Non-Parametric Online AUC Maximization

Balázs Szörényi[1,2,3], Snir Cohen[1,4], Shie Mannor[1,5]

[1] Technion, Haifa, Israel
[2] Research Group on AI, Hungarian Acad. Sci. and Univ. of Szeged, Szeged, Hungary
[3] szorenyi.balazs@gmail.com
[4] snirc@cs.technion.ac.il
[5] shie@ee.technion.ac.il

Abstract. We consider the problems of online and one-pass maximization of the area under the ROC curve (AUC). AUC maximization is hard even in the offline setting and thus solutions often make some compromises. Existing results for the online problem typically optimize for some proxy defined via surrogate losses instead of maximizing the real AUC. This approach is confirmed by results showing that the optimum of these proxies, over the set of all (measurable) functions, maximize the AUC. The problem is that—in order to meet the strong requirements for per round run time complexity—online methods typically work with restricted hypothesis classes and this, as we show, corrupts the above compatibility and causes the methods to converge to suboptimal solutions even in some simple stochastic cases. To remedy this, we propose a different approach and show that it leads to asymptotic optimality. Our theoretical claims and considerations are tested by experiments on real datasets, which provide empirical justification to them.

1 Introduction

The *area under the ROC curve* (AUC) [16] measures how well a mapping h of the instance space to the reals respects the partial order defined by some "ideal" score function s; in the special case of biparite ranking, s is simply a 0-1 valued function. As such, it has important applications in bioinformatics, information retrieval, anomaly detection, and many other areas.

Maximizing the AUC requires an approach different from maximizing the accuracy, even though there are some connections between the two [3, 5, 11]. Over the last decade, several approaches have been proposed and analyzed, guaranteeing consistency [9] and even optimal learning rates in some restricted cases [19]. Subsequently [22], followed by [21, 13], considered AUC maximization in an online setting, while [14] introduced a one-pass AUC maximization framework.

In this paper we first point out two important shortcomings of the existing methods proposed for online and one-pass AUC optimization:

(A) None of them guarantees an optimal solution (not even asymptotically).
(B) They all need to store the whole data. The reason for this is that they require parameter tuning, and thus also multiple passes over the data.

In contrast to (A), the k nearest neighbor method (k-*NN*), as we show, is guaranteed to converge to the optimum. This superiority of k-NN is also supported by the results of our empirical investigations. What is more, even though it clearly requires storing the whole data, it is not more demanding in terms of space complexity then previous algorithms, according to (B). Finally, one could argue that k-NN must perform poorly in terms of running time. This is not true, however: efficient solutions exist and, in fact, our experiments suggest that k-NN is competitive also in this regard. Additionally, dimensionality-originated issues can be taken care of using PCA or related methods.

The rest of the paper is structured as follows. First we introduce the formal framework and the definitions, then we show (A) formally, provide the theoretical justification for the k-NN method, present our experimental results, sum up the most important results from the literature, and finally we conclude with a short discussion.

2 Formal Setup

Given a set of n samples $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ for some positive integer d and $\mathcal{Y} = \{-1, 1\}$, and given some mapping $h : \mathcal{X} \to \mathbb{R}$, the *area under the ROC curve* (AUC) [16] is the empirical mean

$$\mathrm{AUC}(h; \mathcal{X}^+, \mathcal{X}^-) = \sum_{x^+ \in \mathcal{X}^+} \sum_{x^- \in \mathcal{X}^-} \left(\frac{\mathbb{I}\left[h(x^+) > h(x^-)\right]}{T_+ T_-} + \frac{\mathbb{I}\left[h(x^+) = h(x^-)\right]}{2T_+ T_-} \right) \ ,$$

where $\mathcal{X}^+ = \{x_t : y_t = 1, 1 \le t \le n\}$, $\mathcal{X}^- = \{x_t : y_t = -1, 1 \le t \le n\}$, $T_+ = |\mathcal{X}^+|$, $T_- = |\mathcal{X}^-|$, and where $\mathbb{I}[\cdot]$ denotes the indicator function; i.e., $\mathbb{I}[E] = 1$ when event E holds and $\mathbb{I}[E] = 0$ otherwise. The regret hypothesis h with respect to some hypothesis set $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$ is defined as

$$\mathrm{Regret}^{\mathcal{H}}(h; \mathcal{X}^+, \mathcal{X}^-) = \sup_{h' \in \mathcal{H}} \mathrm{AUC}(h'; \mathcal{X}^+, \mathcal{X}^-) - \mathrm{AUC}(h; \mathcal{X}^+, \mathcal{X}^-) \qquad (1)$$

