Online AUC Maximization

Peilin Zhao ZHAO0106@NTU.EDU.SG
Steven C.H. Hoi CHHOI@NTU.EDU.SG
School of Computer Engineering, Nanyang Technological University, Singapore 639798

Rong Jin RONGJIN@CSE.MSU.EDU
Tianbao Yang YANGTIA1@CSE.MSU.EDU
Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48824, USA

Abstract

Most studies of online learning measure the performance of a learner by classification accuracy, which is inappropriate for applications where the data are unevenly distributed among different classes. We address this limitation by developing online learning algorithm for maximizing Area Under the ROC curve (AUC), a metric that is widely used for measuring the classification performance for imbalanced data distributions. The key challenge of online AUC maximization is that it needs to optimize the pairwise loss between two instances from different classes. This is in contrast to the classical setup of online learning where the overall loss is a sum of losses over individual training examples. We address this challenge by exploiting the reservoir sampling technique, and present two algorithms for online AUC maximization with theoretic performance guarantee. Extensive experimental studies confirm the effectiveness and the efficiency of the proposed algorithms for maximizing AUC.

1. Introduction

Online learning has been actively studied in machine learning community (Cesa-Bianchi & Lugosi, 2006), due to its high efficiency to large datasets. Despite the extensive investigation, most studies of online learning measure the performance by either *mistake rate* or *prediction accuracy*. However, these metrics are not appropriate for applications where data are class-imbalanced, as argued and demonstrated in a number of studies (Elkan, 2001; Cortes & Mohri, 2003). To address this challenge, researchers have proposed more meaningful metrics, such as the Receiver Operating Characteristics (ROC) curve (Hanley & McNeil, 1982) and the Area Under the ROC curve (AUC) (Hanley & McNeil, 1982). The ROC curve details the rate of true positives against false positives over the range of possible threshold values. AUC is a decision threshold independent metric that measures the probability for a randomly drawn positive instance to have a higher decision value than a randomly sampled negative instance. Substantial efforts have been devoted to exploring the ROC and AUC metrics for batch machine learning tasks (Bradley, 1997; Rakotomamonjy, 2004; Herschtal & Raskutti, 2004; Brefeld & Scheffer, 2005). However, to the best of our knowledge, no algorithm has been proposed to optimize the AUC metric in an online learning setting.

In this work, we investigate online learning algorithms for maximizing the AUC metric, referred to as **Online AUC Maximization** or **OAM** for short. The key challenge for online AUC maximization is that AUC is written as a sum of pairwise losses between instances from different classes, which is quadratic in the number of received training examples. In contrast, most online learning studies assume the overall loss is a linear combination of losses experienced by individual training examples. Directly applying classical online learning algorithms to maximize AUC requires memorizing all the received training examples, making it unattractive for large-scale applications. In this paper, we propose to overcome this challenge by exploring the reservoir sampling technique (Vitter, 1985), which allows us to represent all the received training examples by buffers of fixed size. We develop online learning algorithms for OAM based on the idea of reservoir sampling, and present theoretical analysis that bounds the difference in AUC between the solution computed by online learning and the optimal solution learned at hindsight. Extensive experiments confirm the effectiveness and the efficiency of the proposed OAM algorithms.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

The rest of the paper is organized as follows. Section 2 reviews the related work of online learning and existing works for maximizing AUC. Section 3 presents the problem of online AUC maximization and the proposed algorithms. Section 4 discusses our empirical evaluations and Section 5 concludes this work.

2. Related Work

Our work is closely related to the studies of online learning and machine learning with imbalanced data or known as cost-sensitive learning.

Online Learning. Many algorithms have been proposed for online learning. The most well-known method is the Perceptron algorithm (Rosenblatt, 1958; Freund & Schapire, 1999). Many modern online learning algorithms (Crammer & Singer, 2003; Gentile, 2001; Crammer et al., 2006) are inspired by the maximum margin principle that has been successfully applied to batch mode learning. Many recent studies explore the connection between online learning and optimization theory (Dredze et al., 2008; Crammer et al., 2008). Despite the extensive investigation, most studies of online learning assume the overall loss is a sum of losses experienced by individual training examples. In contrast, we consider an online learning problem where a loss function is quadratic in the number of training examples, a significantly more challenging problem than the conventional setup of online learning.

Cost-sensitive learning. Cost-sensitive learning has been studied extensively in literature (Elkan, 2001; Li et al., 2002; Crammer et al., 2006). Several algorithms are developed to train classifiers by maximizing AUC. Two well-known algorithms are optimizing the Wilcoxon-Mann-Whitney statistic (Yan et al., 2003) and RankOpt (Herschtal & Raskutti, 2004) that adopts a differentiable approximation of AUC as its objective function. Several studies extend SVM to optimize the AUC metrics (Rakotomamonjy, 2004; Brefeld & Scheffer, 2005). In (Joachims, 2005), the authors present a general framework for optimizing multivariate nonlinear performance measures, including AUC and $F1$.

Despite the extensive studies in batch cost-sensitive learning, few work considers cost-sensitive online learning, except for the study (Crammer et al., 2006). Although it proposes some simple solution for cost-sensitive online learning, it does not directly optimize the AUC metric. To the best of our knowledge, our work is the first online learning study that aims to optimize the AUC metric directly.

