Mini-Batch AUC Optimization

San Gultekin
Columbia University
New York, NY, US
sg3108@columbia.edu

Avishek Saha*
Amazon
Palo Alto, CA, US
avisaha@a9.com

Adwait Ratnaparkhi
Roku Inc.
Los Gatos, CA, US
adwait_ratnaparkhi@yahoo.com

John Paisley
Columbia University
New York, NY, US
jpaisley@columbia.edu

ABSTRACT

Area under the receiver operating characteristics curve (AUC) is an important metric for a wide range of machine learning problems; including the well-known click through rate prediction for online sponsored product ads. Scalable methods for optimizing AUC have recently been proposed; however, handling very large datasets remains an open challenge. 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. Extensive experiments show the practical utility of the proposed method.

ACM Reference Format:
San Gultekin, Avishek Saha, Adwait Ratnaparkhi, and John Paisley. 2018. Mini-Batch AUC Optimization. In *Proceedings of ACM SigKDD Conference Workshop (AdKDD).* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Given a set of positive and negative inputs, the bipartite ranking problem is concerned with building a scoring system, such that the positives are ranked higher than the negatives. 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 the negative one. Therefore, it is a natural measure for the bipartite ranking accuracy.

Bipartite ranking performance is an important metric for datasets with imbalanced labels; e.g. where one is given a dataset with binary labels in which the ratio of positive to negative samples is very low. 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 values are more informative. For this reason, the AUC metric is heavily used for website ad click prediction problems [15], where only a very small fraction of web history contains ads clicked by visitors.

Given that AUC is the primary performance 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 [2], boosting [7], and decision trees [6]. 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*, which has a quadratic growth.

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 [17]. The first work in this direction is Zhao et al. [18], where an online method based on proxy hinge loss is proposed. Later, Gao et al. [8] use the pairwise squared loss function, which eliminates the need for buffering previous instances. Ding et al. [5] propose adaptive gradient/subgradient methods which can also handle sparse inputs, while Hu et al. [12, 13] consider the nonlinear AUC maximization problem using kernel and multiple-kernel methods. Most recently, Ding et al. [4] focus on scalable kernel methods.

While these approaches can significantly increase scalability, for very large datasets their sequential nature can still be problematic. 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. U-processes for ranking problems have previously been explored by Clemencon et al. [3]. However, scalable mini-batch algorithms using U-statistics have not been developed. The nearest work of Ding et al. [4] uses similar mini-batch techniques, but for gradient descent. The proposed method comes with theoretical guarantees, and the number of samples required for good performance does not have a quadratic dependence on the size of dataset. Our experiments show the practical utility and performance improvement of MBA.

*This work is done when the author was affiliated with Yahoo! Inc.

AdKDD, August 2018, London, United Kingdom

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

Given a dataset from an input space $\mathcal{X}$ and outputs space $\mathcal{Y} = \{\pm 1\}$, let $\mathcal{P}$ be the unknown distribution which generates these observation pairs. Given a score function $f : \mathcal{X} \to \mathbb{R}$ the bipartite ranking accuracy of such function is naturally measured by AUC

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

This expectation is the probability that a positive instance is ranked higher than negative instance.However, optimizing AUC gives rise to an NP-hard problem since the objective function in Eq. (1) is a sum of indicators and the number of pairs grows quadratically with the training data. To sidestep this difficulty, a surrogate loss function $\phi(\cdot)$ can be chosen: Replacing $\mathbb{1}\left[f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-) > 0\right]$ with the pairwise convex surrogate loss $\phi(\boldsymbol{x}^+, \boldsymbol{x}^-) = \phi(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-))$, the aim is to minimize the ϕ-risk [16]

$$R_\phi(f) = \mathbb{E}_{\substack{\boldsymbol{x}^+ \sim \mathcal{P}^+ \\ \boldsymbol{x}^- \sim \mathcal{P}^-}} \left[\phi(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-)) \right]. \tag{2}$$

This is the Bayes risk of the scoring function [1]. 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) [9]:

$$\begin{aligned} &\phi_{\text{PSL}}(t) = (1-t)^2\,,\ \phi_{\text{PHL}}(t) = \max(0,\ 1-t)\,, \\ &\phi_{\text{PEL}}(t) = \exp(-t)\,,\ \phi_{\text{PLL}}(t) = \log(1+\exp(-t))\,, \end{aligned} \tag{3}$$

where $t := f(\boldsymbol{x}_i^+) - f(\boldsymbol{x}_j^-)$ is the pairwise scoring difference. Among the recent works on AUC optimization, Zhao et al. [18] and Khalid et al. [14] use PHL, whereas Gao et al. [8] and Ding et al. [5] focus on PSL.

