An Adaptive Moment estimation method for online AUC maximization

Xin Liu, Zhisong Pan*, Haimin Yang, Xingyu Zhou, Wei Bai, Xianghua Niu

1 College of Command and Control, Army Engineering University of PLA, Nanjing, China, 2 College of Communication Engineering, Army Engineering University of PLA, Nanjing, China, 3 Beijing Institute of Applied Meteorology, Beijing, China

* hotpzs@hotmail.com

Abstract

Area Under the ROC Curve (AUC) is a widely used metric for measuring classification performance. It has important theoretical and academic values to develop AUC maximization algorithms. Traditional methods often apply batch learning algorithm to maximize AUC which is inefficient and unscalable for large-scale applications. Recently some online learning algorithms have been introduced to maximize AUC by going through the data only once. However, these methods sometimes fail to converge to an optimal solution due to the fixed or rapid decay of learning rates. To tackle this problem, we propose an algorithm AdmOAM, Adaptive Moment estimation method for Online AUC Maximization. It applies the estimation of moments of gradients to accelerate the convergence and mitigates the rapid decay of the learning rates. We establish the regret bound of the proposed algorithm and implement extensive experiments to demonstrate its effectiveness and efficiency.

Introduction

AUC [1] plays an important role in measuring classification performance, and quantifies the ability of a classifier that assigns a higher score for a randomly chosen positive instance than a randomly drawn negative instance [2]. Compared with accuracy and cross-entropy loss, AUC is independent of the priori class probability distribution and misclassification costs, which makes it more favourable for imbalanced classification tasks [3–6]. Moreover, AUC is largely applied in many real-world scenarios like cancer diagnosis and anomaly detection [7, 8].

In recent decades, many batch learning algorithms [9–12] have been introduced to optimize AUC directly. Despite the success of these batch AUC optimization algorithms, they all require the whole training instances available before training. Besides, they update the model every epoch with all training instances. Therefore, it is not efficient and scalable for large-scale applications in batch learning setting. To address this challenge, online learning technique has been introduced to maximize AUC, which has been shown to be capable for large-scale scenarios [13–15]. The online learning methods update the model with only one instance each
epoch. As a result, it is desirable to apply online learning algorithms for handling large-scale streaming data which arrives sequentially.

However, the task of AUC optimization requires minimizing the sum of the losses between instances from different classes. Therefore, it is difficult to maximize AUC by directly applying online learning, which requires to obtain all previous training instances at current iteration for calculating the sum of pairwise losses. Several recent works [16–19] adopt different approximations of the sum of pairwise losses to avoid storing all received training data, which makes them more feasible for large-scale tasks. In general, there are two kinds of online AUC maximization frameworks. The first framework uses reservoir sampling method that keeps fixed buffers to store some historical instances for calculating pairwise losses [18, 19]. The other framework employs one-pass technique to maximize AUC by processing each instance only once [17]. The work [16] proposed an adaptive one-pass online AUC maximization algorithm called AdaOAM. This method adjusts the learning rates of different dimensions to the geometry of data by applying an adaptive gradient method (AdaGrad) [20].

Despite AdaOAM has achieved good performance, its learning rate may shrink too fast due to the rapid increase of its denominators. As a result, the model may fail to fully converge [21]. To tackle this problem, we propose an algorithm AdmOAM, Adaptive Moment estimation method for Online AUC Maximization. It applies the estimation of moments of gradients to adaptively calculate the learning rates dimensionally based on the framework of [22]. Our method mitigates the rapid decay of learning rates by using exponential moving averages of past gradients as the denominators. Furthermore, AdmOAM is efficient and only requires to store the first and second moments of gradients. Based on the theoretical analysis of AdmOAM, we have found that the regret bound of AdmOAM stays much lower than existing non-adaptive methods and is comparable with AdaOAM. We have also shown the effectiveness of the proposed AdmOAM in experiments on several benchmark datasets in comparison with 4 state-of-the-art online AUC maximization algorithms.

The rest of the paper is organized as follows. We first give an overview of some related works. Then we describe the problem setting and the framework of AdmOAM. We then give the theoretical analysis and provide the experimental results. Finally, we present a summary and some directions for future work.

Related work
In this section, we briefly review three prior works in related topics: online learning, adaptive gradient methods and AUC maximization.

Online learning
Online learning is a type of efficient and scalable machine learning algorithm that updates the model from data sequentially. Compared to the traditional batch or offline learning algorithms, online learning algorithms avoid the time-consuming training process and are capable of retraining efficiently at the arrival of new data [13]. The first online learning algorithm is the Perceptron [23], which updates the model with the first-order gradient information. Passive-Agressive is another type of first-order online algorithm, which applies the margin-based technique [24]. In recent years, several second-order online learning algorithms have been proposed to accelerate the convergence of optimization [25, 26]. Besides, some regularization terms have been introduced to stabilize the online learning models [11, 27]. However, these traditional online approaches are aimed at optimizing the classification accuracy or error rate,
which are inappropriate for imbalanced classification tasks. In contrast, we develop a novel first-order online algorithm by maximizing a imbalanced metric with adaptive gradient method.

Adaptive gradient methods

Online Gradient Descent (OGD) [28] is the dominant method for solving the online convex optimization problems. It updates a model by moving the parameters along the direction opposite the gradient of the loss function with a global learning rate. However, infrequently occurring features are highly informative and require relatively larger learning rates than frequently occurring features. Therefore, OGD can not fully incorporate the knowledge of the geometry of the data with the global learning rate. To tackle this challenge, some researchers have proposed several variants of OGD that perform adaptive gradient optimization by adjusting the learning rates on a per-feature basis iteratively [20, 22, 29]. The most famous adaptive gradient algorithm is Adagrad [20], which can achieve better performance than non-adaptive algorithms both theoretically and experimentally. However, Adagrad has been observed to diverge due to the rapid decay of the learning rates since the denominators of the learning rates are based on the accumulation of the square of the past gradients. To address this problem, some variants of Adagrad have been proposed, such as RMSprop [29], Adam [22] and AMSgrad [30]. These methods use exponential moving average to estimate the moments of the gradients, which can mitigate the rapid decay of the learning rates. Among these variants of Adagrad, Adam is the most widely used method due to its fast convergence and easiness in implement. Furthermore, Adam has been successfully used in many real-world applications like computational biology [31], automated driving [32], text categorization [33], machine translation [34], etc.