3. Online AUC Maximization (OAM)

3.1. Problem Definition

We focus on learning a linear classification model for a binary classification problem with imbalanced data distributions for the two classes. Without loss of generality, we assume positive class to be the rare class. Let us denote by $(\mathbf{x}_t, y_t)$ the training example received at the t-th trial, where $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \{-1, +1\}$, and by $\mathbf{w}_t \in \mathbb{R}^n$ the weight vector learned so far.

We define the AUC measure (Hanley & McNeil, 1982) for binary classification. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{-1,+1\}|\ i \in [T]\}$, we divide it into two sets: the set of positive instances $\mathcal{D}_+ = \{(\mathbf{x}_i^+, +1)|\ i \in [T_+]\}$ and the set of negative instances $\mathcal{D}_- = \{(\mathbf{x}_j^-, -1)|\ j \in [T_-]\}$, where T_+ and T_- are the numbers of positive and negative instances, respectively. For a linear classifier $\mathbf{w} \in \mathbb{R}^d$, its AUC measure based on the dataset $\mathcal{D}$ is defined as follows:

$$\begin{aligned}\mathrm{AUC}(\mathbf{w}) &= \frac{\sum_{i=1}^{T_+}\sum_{j=1}^{T_-} \mathbb{I}_{(\mathbf{w}\cdot\mathbf{x}_i^+ > \mathbf{w}\cdot\mathbf{x}_j^-)}}{T_+T_-} \\ &= 1 - \frac{\sum_{i=1}^{T_+}\sum_{j=1}^{T_-} \mathbb{I}_{(\mathbf{w}\cdot\mathbf{x}_i^+ \le \mathbf{w}\cdot\mathbf{x}_j^-)}}{T_+T_-}\end{aligned}$$

where $\mathbb{I}_\pi$ is the indicator function that outputs 1 if the prediction π holds and 0 otherwise. Thus, maximizing $\mathrm{AUC}(\mathbf{w})$ is equivalent to minimizing $\sum_{i=1}^{T_+}\sum_{j=1}^{T_-} \mathbb{I}_{(\mathbf{w}\cdot\mathbf{x}_i^+ - \mathbf{w}\cdot\mathbf{x}_j^- \le 0)}$. We replace the indicator function with its convex surrogate, i.e., the hinge loss function

$$\ell(\mathbf{w}, \mathbf{x}_i^+ - \mathbf{x}_j^-) = \max\{0, 1 - \mathbf{w}\cdot(\mathbf{x}_i^+ - \mathbf{x}_j^-)\},$$

and find the optimal classifier by minimizing the following objective

$$\frac{1}{2}|\mathbf{w}|_2^2 + C\sum_{i=1}^{T_+}\sum_{j=1}^{T_-}\ell(\mathbf{w}, \mathbf{x}_i^+ - \mathbf{x}_j^-) \tag{1}$$

where $|\mathbf{w}|_2^2/2$ is introduced to regularize the complexity of the linear classifier, and C is a positive penalty parameter of the error term.

3.2. Online Learning Algorithm for AUC Maximization

Our goal is to develop an online learning algorithm to efficiently optimize (1). The key challenge arises from the fact that the overall loss is a sum of losses over *pairwise instances*, quadratic in the number of training examples, making it a significantly more challenging problem than conventional online learning problems.

To address this challenge, we rewrite (1) into a sum of losses for individual instances, i.e.,

$$\frac{1}{2}|\mathbf{w}|_2^2 + C\sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}) \tag{2}$$

where $\mathcal{L}_t(\mathbf{w})$ is defined as

$$\mathcal{L}_t(\mathbf{w}) = \mathbb{I}_{(y_t=+1)}h_+^t(\mathbf{w}) + \mathbb{I}_{(y_t=-1)}h_-^t(\mathbf{w}) \tag{3}$$

In the above, $h_\pm^t(\mathbf{w})$ are defined as:

$$h_+^t(\mathbf{w}) = \sum_{t'=1}^{t-1}\mathbb{I}_{(y_{t'}=-1)}\ell(\mathbf{w}, \mathbf{x}_t - \mathbf{x}_{t'}) \tag{4}$$

$$h_-^t(\mathbf{w}) = \sum_{t'=1}^{t-1}\mathbb{I}_{(y_{t'}=+1)}\ell(\mathbf{w}, \mathbf{x}_{t'} - \mathbf{x}_t) \tag{5}$$

Using this representation, we can directly apply the gradient descent based online learning algorithm (Zinkevich, 2003) and update $\mathbf{w}_t$ by $\mathbf{w}_{t+1} = \mathbf{w}_t - C\nabla\mathcal{L}_t(\mathbf{w})$, where C is the stepsize parameter. The main problem with this approach is that to compute the gradients $\nabla\mathcal{L}_t(\mathbf{w})$, we have to store all the received training examples, making it impractical for large-scale online learning tasks. We address this challenge by caching a small number of received training examples. To this end, we introduce two buffers, B_+ and B_- of size N_+ and N_-, for storing the received positive and negative instances, respectively. For example $(\mathbf{x}_t, y_t)$ received at trial t, we first update the two buffers, and then update the linear classifier $\mathbf{w}_t$ by comparing $\mathbf{x}_t$ to instances in B_+^t if $y_t = -1$ and to instances in B_-^t if $y_t = +1$. Algorithm 1 outlines the overall framework of our approach.