If we further constrain the scoring function to be linear in input features [1], i.e. $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$; the ϕ-risk becomes

$$\begin{aligned} R_\phi(f) &= \mathbb{E}_{\substack{\boldsymbol{x}^+ \sim \mathcal{P}^+ \\ \boldsymbol{x}^- \sim \mathcal{P}^-}} \left[\Big(1 - \boldsymbol{w}^\top (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \Big)^2 \right] \\ &= 1 - 2\boldsymbol{w}^\top \mathbb{E}\left[(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \right] \\ &\qquad + \boldsymbol{w}^\top \mathbb{E}\left[(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top \right] \boldsymbol{w} \\ &= 1 - 2\boldsymbol{w}^\top \boldsymbol{\mu} + \boldsymbol{w}^\top \Sigma \boldsymbol{w}, \end{aligned} \tag{4}$$

where we define $\boldsymbol{x}_{ij} := (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$, and $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}_{ij}]$ and $\Sigma = \mathbb{E}[\boldsymbol{x}_{ij}\boldsymbol{x}_{ij}^\top]$ are the first and second moments of $\boldsymbol{x}_{ij}$. We finally define the solution to the ϕ-risk minimization problem as

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}, \tag{5}$$

where we multiply by 1/2 for notation reasons. Note that $\boldsymbol{\mu}$ and Σ characterize the first and second order statistics of the pairwise differences. This is important because the quality of bipartite ranking does not directly depend on the positive and negative features, but the differences between them. This observation forms the basis of our mini-batch algorithm. Finally, by definition Σ is positive semi-definite and when it is positive definite, there is a unique $\boldsymbol{w}^\star$ that satisfies Eq. (5).

[1] The extension to any feature-transformed space is trivial.

3 MINI-BATCH AUC OPTIMIZATION

Using the pairwise squared loss function we obtain a convex optimization problem in place of the original NP-hard problem. However, Eq. (5) is still difficult since computing the first and second order statistics rely on the knowledge of 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 substitute the following empirical risk

$$\widehat{R}_\phi(f) = \frac{1}{2N_+N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} \phi(f(\boldsymbol{x}_i^+) - f(\boldsymbol{x}_j^-)), \tag{6}$$

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. (6) becomes intractable.

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

$$\begin{aligned} \widehat{R}_\phi(\boldsymbol{w}) = &-\boldsymbol{w}^\top \left[\frac{1}{N_+N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \right] \\ &+ \frac{1}{2}\boldsymbol{w}^\top \left[\frac{1}{N_+N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top \right] \boldsymbol{w} \end{aligned} \tag{7}$$

and define

$$\begin{aligned} \boldsymbol{\mu}_N &= \frac{1}{N_+N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \\ \Sigma_N &= \frac{1}{N_+N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\,(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top. \end{aligned} \tag{8}$$

The variables in Eq. (8) are sample approximations to the first and second moments of the pairwise differences, which are substituted for $\boldsymbol{\mu}$ and Σ in Eq. (5). Overall, the optimization problem to be solved is

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \Sigma_N \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}_N + R_{\text{EL}}(\boldsymbol{w}), \tag{9}$$

where $R_{\text{EL}}(\boldsymbol{w}) := \lambda_1 \|\boldsymbol{w}\|_1 + (1/2)\lambda_2 \|\boldsymbol{w}\|_2^2$ is the elastic net regularizer [19], which we add to prevent overfitting. Note that, unlike Eq. (5), there is 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

$$U_n = \binom{n}{m}^{-1} \sum_{C_{n,m}} h(X_1, \ldots, X_n), \tag{10}$$

where $C_{n,m}$ is the set of all length-m combinations with increasing indices.

3.1 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. (10) as

$$\begin{aligned} \boldsymbol{\mu}_t &:= \frac{1}{B} \sum_{(i,j)\in\mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \\ \Sigma_t &:= \frac{1}{B} \sum_{(i,j)\in\mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\,(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top. \end{aligned} \tag{11}$$

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

$$\begin{aligned} \boldsymbol{\mu}_S &:= \frac{1}{BT} \sum_{(i,j)\in\mathcal{S}_{1:T}} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-) \\ \Sigma_S &:= \frac{1}{BT} \sum_{(i,j)\in\mathcal{S}_{1:T}} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\,(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top, \end{aligned} \tag{12}$$

and the optimization problem constructed by MBA is

$$\boldsymbol{w}_S^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \Sigma_S \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{\mu}_S + R_{\mathrm{EL}}(\boldsymbol{w}). \tag{13}$$

This is the function MBA aims to construct and solve, which itself is an approximation to the global risk minimization problem in Eq. (5). 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 (i) it constructs a global problem first, and (ii) it is learning rate free. We summarize the proposed MBA in Algorithm 1.

