mu}_S$, and $\Delta\boldsymbol{x}_s = (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$ where $\mathcal{S}[s] = (i, j)$. We also recall the Bernstein inequality for a $d \times d$ symmetric, random matrix $\boldsymbol{Z} = \sum_s \boldsymbol{E}_s$ and threshold γ

$$P[\|\boldsymbol{Z}\|_2 > \gamma] \leq 2d \exp\left(\frac{-\gamma^2/2}{\mathbb{V}(\boldsymbol{Z}) + L\gamma/3}\right) \tag{32}$$

where $\|\boldsymbol{E}_s\| \leq L$. The scalar version is recovered by setting $d = 1$.

(i) This part follows the argument for the sample covariance estimator in [23]. For the matrix we can write $\Delta_{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_S - \boldsymbol{\Sigma}_N = \sum_{s \in \mathcal{S}} \frac{1}{S}[\Delta\boldsymbol{x}_s \Delta\boldsymbol{x}_s^\top - \boldsymbol{\Sigma}_N]$. We denote each summand by $\boldsymbol{E}_s = \frac{1}{S}[\Delta\boldsymbol{x}_s \Delta\boldsymbol{x}_s^\top - \boldsymbol{\Sigma}_N]$. It then follows from triangle inequality that

$$\|\boldsymbol{E}_s\|_2 \leq \frac{1}{S}\left[\left\|\Delta\boldsymbol{x}_s \Delta\boldsymbol{x}_s^\top\right\|_2 + \|\boldsymbol{\Sigma}_N\|_2\right] = \frac{4R_x}{S} \ . \tag{33}$$

As each summand is centered and iid, the variance of sum decomposes as $\mathbb{V}(\Delta_{\boldsymbol{\Sigma}}) = \left\|\sum_{s \in \mathcal{S}} \mathbb{E}[\boldsymbol{E}_s^2]\right\|$. We note that the presence of $\boldsymbol{\Sigma}_N$ in the summands does not hinder independence, as this is just a constant quantity. For a single summand the

second moment can be bounded as

$$\begin{aligned}\mathbb{E}[\boldsymbol{E}_s^2] &= \frac{1}{S^2}\boldsymbol{E}\left[\Delta\boldsymbol{x}_s\Delta\boldsymbol{x}_s^\top - \boldsymbol{\Sigma}_N\right]^2 \\ &= \frac{1}{S^2}\left[\boldsymbol{E}\left[\|\Delta\boldsymbol{x}_s\|_2^2\,\Delta\boldsymbol{x}_s\Delta\boldsymbol{x}_s^\top\right] - \boldsymbol{\Sigma}_N^2\right] \\ &\preceq \frac{1}{S^2}\left[2R_x\boldsymbol{\Sigma}_N - \boldsymbol{\Sigma}_N^2\right] \\ &\preceq \frac{2R_x\boldsymbol{\Sigma}_N}{S^2}\ , \end{aligned} \tag{34}$$

from which the variance inequality $\mathbb{V}(\Delta_{\boldsymbol{\Sigma}}) \leq \frac{2R_x\|\boldsymbol{\Sigma}_N\|_2}{S}$ follows. Substituting Eqs. (33) and (34) into Eq. (32) yields the result.

(ii) For this part, the scalar version of Eq. (32) can be used. We have $\Delta_\sigma = (\boldsymbol{w}_N - \boldsymbol{w}_S)^\top(\boldsymbol{\mu}_N - \boldsymbol{\mu}_S) = \sum_{s\in\mathcal{S}}\frac{1}{S}\left[(\boldsymbol{w}_N - \boldsymbol{w}_S)^\top(\boldsymbol{\mu}_N - \boldsymbol{x}_s)\right]$ where we denote each *scalar* summand as $e_s = \frac{1}{S}\left[(\boldsymbol{w}_N - \boldsymbol{w}_S)^\top(\boldsymbol{\mu}_N - \boldsymbol{x}_s)\right]$. Once again it is straightforward to verify each summand is centered and iid. An ℓ_1-norm bound can be obtained by Cauchy-Schwarz inequality

$$\begin{aligned}|e_s| &= \left|\frac{1}{S}(\boldsymbol{w}_N - \boldsymbol{w}_S)^\top(\boldsymbol{\mu}_N - \boldsymbol{x}_s)\right| \\ &\leq \frac{1}{S}\|\boldsymbol{w}_N - \boldsymbol{w}_S\|_2\,\|\boldsymbol{\mu}_N - \boldsymbol{x}_s\|_2 \\ &= \frac{4\sqrt{R_xR_w}}{S}\ . \end{aligned} \tag{35}$$