AUC maximization

AUC has been widely used to evaluate the classification performance. Therefore, several algorithms have been proposed to maximize AUC with different convex surrogate losses [10, 16, 17, 19, 35]. Initially, many efforts have been devoted to optimize AUC in batch learning setting [10, 35]. However, those batch algorithms fail to meet the demands of efficient and scalable learning for large-scale tasks. Therefore, some online AUC maximization algorithms have been proposed [16, 17, 19]. Generally, there are two main online learning frameworks for AUC maximization. The first framework stores several fixed-size buffers and adopts the reservoir sampling technique to update the buffers for representing the received instances. The sizes of these buffers are related to the number of training instances that makes this type of algorithms impractical for large-scale applications. Besides, this framework use hinge losses as the surrogate loss, which has been proven to be inconsistent with AUC [36]. To overcome these limitations, the work [17] proposed a new framework called OPAUC, which applies the square loss as the surrogate loss function in one-pass learning setting. OPAUC utilizes the consistency between the square loss and the AUC score, and only maintains the mean vector and variance matrix of the received instances. Compared to the first framework, the storage requirement of OPAUC is independent of the number of training instances and each instance only requires to go through only once. But the above two frameworks are both based on the OGD for optimization, which prevents them from exploiting the geometrical information of data [20]. For performing more informative gradient-based learning, the work [16] proposed an algorithm AdaOAM by applying one-pass framework and Adagrad [20].
Although AdaOAM has achieved fairly good performance, it may not fully converge according to the fact that the denominators of its learning rates are the accumulation of all previous gradients. The learning rates of AdaOAM would shrink fast with the rapid increase of the denominators, and this degrades its learning performance. To solve this problem, we develop a novel adaptive online AUC maximization algorithm called AdmOAM, which uses the square loss function in one-pass framework and applies Adam [22] for mitigating the rapid decay of the learning rates.

Method
In this section, we present the framework of AdmOAM. We first introduce the problem setting of the online AUC maximization tasks. Then we present the details of AdmOAM.

Problem setting
We concentrate on learning a linear model \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) in binary classification setting. We denote \( X = \mathbb{R}^d \) and \( Y = \{+1, -1\} \) as the feature space and label space, respectively. Let \( D \) denotes an unknown distribution over \( X \times Y \). Let \( S \) denotes a sample that is drawn i.i.d from \( D \). Let \( H \) denotes the hypothesis class. At the \( t \)-th iteration, we denote the received training instance as \( (x_t, y_t) \in S \), and \( w_t \in H \) is the linear model we learned currently. Let \( S_+ = \{(x^+_i, +1) | i \in [n_+]\} \) be the set of positive instances and \( S_- = \{(x^-_i, -1) | i \in [n_-]\} \) be the set of negative instances in sample \( S \), where \( n_+ \) and \( n_- \) refer to the numbers of positive and negative instances, respectively. Then the AUC score of the linear function \( f \) on sample \( S \) can be calculated as:

\[
AUC(f) = \frac{\sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \left[ I[f(x^+_i) > f(x^-_j)] + \frac{1}{2} I[f(x^+_i) = f(x^-_j)] \right]}{n_+ n_-} \quad (1)
\]

where \( w \) is the weight vector of function \( f \) and \( I[\cdot] \) is the indicator function which outputs 1 if condition is satisfied and 0 otherwise. Practically, we use the least square loss \( \ell(w^T (x^+_i - x^-_j)) = (1 - w^T (x^+_i - x^-_j))^2 \) as a surrogate of the indicator function. The square loss function is convex and keeps consistent with AUC [17]. Then we can minimize the following objective function for finding the optimal linear classifier.

\[
\mathcal{L}(w) = \frac{\lambda}{2} \| w \|^2 + \frac{\sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \ell(w^T (x^+_i - x^-_j))}{2n_+ n_-} \quad (2)
\]

where \( \frac{\lambda}{2} \| w \|^2 \) is introduced as the regularizer to reduce the complexity of the linear classifier. Next we present the details of AdmOAM.
Adaptive moment online AUC maximization

In online learning framework, we focus on minimizing the regret of a sequence of algorithms with regard to a competing hypothesis, where the model of the competing hypothesis is the optimal decision in hindsight. The optimal decision $w^*$ is defined as:

$$ w^* = \arg \min_{w \in \chi} \sum_{t=1}^{T} f_t(w), $$

where $\chi$ is the decision set.