Mini-batch optimizations are heavily employed in machine learning, including training of deep neural networks [10] and scalable Bayesian inference [11]. 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 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.

Algorithm 1 Mini-Batch AUC Optimization (MBA)

Require: $B, T, \lambda_1, \lambda_2$
Input: X^+, X^-
Output: $\boldsymbol{w}^\star$
Initialize $\boldsymbol{\mu}_S = \mathbf{0}$ and $\Sigma_S = \mathbf{0}$.
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 $\mathcal{S}_t(i) = (\mathcal{S}_t^+(i), \mathcal{S}_t^-(i)),\ i = 1, \ldots, B.$
  $\boldsymbol{\mu}_S \leftarrow \boldsymbol{\mu}_S + \frac{1}{BT}\sum_{(i,j)\in\mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)$
  $\Sigma_S \leftarrow \Sigma_S + \frac{1}{BT}\sum_{(i,j)\in\mathcal{S}_t} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)\,(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)^\top$
end for
$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \Sigma_S \boldsymbol{w}^\top - \boldsymbol{w}^\top \boldsymbol{\mu}_S + \lambda_1 \|\boldsymbol{w}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{w}\|_2^2$

3.2 Theoretical Analysis

Solving the regularized empirical risk minimization problem in Eq. (9) 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. (13). We now provide a bound that the solution with S samples is close to the truth with $S \ll N$, where the closeness is naturally measured by Euclidean distance $d(\boldsymbol{w}_N - \boldsymbol{w}_S) = \|\boldsymbol{w}_N - \boldsymbol{w}_S\|_2^2$.

We now introduce ℓ_2-norm bounds on data and weight vectors, and for any given input we assume that $\|\boldsymbol{x}\|_2^2 \le R_x$.[2] Next, define the upper bound on weights such that $\max\{\|\boldsymbol{w}_N\|_2^2, \|\boldsymbol{w}_S\|_2^2\} \le R_w$. Note here that $R_w < \infty$ is guaranteed by the ℓ_2 regularization of elastic net. We have the following result.

Theorem 1: Let $\boldsymbol{w}_S^\star$ be the solution returned by MBA using S samples. For $\epsilon > 0$, if

$$S \ge \max\left\{ \log(4d/p)\frac{[48R_w^2\,\|\Sigma_N\|_2 + 16\epsilon R_w]R_x}{3\epsilon^2}, \log(4/p)\frac{48R_w\,\|\Sigma_N\|_2 + 16\epsilon\sqrt{R_x R_w}}{3\epsilon^2} \right\}$$

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

The proof is deferred to the full-length version of this paper. Theorem 1 shows that the number of samples S required to guarantee $\|\boldsymbol{w}_N - \boldsymbol{w}_S\|_2 < \delta$ 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. We mention that, while this result holds for linear input spaces, it readily extends to the finite dimensional nonlinear transforms; as all these transformations are mappings from d dimensions to F dimensions, given such fixed transformation, the result holds where we replace d by F.

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

Table 1: 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	5,000	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

4 EXPERIMENTS

In this section we conduct two types of experiments to demonstrate the performance of MBA. In the first part we use 15 frequently used benchmark datasets from the UCI[3] and LIBSVM[4] repositories. In the second part we consider large scale click through rate (CTR) prediction with two publicly available commercial-size datasets with tens of millions of samples. We summarize all datasets in Table 1. We note that the large datasets used (Avazu and Criteo) are an order of magnitude larger than the ones used in previous studies.

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 method using ridge, lasso, and elastic net respectively. OLR is simply the online logistic regression and SOLR is the sparse regression algorithm presented in [15]. OAM is the first proposed online AUC maximization algorithm using stochastic gradients [18] which uses PHL. On the other hand, AdaAUC is the adaptive gradient AUC maximization algorithm in Ding et al. [5] based on PSL.

We also 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 Ding et al. [4]. Finally MB-PSL is a mini-batch gradient method that uses PSL.