There are two key routines in Algorithm 1, i.e., **UpdateBuffer** and **UpdateClassifier**. Below we discuss efficient implementations and the theoretic guarantees of those implementations.

3.2.1. Update Buffer

The key challenge for buffer updating is to maintain an accurate "sketch" of history under the constraint of fixed buffer size. To this end, we deploy the "reservoir sampling" technique (Vitter, 1985), which is widely used in data streaming community. Specifically, given a received training example $(\mathbf{x}_t, y_t)$, we will add it to the buffer $B_{y_t}^t$ if $|B_{y_t}^t| < N_{y_t}$. Otherwise, with probability $\frac{N_{y_t}}{N_{y_t}^{t+1}}$, we update the buffer $B_{y_t}^t$ by randomly replacing one instance in $B_{y_t}^t$ with $\mathbf{x}_t$.

The key property of reservoir sampling is that the instances in the buffers simulate a uniform sampling from the original dataset. Algorithm 2 outlines the key steps of the UpdateBuffer routine. The following lemma directly follows the property of reservoir sampling. Through the paper, we use E[·] to denote the expectation over the randomly sampled instances in buffers.

Algorithm 1 A Framework for Online AUC Maximization (**OAM**)

```
Input: the penalty parameter $C$, the maximum buffer size $N_+$ and $N_-$
Initialize $\mathbf{w}_1 = \mathbf{0}$, $B_+^1 = B_-^1 = \emptyset$, $N_+^1 = N_-^1 = 0$
for $t = 1, 2, \ldots, T$ do
  Receive a training instance $(\mathbf{x}_t, y_t)$
  if $y_t = +1$ then
    $N_+^{t+1} = N_+^t + 1$, $N_-^{t+1} = N_-^t$, $B_-^{t+1} = B_-^t$, $C_t = C\max(1, N_-^t/N_-)$
    $B_+^{t+1} = $ UpdateBuffer$(B_+^t, \mathbf{x}_t, N_+, N_+^{t+1})$
    $\mathbf{w}_{t+1} = $ UpdateClassifier$(\mathbf{w}_t, \mathbf{x}_t, y_t, C_t, B_-^{t+1})$
  else
    $N_-^{t+1} = N_-^t + 1$, $N_+^{t+1} = N_+^t$, $B_+^{t+1} = B_+^t$, $C_t = C\max(1, N_+^t/N_+)$
    $B_-^{t+1} = $ UpdateBuffer$(B_-^t, \mathbf{x}_t, N_-, N_-^{t+1})$
    $\mathbf{w}_{t+1} = $ UpdateClassifier$(\mathbf{w}_t, \mathbf{x}_t, y_t, C_t, B_+^{t+1})$
  end if
end for
```

Lemma 1. *For any function $f : \mathbb{R}^d \mapsto \mathbb{R}$ and at any iteration t, we have*

$$\frac{1}{|B_+^t|}\mathrm{E}\left[\sum_{\mathbf{x}\in B_+^t} f(\mathbf{x})\right] = \frac{1}{N_+^t}\sum_{i=1}^{t}\mathbb{I}_{(y_i=+1)}f(\mathbf{x}_i)$$

$$\frac{1}{|B_-^t|}\mathrm{E}\left[\sum_{\mathbf{x}\in B_-^t} f(\mathbf{x})\right] = \frac{1}{N_-^t}\sum_{i=1}^{t}\mathbb{I}_{(y_i=-1)}f(\mathbf{x}_i)$$

3.2.2. Update Classifier

This routine takes five input arguments: the current classifier $\mathbf{w}_t$, training example $(\mathbf{x}_t, y_t)$, buffer B and a weight C_t which plays similar role as step size. We present two strategies for updating the classifier $\mathbf{w}_t$.

Sequential Updating The first approach is to treat $\{(\mathbf{x}_t, \mathbf{x}), \mathbf{x} \in B\}$ as a sequence of pairwise instances, and apply an online learning algorithm to update $\mathbf{w}_t$ with respect to the sequence of pairwise instances. Algorithm 3 gives the detailed steps of this approach for updating classifiers. The following lemma gives the property of the classifier returned by the sequential updating approach.

Algorithm 2 A **Reservoir Sampling** Approach for UpdateBuffer

```
Input
  • $B^t$: the current buffer,
  • $\mathbf{x}_t$: a training instance,
  • $N$: the buffer size
  • $N_{t+1}$: the number of instances received till trial $t$
Output: updated buffer $B^{t+1}$
if $|B^t| < N$ then
  $B^{t+1} = B^t \cup \{\mathbf{x}_t\}$
else
  Sample $Z$ from a Bernoulli distribution with $\Pr(Z = 1) = N/N_{t+1}$
  if $Z = 1$ then
    Randomly delete an instance from $B^t$
    $B^{t+1} = B^t \cup \{\mathbf{x}_t\}$
  end if
end if
Return $B^{t+1}$
```

Lemma 2. *Assume $|\mathbf{x}_i|_2 \leq 1$ $\forall i$, then after running Algorithm 3, for any $\mathbf{w}$, we have*

$$\begin{aligned}&\mathrm{E}\left[\mathcal{L}_t(\mathbf{w}_t) - \mathcal{L}_t(\mathbf{w})\right]\\ &\leq \frac{1}{C}\mathrm{E}\left[|\mathbf{w}-\mathbf{w}_t|_2^2 - |\mathbf{w}-\mathbf{w}_{t+1}|_2^2\right] + C|B|[N^t_{-y_t}]^2\end{aligned}$$

where $N^t_{-y_t}$ stands for the number of training examples received before trial t that belong to class $-y_t$.