The regret of hypothesis $\mathcal{H}$ at iteration $T \in \mathbb{N}$ is defined as:

$$ \text{Regret}_T(\mathcal{H}) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(w^*). $$

where $w_t, w^* \in \mathcal{H}$. According to the approach in [17], the overall loss $L(w)$ can be transformed to a sum of losses on each training instance in online setting

$$ L(w) = \sum_{i=1}^{T} L_i(w) \quad \text{where} $$

$$ L_i(w) = \frac{\lambda}{2} ||w||^2 + \frac{\sum_{t=1}^{T} S_i (y \neq y_t)}{2\{i \in [t-1] : y_{y_t} = -1\}} (1 - y_t w^T (x_t - x_i))^2 $$

where $S_i = \{(x, y) | i \in [t]\}$ denotes the i.i.d training sample on the $t$-th iteration, and $L_i(w)$ is an unbiased estimation of $L(w)$. Then the gradient of $L_i(w)$ can be calculated as:

$$ \nabla L_i(w) = \begin{cases} 
\lambda w + x x_i^T w - x_i + \frac{\sum_{y \neq y_t} [x_t + (x x_i^T - x x_i^T - x x_i^T) w]}{T_i} & \text{if} \quad y_i = +1 \\
\lambda w + x x_i^T w + x_i + \frac{\sum_{y \neq y_t} [-x_t + (x x_i^T - x x_i^T - x x_i^T) w]}{T_i} & \text{if} \quad y_i = -1 
\end{cases} $$

where $T_i^+$ and $T_i^-$ denote the numbers of positive and negative instances in $S_i$, respectively.

For calculating $\nabla L_i(w)$ without storing all received instances, we use $c^\pm$ and $\Gamma_i^\pm$ as the mean vectors and covariance matrices of positive and negative instances, respectively.

$$ c^\pm_t = \frac{1}{T_i^\pm} \sum_{i \in [T_i^\pm]} \frac{x_i}{x_i} \quad \text{and} \quad \Gamma_i^\pm = \sum_{i \in [T_i^\pm]} \frac{x_i x_i^T - c_i^\pm [c_i^\pm]^T}{T_i^\pm} $$

The mean vectors and covariance matrices can be updated as follows:

$$ c_i^\pm = c_{i-1}^\pm + \frac{1}{T_i^\pm} (x_i^\pm - c_i^\pm) $$

$$ \Gamma_i^\pm = \Gamma_{i-1}^\pm + c_{i-1}^\pm [c_{i-1}^\pm]^T - c_i^\pm [c_i^\pm]^T + \frac{x_i^\pm (x_i^\pm)^T - \Gamma_{i-1}^\pm - c_{i-1}^\pm [c_{i-1}^\pm]^T}{T_i^\pm}. $$

An Adaptive Moment estimation method for online AUC maximization
Therefore, the gradient $\nabla L_t(w)$ can be reformulated as:

$$
\nabla L_t(w) = \begin{cases} 
\lambda w - x_t + c^-_t + (x_t - c^+_t)(x_t - c^-_t)^T + \Gamma^-_t w & \text{if } y_t = +1 \\
\lambda w + x_t - c^+_t + (x_t - c^-_t)(x_t - c^+_t)^T + \Gamma^+_t w & \text{if } y_t = -1
\end{cases}
$$

(9)

Then we can update the linear classifier by using online gradient descent $w_{t+1} = w_t - \eta_t g_t$, where $\eta_t$ is the learning rate at iteration $t$ and $g_t = \nabla L_t(w)$. According to the properties of strong convexity in [16], the optimal $w$ is satisfied with $\|w\|_2 \leq 1/\sqrt{\lambda}$. As a result, it is reasonable to restrict $w_t$ with $\|w_{t+1}\|_2 \leq 1/\sqrt{\lambda}$ by applying the projected gradient method [28].

$$
w_{t+1} = \Pi_{\frac{1}{\sqrt{\lambda}}} (w_t - \eta_t g_t) = \arg\min_{\|w\|_2 \leq 1/\sqrt{\lambda}} \|w - (w_t - \eta_t g_t)\|_2^2
$$

(10)

However, it has been shown that the model cannot fully exploit the geometrical information of data with a global learning rate [20]. For solving this problem, [16] proposed AdaOAM by updating the learning rates of different features as follows:

$$
\eta^{(i)}_t = \frac{\rho}{\epsilon + \sum_{i=1}^{d} s_{i,t}}
$$

(11)

where $i \in [d]$ is the $i$-th dimensional feature and $t$ is the number of iterations.

**Algorithm 1** The AdmOAM Algorithm

**Input:** The regularization parameter $\lambda > 0$, the step size $\{\eta_t\}_{t=1}^T$, 1st moment (the mean) and 2nd raw moment (the uncentered variance), the exponential decay rates $\beta_1, \beta_2 \in [0, 1)$ and a smooth parameter $\epsilon > 0$.

**Output:** Updated classifier $w_T$.

**Variables:** $m_t, v_t \in \mathbb{R}^d$.

**Initialize** $T_0 = 0, w_0 = 0, m_0 = v_0 = 0, c^+_0 = 0, \Gamma_0^+ = [0]_{1 \times d}$.

for $t = 1, 2, \ldots, T$ do

Observe instance $(x_t, y_t)$;

if $y_t = +1$ then

$T^+_t = T^+_{t-1} + 1, T^-_t = T^-_{t-1};$

$c^+_t = c^+_{t-1} + \frac{1}{\beta^+_t}(x_t - c^-_{t-1})$ and $c^-_t = c^-_{t-1};$

Update $\Gamma^+_{t}$ and $\Gamma^-_{t}$;

else

$T^+_t = T^+_{t-1} + 1, T^-_t = T^-_{t-1};$

$c^+_t = c^+_{t-1} + \frac{1}{\beta^+_t}(x_t - c^-_{t-1})$ and $c^-_t = c^-_{t-1};$

Update $\Gamma^+_{t}$ and $\Gamma^-_{t}$;

end if

Calculate gradient $g_t = \nabla L_t(w_{t-1})$;

Update $m_t, v_t$;