4.1 UCI and LIBSVM Benchmark Data

Table 2 shows the AUC values obtained by six competing algorithms on 15 benchmark datasets. Here the results are reported along with standard deviations. In addition, we conduct a pairwise t-test with 95% significance level, as proposed and used by Gao et al. [8]. 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 2 shows that there is a clear benefit in using the proposed MBA algorithm, whereas the recent AdaAUC algorithm 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 $O(1/\sqrt{t})$ 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.

Another important performance measure is the ability to rank as a function of sample size. We show comparisons for this in Figure 1. We see that the stochastic gradient based methods keep improving as the sample size increases, whereas MBA has a relatively steady performance after having a sample size that is 50-60% percent of the original sample size (not the number of pairs). 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 1, 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 1 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 last dataset, both MBA methods have a clear advantage, which shows that optimizing the global cost function is important.

4.2 Large-scale Web Click Data

For this set of experiments 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 click through rate problem, a 0.1% improvement is considered significant, whereas an improvement of 0.5% results in notable revenue gain. In Figure 2 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 $O(1/\sqrt{t})$, 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

[3] archive.ics.uci.edu/ml/
[4] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Table 2: Comparison of algorithms on 15 benchmark datasets from UCI and LIBSVM repositories. The filled/empty circle symbols indicate one of the MBA algorithms 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

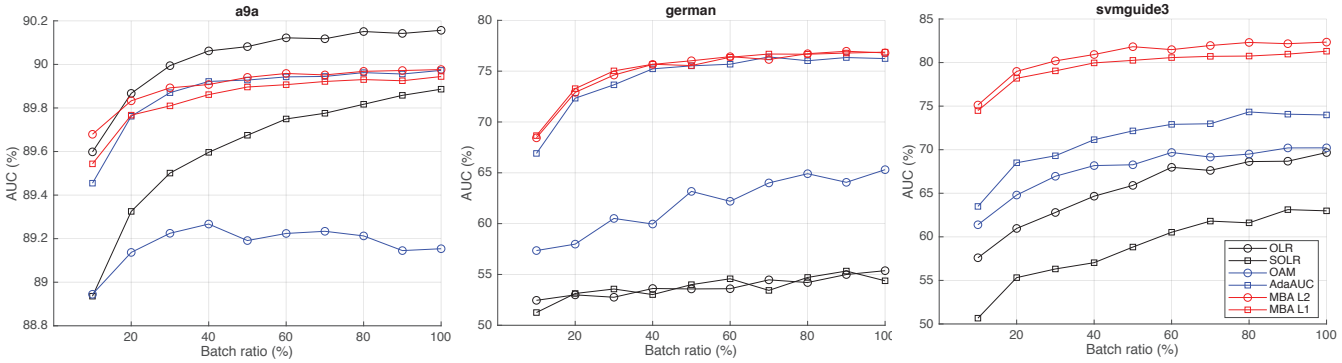

Figure 1: AUC performance of six algorithms as a function of sample size for a9a, german, and svmguide3 selected from LIBSVM.

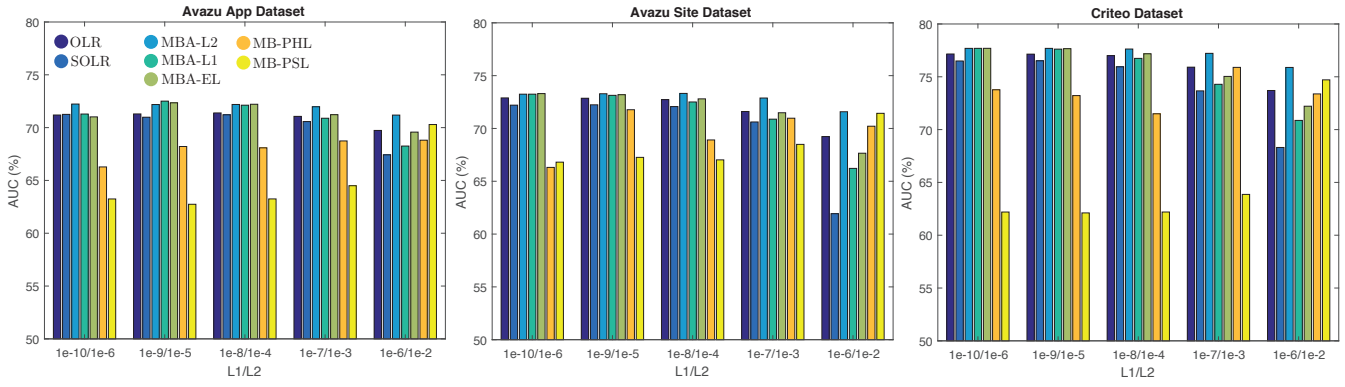

Figure 2: AUC achieved by all algorithms on the Avazu App, Avazu Site, and Criteo datasets. Here the performance is plotted as a function of regularization parameters. The elastic net uses one half of ℓ_1-penalty for both ℓ_1 and ℓ_2 regularization.