We also denote by $\mathrm{AUC}^*(\mathcal{X}^+, \mathcal{X}^-)$ the supremum of $\mathrm{AUC}(h'; \mathcal{X}^+, \mathcal{X}^-)$ over the set of all measurable functions h', and introduce the notation

$$\mathrm{Regret}(h; \mathcal{X}^+, \mathcal{X}^-) = \mathrm{AUC}^*(\mathcal{X}^+, \mathcal{X}^-) - \mathrm{AUC}(h; \mathcal{X}^+, \mathcal{X}^-).$$

Maximizing the AUC is also equivalent to minimizing the empirical risk

$$\mathrm{Risk}(h; \mathcal{X}^+, \mathcal{X}^-) = 1 - \mathrm{AUC}(h; \mathcal{X}^+, \mathcal{X}^-) = \sum_{t,t'=1}^{n} \frac{\ell^{\mathrm{AUC}}(h; (x_t, y_t), (x_{t'}, y_{t'}))}{2T_+ T_-} \qquad (2)$$

of the loss function

$$\ell^{\mathrm{AUC}}(h; (x, y), (x', y')) = \mathbb{I}\left[(h(x) - h(x'))(y - y') < 0\right] + \mathbb{I}\left[h(x) = h(x'), y \ne y'\right].$$

One notorious problem with AUC is that it is non-convex and non-continuous, which makes it hard to work with. Especially in the online and one-pass settings, where having a low (typically constant or logarithmic, but at least sublinear) per round run time complexity is essential. To resolve this issue, papers that aim for maximizing AUC online [22, 21, 13, 14], choose to replace ℓ^{AUC} in (2) by some surrogate loss function $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$, and instead of maximizing the AUC, they minimize the surrogate risk

$$\mathrm{Risk}^{\ell}(h; \mathcal{X}^+, \mathcal{X}^-) = \sum_{t,t'=1}^{t} \tfrac{\ell(h;(x_t,y_t),(x_{t'},y_{t'}))}{2T_+T_-} \quad ,$$

and derive bounds for $\mathrm{Regret}^{\ell,\mathcal{H}}(h; \mathcal{X}^+, \mathcal{X}^-)$, which is obtained by replacing AUC in (1) by $1 - \mathrm{Risk}^{\ell}$.

If $(x_,y_1), (x_2, y_2), \ldots$ are i.i.d. samples from some probability distribution $\mathbf{P}$ over $\mathcal{X} \times \mathcal{Y}$, then one can define

$$\mathrm{Risk}(h) = \mathbf{E}\left[\ell^{\mathrm{AUC}}(h; (X, Y), (X', Y')) \,\middle|\, Y > Y'\right] \tag{3}$$

and $\mathrm{AUC}(h) = 1 - \mathrm{Risk}(h)$. Similarly as above, replacing ℓ^{AUC} in (3) by some other loss function ℓ one obtains surrogate measures $\mathrm{Risk}^{\ell}(h)$ and $\mathrm{Regret}^{\ell,\mathcal{H}}(h)$. Along the same analogy, we also use the notation AUC^* and $\mathrm{Regret}(h)$.