$m_t = m_{t-1} / (1 - \beta^+_t), v_t = v_{t-1} / (1 - \beta^-_t);$

$w_t = \Pi_{\frac{1}{\sqrt{\lambda}}}(w_{t-1} - \eta_t \cdot m_t / (\sqrt{v_t} + \epsilon));$

end for

When the value of the accumulation of previous gradients increases too fast, the learning rate of AdaOAM would shrink to a much small value, which can result in a slow convergence. Therefore, we propose AdmOAM for alleviating the rapid decay of learning rates, and our work is inspired by [22]. For the learning rates of different features, AdmOAM adaptively updates them with the estimations of moments of gradients. By using the exponential moving averages of previous gradients as the denominators, AdmOAM mitigates the rapid decay of learning rates.
Specifically, we denote \( \mathbf{m} \) and \( \mathbf{v} \) as the exponential moving averages of the gradients and the squared gradients, respectively. These two vectors are introduced as the estimations of the first and second moments of gradients. They can be updated as follows:

\[
\begin{align*}
\mathbf{m}_t &= \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot \mathbf{g}_t \\
\mathbf{v}_t &= \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot \mathbf{g}_t^2
\end{align*}
\]

where \( \beta_1, \beta_2 \in [0, 1) \) are the exponential decay rates of \( \mathbf{m} \) and \( \mathbf{v} \). Due to the property of exponential moving average, the denominator would not shrink too fast. Besides, AdmOAM only requires extra \( O(d) \) space for storing \( \mathbf{m} \) and \( \mathbf{v} \) as compared to the efficient OPAUC. Note that if \( \mathbf{m} \) and \( \mathbf{v} \) are zero vectors in initialization, then the correction of the bias is needed according to [22]. Therefore, the classifier \( \mathbf{w} \) with the initialization bias correction can be updated as:

\[
\begin{align*}
\bar{\mathbf{m}}_t &= \mathbf{m}_t / (1 - \beta_1^t) \\
\bar{\mathbf{v}}_t &= \mathbf{v}_t / (1 - \beta_2^t) \\
\mathbf{w}_t &= \Pi_\mathbf{w} \left( \mathbf{w}_{t-1} - \eta_t \cdot \bar{\mathbf{m}}_t / (\sqrt{\bar{\mathbf{v}}_t + \epsilon}) \right)
\end{align*}
\]

where \( \epsilon > 0 \) is a smooth parameter for preventing the denominator becoming zero. The framework of AdmOAM is shown in Algorithm 1.

**Theoretical analysis**

Next we present our main theoretical results of AdmOAM.

**Lemma 1.** Let \( \mathbf{w}_t \) and \( \mathbf{g}_t \) \( (t \in [T]) \) be the weight vector and gradient defined in the Algorithm 1, and we have

\[
(g_{t,i})^2 \leq 2(\lambda \mathbf{w}_{t-1,i})^2 + 2(1 + \frac{\sqrt{\sum_{j=1}^d r_{t,i}^2}}{\sqrt{\lambda}})^2 r_{t,i}^2
\]

where we denote the \( i \)-th dimension of the gradient at iteration \( t \) as \( g_{t,i} \) and \( r_{t,i} = max_{j < i} |x_{t,j} - x_{t,i}| \).

**Proof.** Firstly, we define \( \mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathbb{C}^d} \sum_{i=1}^d \mathcal{L}_i(\mathbf{w}) \) as the optimal weight vector of the linear model in hindsight. Objective function \( \mathcal{L}_i(\mathbf{w}) \) uses \( \frac{1}{2} \| \mathbf{w} \|_2^2 \) as the regularizer. According to the strongly convex property, we have \( \| \mathbf{w}^* \|_2^2 \leq 1 / \lambda \) [16]. As a result, we restrict \( \mathbf{w}_t \) with \( \| \mathbf{w}_t \|_2 \leq 1 / \sqrt{\lambda} \) by applying the projected gradient update rule. Besides, according to the definition of the gradient of \( \mathcal{L}_i(\mathbf{w}_{t-1}) \), we have \( \mathbf{g}_t = \nabla \mathcal{L}_i(\mathbf{w}_{t-1}) \). If \( y_t = 1 \), we have

\[
\begin{align*}
(g_{t,i})^2 &= \left[ \lambda \mathbf{w}_{t-1,i} + \frac{\sum_{j:y_{t-1,j} = 1} (1 - y_t (x_t - x_{t-1,j}) y_j (x_{t,j} - x_{t,i}))}{T_i} \right]^2 \\
&= \left[ \lambda \mathbf{w}_{t-1,i} + \frac{\sum_{j:y_{t-1,j} = 1} (x_t - x_{t,j}) (x_{t,j} - x_{t,i})}{T_i} \right]^2
\end{align*}
\]

where \( T_i^- \) denotes the number of the received negative instances at iteration \( t \). By applying
inequality \( \langle w, v \rangle \leq \|w\|_2 \|v\|_2 \) and \((a + b)^2 \leq 2a^2 + 2b^2\), we have

\[
(g_{t,j})^2 \leq \left[ \lambda \|w_{t-1,j}\| + \frac{T_j}{T} \right]^2 \\
\leq \left[ \lambda \|w_{t-1,j}\| + (1 + \sqrt{\frac{\sum_{i=1}^{d} r_{i,j}^2}{\sqrt{\lambda}}} r_{i,j})^2 \\
\leq 2(\lambda \|w_{t-1,j}\|)^2 + 2(1 + \sqrt{\frac{\sum_{i=1}^{d} r_{i,j}^2}{\sqrt{\lambda}}} r_{i,j})^2
\]

This upper bound also holds for \( y_t = -1 \).