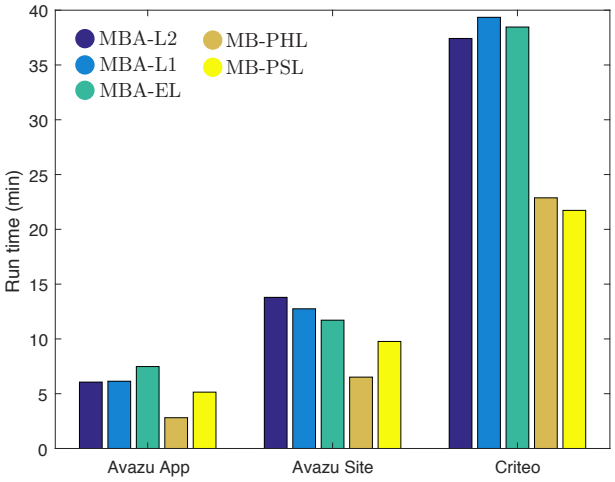

Figure 3: Runtime comparison of MBA with MB-PSL and MB-PHL. As the latter two only require a gradient computation they are faster than MBA, but with significantly reduced performance. On the other hand MBA can process ten million samples under an hour, which shows the scalability of this approach.

results. On the other hand MBA does not require this parameter, which is an important advantage.

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 3. 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 3, 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.

5 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.

REFERENCES

[1] Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. 2006. Convexity, classification, and risk bounds. *J. Amer. Statist. Assoc.* (2006).
[2] Ulf Brefeld and Tobias Scheffer. 2005. AUC maximizing support vector learning. In *ICML Workshop on ROC Analysis in Machine Learning*.
[3] Stephan Clemencon, Gabor Lugosi, and Nicolas Vayatis. 2008. Ranking and empirical minimization of U-statistics. *The Annals of Statistics* (2008).
[4] Yi Ding, Chenghao Liu, Peilin Zhao, and Steven C.H. Hoi. 2017. Large Scale Kernel Methods for Online AUC Maximization. In *International Conference on Data Mining (ICDM)*.
[5] Yi Ding, Peilin Zhao, Steven C. H. Hoi, and Yew Soon Ong. 2015. An Adaptive Gradient Method for Online AUC Maximization. In *Association for Advancement of Artificial Intelligence (AAAI)*.
[6] Cesar Ferri, Peter Flach, and Jose Hernandez-Orallo. 2012. Learning Decision Trees Using the Area Under the ROC Curve. In *International Conference on Machine Learning (ICML)*.
[7] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 2003. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research* (2003).
[8] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. 2013. One-Pass AUC Optimization. In *International Conference on Machine Learning (ICML)*.
[9] Wei Gao and Zhi-Hua Zhou. 2015. On the Consistency of AUC Pairwise Optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
[11] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. 2013. Stochastic Variational Inference. *Journal of Machine Learning Research* 14, 1 (May 2013), 1303–1347.
[12] Junjie Hu, Haikin Yang, Michael Lyu, Irwin King, and Anthony So. 2015. Kernelized Online Imbalanced Learning with Fixed Budgets. In *Association for Advancement of Artificial Intelligence (AAAI)*.
[13] Junjie Hu, Haikin Yang, Michael Lyu, Irwin King, and Anthony So. 2016. Online Nonlinear AUC Maximization for Imbalanced Data Sets. *IEEE Transactions on Neural Networks and Learning Systems* (2016).
[14] Majdi Khalid, Indrakshi Ray, and Hamidreza Chitsaz. 2016. Confidence-Weighted Bipartite Ranking. In *Advanced Data Mining and Applications*.
[15] 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. 2013. Ad click prediction: a view from the trenches. In *Conference on Knowledge Discovery and Data Mining (KDD)*.
[16] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of machine learning*. MIT Press.
[17] Tong Zhang. 2004. Solving Large Scale Linear Prediction Problems using Stochastic Gradient Descent Algorithms. In *International Conference on Machine Learning (ICML)*.
[18] Peilin Zhao, Steven C. H. Hoi, Rong Jin, and Tianbao Yang. 2011. Online AUC Maximization. In *International Conference on Machine Learning (ICML)*.
[19] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology* (2005).