For the variance we once again have the decomposition $\mathbb{V}(\Delta_\sigma) = \sum_{s\in\mathcal{S}}\mathbb{E}[e_s^2]$ and for a single term we have

$$\begin{aligned}\mathbb{E}[e_s^2] &= \mathbb{E}\left[\left[\frac{1}{S}(\boldsymbol{w}_N - \boldsymbol{w}_S)^\top(\boldsymbol{\mu}_N - \boldsymbol{x}_s)\right]^2\right] \\ &= \frac{1}{S^2}\Big[\mathbb{E}[\Delta_{\boldsymbol{w}}^\top\boldsymbol{x}_s\boldsymbol{x}_s^\top\Delta_{\boldsymbol{w}}] + \frac{1}{S^2}\mathbb{E}[\Delta_{\boldsymbol{w}}^\top\boldsymbol{\mu}_N\boldsymbol{\mu}_N^\top\Delta_{\boldsymbol{w}}] \\ &\quad - \frac{1}{S^2}\mathbb{E}[\Delta_{\boldsymbol{w}}^\top\boldsymbol{x}_s\boldsymbol{\mu}_N^\top\Delta_{\boldsymbol{w}}] - \frac{1}{S^2}\mathbb{E}[\Delta_{\boldsymbol{w}}^\top\boldsymbol{\mu}_N\boldsymbol{x}_s^\top\Delta_{\boldsymbol{w}}]\Big] \\ &= \frac{1}{S^2}\Delta_{\boldsymbol{w}}^\top\boldsymbol{\Sigma}_N\Delta_{\boldsymbol{w}} - \frac{1}{S^2}\Delta_{\boldsymbol{w}}^\top\boldsymbol{\mu}_N\boldsymbol{\mu}_N^\top\Delta_{\boldsymbol{w}} \\ &\leq \frac{2R_w\,\|\boldsymbol{\Sigma}_N\|_2}{S^2} \end{aligned} \tag{36}$$

where for the last line we used the upper bounds $\|\Delta_{\boldsymbol{w}}\|_2^2 \leq 2R_w$ and $\Delta_{\boldsymbol{w}}^\top\boldsymbol{\Sigma}_N\Delta_{\boldsymbol{w}} \leq \|\Delta_{\boldsymbol{w}}\|_2^2\|\boldsymbol{\Sigma}_N\|_2^2$ and dropped the negative term. Note that the first upper bound holds by triangle inequality, and the second one follows from the maximum eigenvalue bound of a quadratic form. Plugging Eqs. (35) and (36) into Eq. (32) we get the desired result. ■

REFERENCES

[1] H. Vincent Poor, *An Introduction to Signal Detection and Estimation*, Springer, 1998.

[2] Daniel J. Lingenfelter, Jeffrey A. Fessler, Clayton D. Scott, and Zhong He, "Asymptotic source detection performance of gamma-ray imaging systems under model mismatch," *IEEE Transactions on Signal Processing*, 2011.

[3] Jianshu Chen, Zaid J. Towfic, and Ali H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, 2015.

[4] Satish G. Iyengar, Pramod K. Varshney, and Thyagaraju Damarla, "A parametric copula-based framework for hypothesis testing using heterogeneous data," *IEEE Transactions on Signal Processing*, 2011.

[5] Hao Chen, Pramod K. Varshney, Steven M. Kay, and James H. Michels, "Theory of the stochastic resonance effect in signal detection: Part I - Fixed detectors," *IEEE Transactions on Signal Processing*, 2007.