Proof. Since $\mathbf{w}^{i+1}$ is an optimal solution to (6) and the objective function in (6) is strongly convex, we have the following inequality for any $\mathbf{w}$

$$\begin{aligned}|\mathbf{w}-\mathbf{w}^i|_2^2 + C_t\ell(\mathbf{w}, y_t(\mathbf{x}_t - \mathbf{x})) &\geq |\mathbf{w}-\mathbf{w}^{i+1}|_2^2\\ &+|\mathbf{w}^{i+1}-\mathbf{w}^i|_2^2 + C_t\ell(\mathbf{w}^{i+1}, y_t(\mathbf{x}_t-\mathbf{x}))\end{aligned}$$

and therefore

$$\begin{aligned}C_t\left[\ell(\mathbf{w}^{i+1}, y_t(\mathbf{x}_t-\mathbf{x})) - \ell(\mathbf{w}, y_t(\mathbf{x}_t-\mathbf{x}))\right] &\leq\\ |\mathbf{w}-\mathbf{w}^i|_2^2 - |\mathbf{w}-\mathbf{w}^{i+1}|_2^2 - |\mathbf{w}^{i+1}-\mathbf{w}^i|_2^2\end{aligned}$$

Let $B = \left\{\mathbf{x}_B^1, \ldots, \mathbf{x}_B^{|B|}\right\}$. Adding all the inequalities together, we have

$$\begin{aligned}&C_t\sum_{i=1}^{|B|}\ell(\mathbf{w}^{i+1}, y_t(\mathbf{x}_t-\mathbf{x}_B^i)) - \ell(\mathbf{w}, y_t(\mathbf{x}_t-\mathbf{x}_B^i))\\ &\leq |\mathbf{w}-\mathbf{w}^1|_2^2 - |\mathbf{w}-\mathbf{w}^{|B|+1}|_2^2 - \sum_{i=1}^{|B|}|\mathbf{w}^{i+1}-\mathbf{w}^i|_2^2\end{aligned}$$

Using the fact

$$\begin{aligned}&|\ell(\mathbf{w}^{i+1}, y_t(\mathbf{x}_t-\mathbf{x})) - \ell(\mathbf{w}^1, y_t(\mathbf{x}_t-\mathbf{x}))|\\ &\leq 2|\mathbf{w}^{i+1}-\mathbf{w}^1|_2 \leq 2\sum_{j=1}^{i}|\mathbf{w}^{j+1}-\mathbf{w}^j|_2\end{aligned}$$

we have

$$\begin{aligned}&C_t\left[\sum_{i=1}^{|B|}\ell(\mathbf{w}^1, y_t(\mathbf{x}_t-\mathbf{x}_B^i)) - \ell(\mathbf{w}, y_t(\mathbf{x}_t-\mathbf{x}_B^i))\right]\\ &\leq |\mathbf{w}-\mathbf{w}^1|_2^2 - |\mathbf{w}-\mathbf{w}^{B+1}|_2^2\\ &\quad +2\sum_{i=1}^{|B|}\left(C_t|B||\mathbf{w}^{i+1}-\mathbf{w}^i|_2 - |\mathbf{w}^{i+1}-\mathbf{w}^i|_2^2\right)\\ &\leq |\mathbf{w}-\mathbf{w}^1|_2^2 - |\mathbf{w}-\mathbf{w}^{B+1}|_2^2 + C_t^2|B|^3\end{aligned}$$

The last step follows $-x^2 + 2ax \leq a^2$. Define $D^t(-y_t) = \{\mathbf{x}_{t'} : t' \in [t-1], y_{t'} = -y_t\}$. According to the theory of Reservoir sampling, each element in buffer B is a uniform sampling from $D^t(-y_t)$. Using Lemma 1, we have

$$\begin{aligned}&\frac{1}{|B|}\mathrm{E}\left[\sum_{i=1}^{|B|}\ell(\mathbf{w}^1, y_t(\mathbf{x}_t-\mathbf{x}_B^i))\right] =\\ &\frac{1}{|D^t(-y_t)|}\sum_{\mathbf{x}\in D^t(-y_t)}\ell(\mathbf{w}^1, y_t(\mathbf{x}_t-\mathbf{x}))\end{aligned}$$

We complete the proof by using the fact $|D^t(-y_t)| = N^t_{-y_t}$ and the definition of $\mathcal{L}_t$ in (3). □

The following theorem regarding the regret bound directly follows Lemma 2.