**Lemma 2.** Assume the gradient of the objective function \( f_t \) is bounded, \( \sup_{w \in \mathcal{X}} \|g_t(w)\|_2 \leq G \), \( \sup_{w \in \mathcal{X}} \|g_t(w)\|_\infty \leq G_\infty \) and the distance between any elements of the hypothesis class is bounded, \( \sup_{w \in \mathcal{X}} \|w - u\|_2 \leq D \), \( \sup_{w \in \mathcal{X}} \|w - u\|_\infty \leq D_\infty \) and \( \beta_1, \beta_2 \in [0, 1) \) satisfy

\[ \gamma = \frac{\beta_2}{\sqrt{\beta_1}} \leq 1. \]

Let \( \eta_t = \frac{\beta_2}{\sqrt{\beta_1}} \) and \( \beta, \beta = \beta_1 \beta_2^{-1}, \rho \in (0, 1) \). For any \( T > 1 \), the following regret bound holds

\[
R(T) \leq \frac{D^2}{2\eta (1 - \beta_2)} \sum_{i=1}^{d} \sqrt{T \sum_{j=1}^{d} r_{i,j}^2} + \frac{\eta (1 + \beta_1) G_\infty}{(1 - \beta_1) \sqrt{1 - \beta_2 (1 - \gamma)^2}} \sum_{i=1}^{d} \|g_{t,i}\|_2
\]  

\[ + \frac{D'^2 G_\infty \sqrt{1 - \beta_2}}{2 \eta (1 - \beta_2)} \sum_{i=1}^{d} \sqrt{\sum_{j=1}^{d} r_{i,j}^2} \]

where \( g_{1:T, i} = [g_{1,i}, g_{2,i}, \ldots, g_{d,i}] \).

**Theorem 1.** Assume \( \|x\|_\infty \leq 1, \beta_1, \beta_2 \in [0, 1) \) satisfy \( \gamma = \frac{\beta_2}{\sqrt{\beta_1}} \leq 1. \) Let \( \eta_t = \frac{\beta_2}{\sqrt{\beta_1}} \) and \( \beta_1, \beta_2 = \beta_1 \beta_2^{-1}, \rho \in (0, 1) \). For any \( T > 1 \), \( \text{AdmOAM} \) can achieve following regret bound

\[
R(T) \leq A_1 \sum_{i=1}^{d} \sqrt{T \sum_{j=1}^{d} (1 - \beta_2) \beta_2^{-1} B_{i,j}} + A_2 \sum_{i=1}^{d} \sqrt{\sum_{j=1}^{d} B_{i,j}} + A_3
\]

where \( A_1 = \frac{2}{2\eta (1 - \beta_2)} A_2 = \frac{2\eta (1 + \beta_1) \sqrt{\lambda + 4(1 + 2\sqrt{\frac{2}{\sqrt{\lambda}}})^2}}{(1 - \beta_1) \sqrt{\lambda + 4(1 + 2\sqrt{\frac{2}{\sqrt{\lambda}}})^2}}, A_3 = \sum_{i=1}^{d} \frac{2\sqrt{\lambda + 4(1 + 2\sqrt{\frac{2}{\sqrt{\lambda}}})^2}}{\eta (1 - \beta_1) \sqrt{1 - \beta_2 (1 - \gamma)^2}} \]

\[ B_{i,j} = 2(\lambda w_{i,j})^2 + 2(1 + \frac{\sqrt{\sum_{i=1}^{d} r_{i,j}^2}}{\sqrt{\lambda}} r_{i,j})^2 \leq 2(\lambda w_{i,j})^2 + 2(1 + \frac{\sqrt{\sum_{i=1}^{d} r_{i,j}^2}}{\sqrt{\lambda}} r_{i,j})^2 \]

Proof. Since the inequality \( \|w_t\|_2 \leq \frac{1}{\sqrt{\lambda}} \) holds for any \( t \in [T] \), we have \( \sup_{w \in \mathcal{X}} \|w - u\|_2 \leq \frac{\lambda}{\sqrt{\lambda}} \) and \( \sup_{w \in \mathcal{X}} \|w - u\|_\infty \leq \frac{\lambda}{\sqrt{\lambda}} \) according to the triangle inequality and \( \|w\|_\infty \leq \|w\|_2 \).

Using Lemma 1 and assumption \( \|x\|_\infty \leq 1 \), we have \( r_{i,j} \leq \max_{j \in [d]} |x_{i,j} - x_{i,j}| \leq 2 \) and

\[ \sup_{w \in \mathcal{X}} \|g_t(w)\|_2 \leq \max_{i \in [d]} |g_{i,j}| \leq \sqrt{2d \max_{i \in [d]} |g_{i,j}|} \leq \sqrt{2d \lambda + 8d(1 + 2\sqrt{\frac{2}{\sqrt{\lambda}}})^2} \]. Then it is easy to obtain

\[ \sup_{w \in \mathcal{X}} \|g_t(w)\|_2 \leq \sqrt{2d \max_{i \in [d]} |g_{i,j}|} \leq \sqrt{2d \lambda + 8d(1 + 2\sqrt{\frac{2}{\sqrt{\lambda}}})^2} \].
According to the definitions of $\nu$ and $g_{i:T,i}$, we have

\[
\sum_{i=1}^{d} \sqrt{T}\nu_{T,i} = \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (1 - \beta_{2}) \beta_{2}^{T-j} B_{ij} \leq \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (1 - \beta_{2}) \beta_{2}^{T-j} B_{ij} \sqrt{1 - \beta_{2}^{T}} \tag{19}
\]

\[
\sum_{i=1}^{d} \|g_{i:T,i}\|_{2} = \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} g_{ij}^{2} \leq \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} B_{ij} \tag{20}
\]

Plugging (19), (20), the bound of the gradient and the bound of the distance between any weight vectors into (17), we have

\[
R(T) \leq \frac{2}{\lambda \eta (1 - \beta_{1})} \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (1 - \beta_{2}) \beta_{2}^{T-j} B_{ij} \sqrt{1 - \beta_{2}^{T}} + \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} \frac{2 \sqrt{2(1 - \beta_{2})} \sqrt{\lambda + 4(1 + 2 \sqrt{\frac{d}{\lambda}})^{2}}}{\eta (1 - \beta_{1})(1 - \gamma)^{2}} \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} B_{ij} \]

which completes the proof.