[6] Hao Chen, Pramod K. Varshney, Steven M. Kay, and James H. Michels, "Theory of the stochastic resonance effect in signal detection: Part II - Variable detectors," *IEEE Transactions on Signal Processing*, 2008.

[7] Suat Bayram, San Gultekin, and Sinan Gezici, "Noise enhanced hypothesis testing according to restricted neyman pearson criterion," *Digital Signal Processing*, 2014.

[8] Xia Hong, Sheng Chen, and Chris J. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Transactions on Neural Networks*, 2007.

[9] Cristiano L. Castro and Antonio P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, 2013.

[10] Minlong Lin, Ke Tang, and Xin Yao, "Dynamic sampling approach to training neural networks for multiclass imbalance classification," in *IEEE Transactions on Neural Networks and Learning Systems*, 2013.

[11] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica, "Ad click prediction: a view from the trenches," in *Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

[12] Ulf Brefeld and Tobias Scheffer, "Auc maximizing support vector learning," in *ICML Workshop on ROC Analysis in Machine Learning*, 2005.

[13] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer, "An efficient boosting algorithm for combining preferences," *Journal of Machine Learning Research*, 2003.

[14] Cesar Ferri, Peter Flach, and Jose Hernandez-Orallo, "Learning decision trees using the area under the roc curve," in *International Conference on Machine Learning (ICML)*, 2012.

[15] Tong Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *International Conference on Machine Learning (ICML)*, 2004.

[16] Peilin Zhao, Steven C. H. Hoi, Rong Jin, and Tianbao Yang, "Online AUC maximization," in *International Conference on Machine Learning (ICML)*, 2011.

[17] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou, "One-pass AUC optimization," in *International Conference on Machine Learning (ICML)*, 2013.

[18] Yi Ding, Peilin Zhao, Steven C. H. Hoi, and Yew Soon Ong, "An adaptive gradient method for online auc maximization," in *Association for Advancement of Artificial Intelligence (AAAI)*, 2015.

[19] Junjie Hu, Haikin Yang, Michael Lyu, Irwin King, and Anthony So, "Kernelized online imbalanced learning with fixed budgets," in *Association for Advancement of Artificial Intelligence (AAAI)*, 2015.

[20] Junjie Hu, Haikin Yang, Michael Lyu, Irwin King, and Anthony So, "Online nonlinear auc maximization for imbalanced data sets," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[21] Yi Ding, Chenghao Liu, Peilin Zhao, and Steven C.H. Hoi, "Large scale kernel methods for online auc maximization," in *International Conference on Data Mining (ICDM)*, 2017.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[23] Joel Tropp, "An introduction to matrix concentration inequalities," *Foundations and Trends in Machine Learning*, 2015.

[24] Stephan Clemencon, Gabor Lugosi, and Nicolas Vayatis, "Ranking and empirical minimization of u-statistics," *The Annals of Statistics*, 2008.

[25] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of machine learning*, MIT Press, 2012.

[26] Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, 2006.

[27] Wei Gao and Zhi-Hua Zhou, "On the consistency of AUC pairwise optimization," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[28] Majdi Khalid, Indrakshi Ray, and Hamidreza Chitsaz, "Confidence-weighted bipartite ranking," in *Advanced Data Mining and Applications*, 2016.

[29] Hui Zou and Trevor Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology*, 2005.

[30] Aad W. Van der Vaart, *Asymptotic Statistics*, Cambridge University Press, 1998.

[31] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, May 2013.
[32] Nicolo Cesa-Bianchi and Gabor Lugosi, *Prediction, learning, and games*, Cambridge University Press, 2006.
[33] Mert Pilanci and Martin J. Wainwright, "Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares," *Journal of Machine Learning Research*, 2016.
[34] Ali Rahimi and Ben Recht, "Random features for large scale kernel machines," in *Advances in Neural Information Processing Systems (NIPS)*, 2008.
[35] Ali Rahimi and Ben Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2008.
[36] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2014.
[37] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.