Theorem 1. *After running the Algorithm 1 with (i) the sequential updating in Algorithm 3 for **Update-Classifier** and (ii) the reservoir sampling in Algorithm 2 for **UpdateBuffer**, for any $\mathbf{w}$, we have*

$$\mathrm{E}\left[\sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}_t)\right] \leq \sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}) + \frac{|\mathbf{w}|_2^2}{C} + \frac{C}{3}\left(N_+T_+^3 + N_-T_-^3\right)$$

where T_+ and T_- are the total number of positive and negative instances received over T trials. For any $|\mathbf{w}|_2 \leq D$, choosing $C = \sqrt{3}D/\sqrt{N_+T_+^3 + N_-T_-^3)}$, we have

$$\mathrm{E}\left[\sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}_t)\right] \leq \sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}) + D\sqrt{3(N_+T_+^3 + N_-T_-^3)}$$

Algorithm 3 A **Sequential Updating** Approach for UpdateClassifier

```
Input
  • $\mathbf{w}_t$: the current classifier,
  • $(\mathbf{x}_t, y_t)$: a training example,
  • $B$: the buffer to which $(\mathbf{x}_t, y_t)$ will be compared
  • $C_t$: a parameter that weights the comparison
    between $(\mathbf{x}_t, y_t)$ and $B$
Output: updated classifier $\mathbf{w}_{t+1}$
Initialize $\mathbf{w}^1 = \mathbf{w}_t$ and $i = 1$
for $\mathbf{x} \in B$ do
  Update classifier $\mathbf{w}^i$ by

  $\mathbf{w}^{i+1} = \arg\min_{\mathbf{w}} |\mathbf{w} - \mathbf{w}^i|_2^2 + C_t \ell(\mathbf{w}, y_t(\mathbf{x}_t - \mathbf{x}))$   (6)

  $i = i + 1$
end for
Return $\mathbf{w}_{t+1} = \mathbf{w}^{|B|+1}$
```

Remark. First, since $\sum_{t=1}^{T} \mathcal{L}_t(\mathbf{w}) \sim O(T_+T_-)$, it can be significantly larger than $\sqrt{N_+T_+^3 + N_-T_-^3}$ for a large T, if D, N_+ and N_- are assumed to be constant. Second, the result in Theorem 1 may contradict the intuition since the regret bound increases as the buffer size increases. This is because it only bounds the expectation. If we construct the high probability bound, there will be an additional term $O(T_+/\sqrt{N_+} + T_-/\sqrt{N_-})$ due to the variance of the reservoir sampling. This variance term decreases as buffer size increases. It is interesting to see that the optimal buffer size is $N_+ = O(\sqrt{T_+})$ and $N_- = O(\sqrt{T_-})$, which is consistent with our empirical observation that a larger buffer size does not necessarily improve the AUC metric.

Finally, to run Algorithm 3 efficiently, the following proposition gives the closed-form solution to (6).

Proposition 1. *For the optimization problem (6), its closed-form solution is given by $\mathbf{w}^{i+1} = \mathbf{w}^i + \tau y_t[\mathbf{x}_t - \mathbf{x}]$ where τ can be computed by:*

$$\tau = \min\left\{\frac{C_t}{2}, \frac{\ell(\mathbf{w}^i, y_t(\mathbf{x}_t - \mathbf{x}))}{|\mathbf{x}_t - \mathbf{x}|_2^2}\right\}.$$

The above proposition is similar to the updating rules derived in (Crammer et al., 2006).

Gradient Updating The second approach is to treat $\mathcal{L}_t(\mathbf{w})$ as a single loss function and apply the gradient descent approach to update the solution $\mathbf{w}_t$. This is given in Algorithm 4. The following lemma gives the property of the classifier returned by the gradient updating approach.

Algorithm 4 A **Gradient Updating** Approach for UpdateClassifier

```
Input
  • $\mathbf{w}_t$: the current classifier,
  • $(\mathbf{x}_t, y_t)$: a training example,
  • $B$: the buffer to which $(\mathbf{x}_t, y_t)$ will be compared
  • $C_t$: a parameter that weights the comparison
    between $(\mathbf{x}_t, y_t)$ and $B$
Output: updated classifier $w_{t+1}$
Initialize $\mathbf{w}_{t+1} = \mathbf{w}_t$
for $\mathbf{x} \in B$ do
  if $y_t\mathbf{w}_t \cdot (\mathbf{x}_t - \mathbf{x}) \leq 1$ then
    $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} + C_t y_t(\mathbf{x}_t - \mathbf{x})/2$
  end if
end for
Return $\mathbf{w}_{t+1}$
```

Lemma 3. *Assume $|\mathbf{x}_i|_2 \leq 1\ \forall i$, then after running Algorithm 4, for any $\mathbf{w}$, we have*

$$\begin{aligned}&\mathrm{E}\left[\mathcal{L}_t(\mathbf{w}_t) - \mathcal{L}_t(\mathbf{w})\right]\\ \leq\ & \frac{1}{C}\mathrm{E}\left[|\mathbf{w} - \mathbf{w}_t|_2^2 - |\mathbf{w} - \mathbf{w}_{t+1}|_2^2\right] + C|B|[N_{-y_t}^t]^2\end{aligned}$$

where $N_{-y_t}^t$ stands for the number of training examples received before trial t that belong to class $-y_t$.

We skip the proof since it directly follows the property of gradient descent methods for online learning and is very similar to that for Lemma 3. Although the gradient updating approach gives the same guarantee as that of the sequential updating approach, it simplicity in computation makes it more attractive for large classification problems. Finally, since the regret bound of Algorithm 1 using the gradient descent approach for UpdateClassifier is identical to that using the sequential updating approach, it is skipped here.

4. Experimental Results

In this section, we evaluate the empirical performance of the proposed Online AUC Maximization (OAM) algorithms for cost-sensitive online learning tasks.