If the features are rather dense, then we have

\[
B_{ij} = 2(\lambda w_{ij})^{2} + 2(1 + \frac{\sum_{k=1}^{d} r_{kj}^{2}}{\sqrt{\lambda}}) r_{ij}^{2} \leq 2\lambda + 8(1 + \frac{2\sqrt{d}}{\sqrt{\lambda}})^{2}. \tag{21}
\]

according to the inequality $\max_{x \in [d]} |w_{ij}| \leq \|w\|_{2} \leq 1/\sqrt{\lambda}$ and $r_{ij} \leq 2$ for any $i \in [d]$.

If we denote the constant $C = 2\lambda + 8(1 + \frac{2\sqrt{d}}{\sqrt{\lambda}})^{2}$, it is easy to obtain:

\[
\sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (1 - \beta_{2}) \beta_{2}^{T-j} B_{ij} \leq \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (1 - \beta_{2}) \beta_{2}^{T-j} C \leq \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (\beta_{2}^{T-j} - \beta_{2}^{T-j-1}) C \leq \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (\beta_{2}^{T-j} - \beta_{2}^{T-j-1}) C \leq d \sqrt{T} \sum_{i=1}^{d} \sqrt{T} \sum_{j=1}^{T} (\beta_{2}^{T-j} - \beta_{2}^{T-j-1}) C \leq d \sqrt{T} \sum_{i=1}^{d} C \]

\[
= d \sqrt{T} C \tag{22}
\]
Also we have:

$$\sum_{i=1}^{d} \sqrt{\sum_{j=1}^{T} B_{i,j}} \leq d \sqrt{TC}.$$  \hspace{1cm} (23)

Therefore, AdmOAM’s regret bound is $O(\sqrt{T})$ for the dense feature space and its convergence rate is $O(1/\sqrt{T})$ as in the general case of the non-adaptive algorithms.

When the features are sparse, the term $B_{i,j}$ should be much smaller than $C$. Therefore, the regret bound of AdmOAM should be much smaller than $O(\sqrt{T})$, which results in a faster convergence. Besides, AdmOAM achieves comparable convergence rate with respect to AdaOAM according to [22].

From the above analysis, we can conclude that AdmOAM converges faster than the non-adaptive algorithms and stays in comparable convergence rate as AdmOAM.

**Experimental results**

In this section, we evaluate the performance of AdmOAM on several standard benchmark datasets.

**Compared algorithms**

Since we only concentrate on online scenarios, we do not take existing batch learning methods into consideration. We compare AdmOAM with 4 competing online AUC maximization algorithms:

- **OAM\textsubscript{seq}:** The OAM algorithm using sequential updating [19];
- **OAM\textsubscript{gra}:** The OAM algorithm using online gradient updating [19];
- **OPAUC:** One-pass AUC optimization algorithm [17];
- **AdaOAM:** The adaptive subgradient online AUC optimization algorithm [16].

**Experimental testbed and setup**

We conduct the experiments on 13 benchmark datasets, which can be downloaded from the LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/) and the UCI websites (http://www.ics.uci.edu/~mlearn/MLRepository.html). Note that the glass, vehicle, dna and acoustic are multi-class datasets, we transform them into binary-class by randomly setting one class as the positive class, and the others as negative. The sparsity of the dataset is defined as the number of zero elements divided by the total number of elements in its feature matrix. Besides, the features have been rescaled to $[-1, 1]$. The details of the datasets are summarized in Table 1.

We conduct nested cross-validation for hyperparameter searching and model evaluation. In the outer cross-validation, we conduct 5-fold cross-validation on each benchmark dataset, where 4 folds are for training and the remaining fold is treated as the test set. In the inner cross-validation, we apply 5-fold cross-validation on the training set for hyperparameter searching. After the process of the inner cross-validation, we train the model on the whole training set with the tuned hyperparameters. Finally, we calculate the AUC score of the model on the test set for model evaluation. For further reducing the variance in the results, we apply 4 independent 5-fold nested cross-validation on each dataset. Therefore the AUC performance of each algorithm on different datasets is the average over 20 independent runs. For the
hyperparameter searching, we tune the learning rate $\eta \in 2^{-10:1:3}$ and the regularization parameter $\lambda \in 2^{-10:1:3}$ for AdmOAM, AdaOAM and OPAUC. For the exponential decay rates of AdmOAM, we decide $\beta_1 \in [0.1: 0.1: 0.9]$ and $\beta_2 \in [0.099: 0.1: 0.999]$. For buffer sampling algorithms like OAM$_{seq}$ and OAM$_{gra}$, we tune the penalty parameter $C \in 2^{-10:1:10}$, and the size of the buffer is set at 100 as recommended in [19].

### Evaluation on benchmark datasets

In this subsection, we analyse the average AUC values, convergence rate and running time of AdmOAM with compared methods. Table 2 shows the average AUC values over 20 independent runs on 13 benchmark datasets.

Based on the results in Table 2, we have several observations. Firstly, the two adaptive methods AdmOAM and AdaOAM achieve higher AUC score than the other three non-adaptive methods in most cases. Therefore, the adaptive learning strategy can effectively improve the

| Datasets     | instances | features | $T_+/T_-$ | sparsity(%) |
|-------------|-----------|----------|-----------|-------------|
| glass       | 214       | 9        | 2.0571    | 0.1558      |
| svmguide4   | 300       | 10       | 5.8182    | 0.0333      |
| heart       | 690       | 14       | 1.2500    | 3.7607      |
| australian  | 690       | 14       | 1.2476    | 12.5569     |
| vehicle     | 846       | 18       | 2.9906    | 1.9766      |
| german      | 1,000     | 24       | 2.3333    | 4.1625      |
| svmguide3   | 1,243     | 22       | 3.1993    | 0.3730      |
| w1a         | 2,477     | 300      | 33.4027   | 96.1768     |
| dna         | 3,186     | 180      | 1.0796    | 74.7329     |
| a9a         | 32,561    | 123      | 3.1526    | 88.7243     |
| cod-rna     | 59,535    | 8        | 2.0000    | 0.0015      |
| acoustic    | 78,823    | 50       | 3.3165    | 0           |
| ijcnn1      | 141,691   | 22       | 9.4453    | 0           |

https://doi.org/10.1371/journal.pone.0215426.t001

| Datasets     | AdmOAM    | AdaOAM    | OPAUC     | OAM$_{seq}$ | OAM$_{gra}$ |
|-------------|-----------|-----------|-----------|-------------|-------------|
| glass       | .8253±.0554 | .8144±.0500 | .7966±.0722 | .8062±.0731 | .7407±.0755 |
| svmguide4   | .9491±.0245 | .8429±.0434 | .8006±.0657 | .9040±.0386 | .9091±.0569 |
| heart       | .9111±.0299 | .9083±.0315 | .9073±.0586 | .8024±.0767 | .8678±.0586 |
| australian  | .9304±.0218 | .9285±.0206 | .9269±.0210 | .8683±.0278 | .8252±.0643 |
| vehicle     | .8316±.0240 | .7927±.0184 | .7854±.0285 | .7170±.0572 | .7469±.0634 |
| german      | .8003±.0216 | .7990±.0227 | .7963±.0249 | .7544±.0300 | .7675±.0320 |
| svmguide3   | .7527±.0382 | .7434±.0343 | .7315±.0399 | .7003±.0541 | .6900±.0460 |
| w1a         | .9439±.0222 | .9317±.0355 | .9184±.0391 | .8891±.0570 | .8978±.0528 |
| dna         | .9840±.0033 | .9826±.0031 | .9832±.0032 | .9622±.0050 | .9620±.0052 |
| a9a         | .9003±.0039 | .9002±.0039 | .8998±.0039 | .8459±.0138 | .8469±.0150 |
| cod-rna     | .9762±.0011 | .9740±.0012 | .9701±.0012 | .8578±.0662 | .8256±.0607 |
| acoustic    | .8933±.0016 | .8907±.0019 | .8901±.0018 | .8389±.0168 | .8408±.0165 |
| ijcnn1      | .9094±.0027 | .8976±.0028 | .8471±.0025 | .8441±.0419 | .8766±.0396 |

https://doi.org/10.1371/journal.pone.0215426.t002
Secondly, AdmOAM obtains better or comparable performance than AdaOAM for most datasets. Especially in the svmguide4 and vehicle datasets, AdmOAM achieves much higher AUC scores than AdaOAM. This indicates the effectiveness of AdmOAM over AdaOAM.

Next we provide the analysis on the speed of the convergence of AdmOAM. For each online AUC maximization algorithm, it updates the model from a sequence of training data one at a time. For comparing the convergence speed, we evaluate the AUC score of different online learning algorithms on the testing set. Compared with reservoir sampling methods like OAM$^{seq}$ and OAM$^{gra}$, the algorithms based on the one-pass learning mode obtain better performance according to the results in Table 2. Therefore, we compare the convergence rate of AdmOAM with AdaOAM and OPAUC. Fig 1(a)–1(d) depict the convergence curves on 4 benchmark datasets with the error bars. Specifically, we report the average AUC score across 20 independent runs on the testing datasets at different iterations. From Fig 1, we can observe that AdmOAM converges faster than the other two algorithms. With the increasing of the number of iterations, AdmOAM achieves a higher AUC score than AdaOAM and OPAUC. This validates our theoretical analysis and demonstrates the effectiveness of AdmOAM.
We also present the running time on 13 datasets in Fig 2. On most datasets, AdmOAM is more efficient than OAM\textsubscript{seq} and OAM\textsubscript{gra}, and stays competitive with AdaOAM in the computational complexity. Compared to OPAUC, AdmOAM needs to spend a little more time for updating two extra vectors of the first and second moments.

**Evaluation of parameter sensitivity**

Since AdmOAM adaptively updates the learning rates, we mainly focus on the parameter sensitivity of learning rate and the other parameters are fixed at the tuned values. We report the average test AUC score across 5 independent runs (1 trail of 5-fold cross-validation) with the range of learning rates $\eta \in 2^{-10:1:3}$ in Fig 3. Based on the results in Fig 3, we can observe that AdmOAM is less sensitive to the learning rate than OPAUC especially when the value of the learning rate is over $2^{-2}$. In [16], the author claimed that AdaOAM is insensitive to the parameter settings. From Fig 3, we can observe that AdmOAM obtains comparable or better average AUC score than AdaOAM. The above results indicate that AdmOAM can effectively adjust its per-coordinate learning rate and is less sensitive to the parameter settings.

**Conclusion and future work**

In this paper we proposed AdmOAM, an Adaptive Moment estimation method for Online AUC Maximization. It applies the estimation of moments of gradients to accelerate the convergence and mitigate the rapid decay of the learning rates. Theoretically, we have analysed the regret bound of the proposed algorithm. It can achieve a lower bound than non-adaptive online AUC maximization algorithms and stay competitive to AdaOAM. Moreover, we evaluated its performance with several competing algorithms on benchmark datasets. The experimental results validate the theoretical analysis and indicate the effectiveness of the proposed algorithm.

For future work, there are several research directions. Firstly, AdmOAM uses all features for AUC maximization. This is not efficient and scalable for high-dimensional sparse datasets. It would be interesting to combine AdmOAM with feature selection techniques for learning a sparse model. Secondly, AdmOAM is not suitable for the non-linearly separable data with...
linear model. It would be interesting to combine AdmOAM with online kernel learning methods for handling the nonlinearity of the data.