4.1. Compared Algorithms

We compare the proposed OAM algorithms with the state-of-the-art online learning algorithms. Since our study is focused on online learning, for fair comparison, we do not compare with existing batch AUC studies. Specifically, the compared algorithms in our experiments include:

- "Perceptron": the classical perceptron algorithm (Rosenblatt, 1958);

- "PA": the Passive-Aggressive algorithm (the PA-I algorithm) (Crammer et al., 2006);
- "CW-full": the confidence-based weighted online learning algorithm (Crammer et al., 2008);
- "CPA_{PB}": the Prediction-Based Cost-sensitive Passive-Aggressive algorithm (Crammer et al., 2006);
- "CPA_{ML}": the Max-Loss Cost-sensitive Passive-Aggressive algorithm (Crammer et al., 2006);
- "OAM_{seq}": the proposed OAM algorithm by the sequential updating approach;
- "OAM_{gra}": the proposed OAM algorithm by the gradient descent updating approach;
- "OAM_{inf}": the proposed OAM algorithm by assuming infinite buffer size so that we can ideally store all the historical examples.

The last OAM_{inf} algorithm is included to examine how effective is the proposed reservoir sampling approach used in the OAM_{seq} and OAM_{gra} algorithms.

4.2. Experimental Testbed and Setup

To extensively examine the performance, we conduct experiments on a variety of benchmark datasets from web machine learning repositories. Table 1 shows the details of 12 binary-class datasets used in our experiments. All of these datasets can be downloaded from LIBSVM website [1] and UCI machine learning repository [2]. Note that several datasets (segment, satimage, vowel, letter, poker) are originally multi-class, which were converted to class-imbalanced binary datasets in our experiments. These datasets are chosen fairly randomly in order to cover different properties, e.g. datasets of various class-imbalance ratios.

For each dataset, we randomly divide it into 5 folds. We choose 4 folds for training and the remaining one fold as test set. To reduce the variance in results, for each dataset, we generate 4 independent 5-fold partitions, leading to a total of 20 runs per dataset. The reported AUC results are averaged over 20 runs.

To make fair comparisons, all algorithms adopt the same setup. For the CW-full algorithm, we apply a 5-fold cross-validation to the training set to find the best $\eta \in [0.5 : 0.1 : 1]$. For the two CPA_{PB} and CPA_{ML} algorithms, a similar 5-fold cross-validation is applied with a grid search to find the best parameter $C \times \rho(+1; -1) \in 2^{[-10:10]} \times 2^{[0:10]}$. For the proposed OAM algorithms, we fix $N_- = N_+ = 100$ for every case, and adopt the similar 5-fold cross-validation to find the best penalty parameter $C \in 2^{[-10:10]}$.

Table 1. Details of the datasets in our experiments.

Dataset	# instances	# dimensions	T_-/T_+
sonar	208	60	1.1443
fourclass	862	2	1.8078
svmguide1	3089	4	1.8365
magic04	19020	10	1.8439
german	1000	24	2.3333
svmguide3	1243	22	3.1993
segment	2310	19	6.0000
ijcnn1	141691	22	9.4453
satimage	4435	36	9.6867
vowel	528	10	10.0000
letter	15000	16	26.8810
poker	25010	10	47.7524

After the best parameters are found, all the experiments were conducted over 20 runs of different random permutations for each dataset. All the results were reported by averaging over these 20 runs. For performance metric, we evaluate the online learning algorithms by measuring AUC value on the test set.

4.3. Performance Evaluation

Table 2 summarizes the average AUC performance of the compared algorithms over the 12 datasets. From the experimental results, we can draw several observations as follows.

Table 2. Evaluation of average AUC performance.

Algorithm	sonar	fourclass	svmguide1
Perceptron	0.780 ± 0.060	0.690 ± 0.165	0.883 ± 0.140
PA-I	0.790 ± 0.057	0.668 ± 0.168	0.799 ± 0.219
CW-full	0.793 ± 0.059	0.758 ± 0.032	0.922 ± 0.019
CPA_{PB}	0.798 ± 0.059	0.812 ± 0.020	0.891 ± 0.007
CPA_{ML}	0.827 ± 0.052	0.812 ± 0.020	0.885 ± 0.008
OAM_{seq}	0.850 ± 0.042	0.831 ± 0.020	0.988 ± 0.003
OAM_{gra}	0.849 ± 0.043	0.826 ± 0.020	0.988 ± 0.002
OAM_{inf}	0.849 ± 0.043	0.831 ± 0.020	0.989 ± 0.002
Algorithm	**magic04**	**german**	**svmguide3**
Perceptron	0.723 ± 0.069	0.701 ± 0.039	0.696 ± 0.038
PA-I	0.564 ± 0.111	0.701 ± 0.033	0.707 ± 0.037
CW-full	0.746 ± 0.027	0.757 ± 0.025	0.723 ± 0.036
CPA_{PB}	0.730 ± 0.030	0.698 ± 0.034	0.707 ± 0.038
CPA_{ML}	0.734 ± 0.026	0.701 ± 0.033	0.707 ± 0.037
OAM_{seq}	0.778 ± 0.029	0.775 ± 0.037	0.760 ± 0.035
OAM_{gra}	0.765 ± 0.032	0.773 ± 0.033	0.755 ± 0.034
OAM_{inf}	0.784 ± 0.026	0.776 ± 0.034	0.768 ± 0.035
Algorithm	**segment**	**ijcnn1**	**satimage**
Perceptron	0.852 ± 0.024	0.647 ± 0.088	0.605 ± 0.025
PA-I	0.863 ± 0.021	0.531 ± 0.074	0.646 ± 0.024
CW-full	0.896 ± 0.021	0.829 ± 0.021	0.619 ± 0.024
CPA_{PB}	0.888 ± 0.018	0.908 ± 0.012	0.811 ± 0.022
CPA_{ML}	0.886 ± 0.021	0.910 ± 0.011	0.828 ± 0.024
OAM_{seq}	0.956 ± 0.013	0.920 ± 0.017	0.919 ± 0.014
OAM_{gra}	0.955 ± 0.014	0.916 ± 0.018	0.911 ± 0.017
OAM_{inf}	0.956 ± 0.013	0.928 ± 0.015	0.921 ± 0.013
Algorithm	**vowel**	**letter**	**poker**
Perceptron	0.859 ± 0.055	0.551 ± 0.092	0.502 ± 0.031
PA-I	0.863 ± 0.063	0.533 ± 0.104	0.506 ± 0.028
CW-full	0.870 ± 0.063	0.804 ± 0.025	0.457 ± 0.058
CPA_{PB}	0.887 ± 0.049	0.784 ± 0.056	0.508 ± 0.068
CPA_{ML}	0.923 ± 0.032	0.802 ± 0.035	0.524 ± 0.067
OAM_{seq}	0.931 ± 0.046	0.820 ± 0.016	0.592 ± 0.060
OAM_{gra}	0.928 ± 0.046	0.817 ± 0.023	0.586 ± 0.062
OAM_{inf}	0.931 ± 0.041	0.828 ± 0.010	0.594 ± 0.067

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/
[2] http://www.ics.uci.edu/~mlearn/MLRepository.html

First of all, by examining the three regular online learning algorithms (Perceptron, PA, CW-full), we found that CW-full achieved the best performance among them for most cases, while PA yields the worst performance. We attribute the poor performance of PA to its aggressive updating strategy that does not take into account the class-imbalance issue. This result indicates the importance of developing cost-sensitive online learning techniques. This is further verified by the observation that the two cost-sensitive PA algorithms, CPA_{PB} and CPA_{ML}, do achieve considerably better AUC performance than regular PA algorithm on most of the datasets.

Second, by comparing the proposed OAM algorithms against the existing online learning algorithms, we found that the OAM algorithms significantly surpass all the existing online learning algorithms on most datasets. For example, on dataset "svmguide3", the AUC values for the baseline online learning algorithms are lower than 70%, while the OAM algorithms are able to attain 76% for average AUC. These results showed that the OAM algorithms are significantly more effective than regular online learning algorithms for cost-sensitive learning tasks.

Third, by examining the proposed OAM algorithms, we found that the two OAM algorithms using fixed buffer sizes (OAM_{seq} and OAM_{gra}) perform comparably on all the datasets. Further, by comparing these two algorithms against the OAM_{inf} algorithm using unlimited buffers, we found that the AUC performance of the OAM algorithms with fixed buffer sizes are in general fairly comparable to that of the OAM_{inf} algorithm. These encouraging results showed that the OAM algorithms of fixed buffer sizes are able to maintain an accurate sketch of historical training examples by exploring the reservoir sampling technique.

To further examine the efficacy of the reservoir sampling technique for OAM, in the next subsection, we conduct additional experiments to examine the effect of buffer sizes on the two OAM algorithms.

4.4. Evaluation of Varied Buffer Sizes

Figure 1 evaluates the AUC performance of the two OAM algorithms (OAM_{seq} and OAM_{gra}) with varied buffer sizes across several different datasets. Several observations can be drawn from the results.

First of all, we found that when the buffer size is extremely small (e.g. buffer size equal to 1), the two OAM algorithms (OAM_{seq} and OAM_{gra}) achieved the lowest AUC performance on all datasets. This is reasonable as it is almost impossible to maintain a good sketch of the history using only such small buffer sizes.

Second, we observe that when we slightly increase the buffer size to a larger value, the AUC performance of the OAM algorithms can be boosted considerably. This shows that the reservoir sampling technique is able to sample good historical examples with limited buffer size towards online AUC maximization tasks.

Finally, we found that when the buffer size is large enough, the AUC performances of OAM algorithms tend to become saturated, where further increasing the buffer size has very limited improvement. In some cases, it may lead to the reduction in AUC. This is consistent with the remark we made before, i.e., the optimal buffer size is $O(\sqrt{T})$ and a larger buffer size does not necessarily lead to a better AUC performance.

5. Conclusion

This paper studied a new type of online learning problem, i.e., Online AUC Maximization (OAM), which aims to online learn a model by maximizing the AUC metric. It is more challenging than conventional online learning tasks where the goal is often to minimize the mistake rate of online predictions. The key challenge of OAM is that it requires to minimize the losses between any pair of two instances belonging to different classes, which usually needs to memorize all the received training instances in the online learning process. To address this challenge, we proposed an effective framework for OAM by applying the reservoir sampling technique, which is able to maintain a good sketch of history in fixed-size buffers. We presented two different OAM algorithms and theoretically analyzed the bounds of the proposed algorithms. Finally, our promising results from extensive experiments validate the empirical efficacy of our algorithms.