**Acknowledgments**

Our work is supported by the National Key Research Development Program of China (No. 2017YFB0802800). The authors would like to thank the Editor-in-Chief and anonymous reviewers for their insightful and constructive commendations that have led to an improved version of this paper.

**Author Contributions**

**Conceptualization:** Xin Liu, Xingyu Zhou.

**Formal analysis:** Xin Liu.

**Methodology:** Xin Liu.

**Writing – original draft:** Xin Liu.
Writing – review & editing: Xin Liu, Zhisong Pan, Haimin Yang, Xingyu Zhou, Wei Bai, Xianghua Niu.

References
1. Hanley JA, McNeil BJ. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology. 1982; 143(1):29–36.
2. Brefeld U, Scheffer T. AUC maximizing support vector learning. In: Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning; 2005.
3. Maloof MA. Learning when data sets are imbalanced and when costs are unequal and unknown. In: Proceedings of the ICML Workshop on Learning from Imbalanced Data Sets II. vol. 2; 2003. p. 2–1.
4. Ling CX, Huang J, Zhang H. AUC: a better measure than accuracy in comparing learning algorithms. In: Conference of the Canadian Society for Computational Studies of Intelligence. Springer; 2003. p. 329–341.
5. Yan L, Dodier RH, Mozer M, Wolniewicz RH. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In: ICML; 2003. p. 848–855.
6. Huang J, Ling CX. Using AUC and accuracy in evaluating learning algorithms. IEEE Transactions on Knowledge and Data Engineering. 2005; 17(3):299–310. https://doi.org/10.1109/TKDE.2005.50
7. Abe N, Zadrozny B, Langford J. Outlier detection by active learning. In: KDD; ACM; 2006. p. 504–509.
8. Statnikov A, Wang L, Aliferis CF. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC Bioinformatics. 2008; 9(1):319. https://doi.org/10.1186/1471-2105-9-319 PMID: 18647401
9. Cortes C, Mohri M. AUC optimization vs. error rate minimization. In: NIPS; 2004. p. 313–320.
10. Joachims T. A support vector method for multivariate performance measures. In: ICML; ACM; 2005. p. 377–384.
11. Kotlowski W, Dembczynski KJ, Huellermeier E. Bipartite ranking through minimization of univariate loss. In: ICML; 2011. p. 1113–1120.
12. Rakotomamonjy A. Optimizing area under ROC curve with SVMs. In: ROCAI; 2004. p. 71–80.
13. Bottou L, Cun YL. Large scale online learning. In: NIPS; 2004. p. 217–224.
14. Cesa-Bianchi N, Conconi A, Gentile C. On the generalization ability of on-line learning algorithms. IEEE Transactions on Information Theory. 2004; 50(9):2050–2057. https://doi.org/10.1109/TIT.2004.833339
15. Rakhlin A, Shamir O, Sridharan K. Making gradient descent optimal for strongly convex stochastic optimization. In: ICML; 2012. p. 449–456.
16. Ding Y, Zhao P, Hoi SC, Ong YS. An adaptive gradient method for online AUC maximization. In: AAM; 2015. p. 2568–2574.
17. Gao W, Jin R, Zhu S, Zhou ZH. One-pass AUC optimization. In: ICMIA; 2013. p. 906–914.
18. Kar P, Sriperumbudur B, Jain P, Karnick H. On the generalization ability of online learning algorithms for pairwise loss functions. In: ICML; 2013. p. 441–449.
19. Zhao P, Hoi SC, Jin R, Yang T. Online AUC maximization. In: ICMIA; 2011. p. 233–240.
20. Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research. 2011; 12(Jul):2121–2159.
21. Lu Y, Lund J, Boyd-Graber J. Why ADAGRAD fails for online topic modeling. In: EMNLP; 2017. p. 446–451.
22. Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.
23. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review. 1958; 65(6):386. https://doi.org/10.1037/h0042519 PMID: 13602029
24. Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y. Online passive-aggressive algorithms. Journal of Machine Learning Research. 2006; 7(Mar):551–585.
25. Dredze M, Crammer K, Pereira F. Confidence-weighted linear classification. In: ICML; ACM; 2008. p. 264–271.
26. Crammer K, Kulesza A, Dredze M. Adaptive regularization of weight vectors. In: NIPS; 2009. p. 414–422.
27. Duchi JC, Shalev-Shwartz S, Singer Y, Tewari A. Composite objective mirror descent. In: COLT; 2010. p. 14–26.
28. Zinkevich M. Online convex programming and generalized infinitesimal gradient ascent. In: ICML; 2003. p. 928–936.
29. Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. 2012; 4(2):26–31.
30. Reddi SJ, Kale S, Kumar S. On the convergence of Adam and beyond. ICLR. 2018.
31. Angermuller C, Parmamaa T, Parts L, Stegle O. Deep learning for computational biology. Molecular Systems Biology. 2016; 12(7):878–878. https://doi.org/10.15252/msb.20156651 PMID: 27474269
32. Sallab A, Gamal M. End-To-End multi-modal sensors fusion system for urban automated driving. In: Proceedings of the NIPS Workshop on Machine Learning for Intelligent Transportation Systems; 2018.
33. Chen G, Ye D, Xing Z, Chen J, Cambria E. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. International Symposium on Neural Networks. 2017; p. 2377–2383.
34. Lample G, Ott M, Conneau A, Denoyer L, Ranzato M. Phrase-Based neural unsupervised machine translation. EMNLP. 2016; p. 5039–5049.
35. Herschtal A, Raskutti B. Optimising area under the ROC curve using gradient descent. In: ICML. ACM; 2004. p. 49–57.
36. Gao W, Zhou ZH. On the consistency of AUC pairwise optimization. In: IJCAI; 2015. p. 939–945.