Acknowledgments

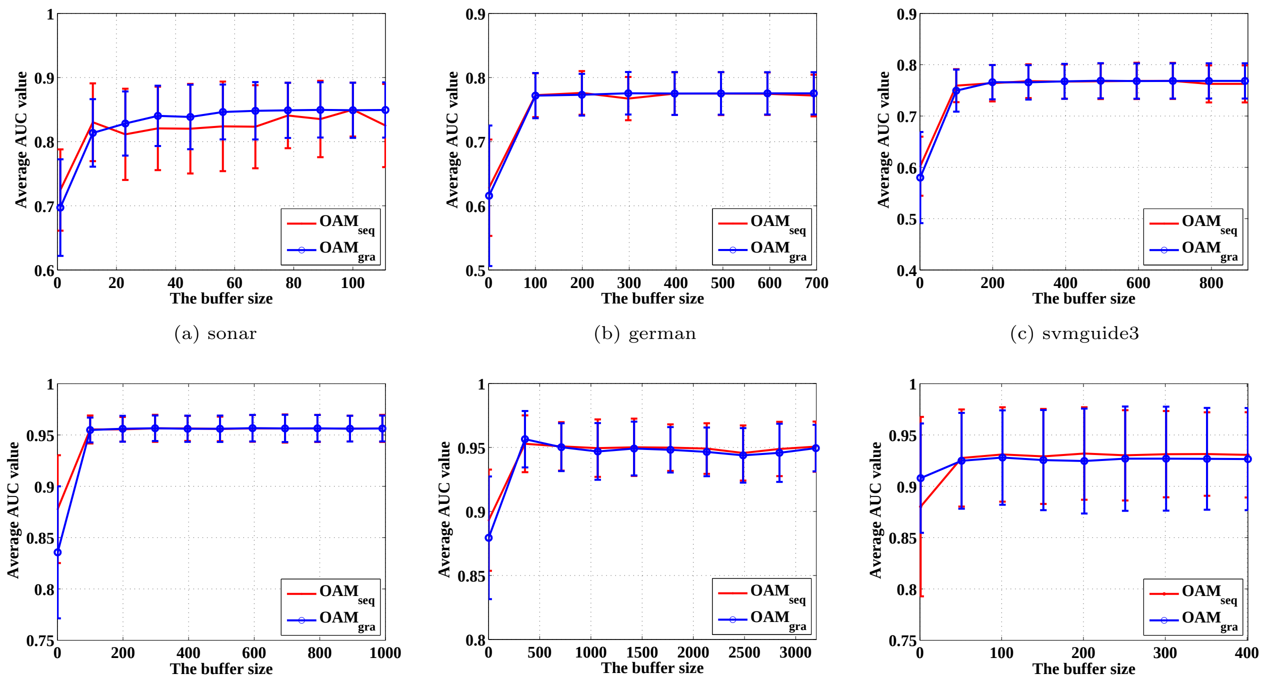
This work was supported by Singapore MOE ARC tier-2 research grant (T208B2203), National Science Foundation (IIS-0643494), and Office of Navy Research (N00014-09-1-0663).

References

Bradley, Andrew P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

Brefeld, Ulf and Scheffer, Tobias. Auc maximizing support vector learning. In *In Proc. ICML workshop on ROC Analysis in Machine Learning*, 2005.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, Learning, and Games.* Cambridge University Press, 2006.



(a) sonar (b) german (c) svmguide3

(d) segment (e) satimage (f) vowel

Figure 1. Evaluation of *AUC* performance with respect to varied buffer sizes (i.e., varied $N_{+1} = N_{-1}$ values).

Cortes, Corinna and Mohri, Mehryar. Auc optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Crammer, Koby and Singer, Yoram. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

Crammer, Koby, Dekel, Ofer, Keshet, Joseph, Shalev-Shwartz, Shai, and Singer, Yoram. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Crammer, Koby, Dredze, Mark, and Pereira, Fernando. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 345–352, 2008.

Dredze, Mark, Crammer, Koby, and Pereira, Fernando. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning (ICML2008)*, pp. 264–271, 2008.

Elkan, Charles. The foundations of cost-sensitive learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 973–978, 2001.

Freund, Yoav and Schapire, Robert E. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, 1999.

Gentile, Claudio. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

Hanley, James A. and McNeil, Barbara J. The meaning and use of the area under of receiver operating characteristic (roc) curve. In *Radiology*, 1982.

Herschtal, Alan and Raskutti, Bhavani. Optimising area under the roc curve using gradient descent. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

Joachims, Thorsten. A support vector method for multivariate performance measures. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML2005)*, pp. 377–384, 2005.

Li, Yaoyong, Zaragoza, Hugo, Herbrich, Ralf, Shawe-Taylor, John, and Kandola, Jaz S. The perceptron algorithm with uneven margins. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML2002)*, pp. 379–386, 2002.

Rakotomamonjy, Alain. Optimizing area under roc curve with svms. In *1st International Workshop on ROC Analysis in Artificial Intelligence (ROCAI)*, pp. 71–80, 2004.

Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

Vitter, Jeffrey Scott. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

Yan, Lian, Dodier, Robert H., Mozer, Michael, and Wolniewicz, Richard H. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 848–855, 2003.

Zinkevich, Martin. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 928–936, 2003.