One-pass and Online Setting

The one-pass and the online settings both have the same underlying protocol: in each round t, the learner proposes some hypothesis $h_t : \mathcal{X} \to \mathbb{R}$ based on its previous experience, and then it observes the sample (x_t, y_t). The two frameworks only differ in their objectives:

- In the *online setting* we are concerned with the evolution of the empirical AUC—that is, with
$$\mathrm{AUC}_t = \mathrm{AUC}(h_t; \mathcal{X}_t^+, \mathcal{X}_t^-)$$
for $t = 1, \ldots, n$, where $\mathcal{X}_t^+ = \{x_i : y_i = 1, 1 \le i \le t\}$ and $\mathcal{X}_t^- = \{x_i : y_i = -1, 1 \le i \le t\}$.
- In the *one-pass setting* the generalization ability of the learner is tested after the whole data had been processed. More precisely, the measure of performance is $\mathrm{AUC}(h_n)$.

3 Surrogate Measures and Restricted Classes

Existing results for the online and one-pass AUC maximization problem optimize for some surrogate risk, instead of working with the AUC directly. In particular, many of them work with the square loss ℓ_2 (see Example 1). This approach is also confirmed by results showing *consistency* i.e., that $\mathrm{Regret}(h_t)$ converges to 0 whenever $\mathrm{Regret}^{\ell}(h_t)$ does for some sequence $h_1, h_2, \ldots$ of functions, as t goes to infinity. (See more about this in the section about the related work.)

These important results require, however, careful interpretation. And this is the starting point of our investigations: we claim that utilization of consistency is only legitimate when the hypotheses class $\mathcal{H}$ of our interest contains a global optimizer of the surrogate loss; that is, if $\sup_h \mathrm{AUC}^{\ell}(h) = \sup_{h \in \mathcal{H}} \mathrm{AUC}^{\ell}(h)$. When working with the set $\mathcal{H}_{\mathrm{lin}} = \{h^w(x) = w^\top x : w \in \mathbb{R}^d\}$ of linear functions—which is the case for *all* existing results for online and one-pass AUC maximization—this criterion is not fulfilled. Indeed, even though square loss is consistent (see [14]), in Example 1 below it holds that $\mathrm{AUC}(h') \ll \sup_{h \in \mathcal{H}_{\mathrm{lin}}} \mathrm{AUC}(h)$ for any $h' \in \mathrm{argmin}_{h \in \mathcal{H}_{\mathrm{lin}}} \mathrm{Risk}^{\ell_2}(h)$.

The hinge loss $\ell^\gamma(h; (x, y), (x', y')) = \mathbb{I}[y \neq y'] \left[\gamma - \frac{1}{2}(y - y')(h(x) - h(x'))\right]_+$ was also used in algorithmic solutions, but that does not even satisfy consistency [15].

3.1 Square Loss with Linear Hypotheses

This section presents the example that existing results can fail completely in maximizing the real AUC even in a simple case. This is demonstrated by the following example.

Example 1 *Consider the setting when* $\mathcal{X} = \mathbb{R}^2$, *and* $\mathbf{P}[X = (-\epsilon, -1 + \epsilon)|Y = 1] = 1$ *and* $\mathbf{P}[X = (0, -1 - \epsilon)|Y = -1] = \mathbf{P}[X = (0, 1)|Y = -1] = \mathbf{P}[X = (1, 0)|Y = -1] = 1/3$ *for some small* $\epsilon > 0$.

[14] first shows that the square loss $\ell_2(h; (x, y), (x', y')) = (1 - \frac{y - y'}{2}(h(x) - h(x'))^2$ *is consistent with* AUC, *and then use* ℓ_2 *as a surrogate loss to find the best linear score function* $h^w(x) = w^\top x$. *However, in the case above,* AUC *is maximized at* h^{w^*} *where* $w^* = (-1, 0)$, *where it actually takes value 1 (one has a very small freedom though, depending on the size of* ϵ*), and thus* $\mathrm{Risk}(h^{w^*}) = 0$ *for the corresponding linear function* h^{w^*}. *On the other hand, the surrogate measure for* h^{w^*} *is*

$$\mathrm{Risk}^{\ell_2}(h^{w^*}) = \mathbf{E}[\ell_2(h^{w^*}; X, X')|Y > Y']$$
$$= \tfrac{1}{3}\left(1 + (w^*)^\top(\epsilon, 2\epsilon)\right)^2 + \tfrac{1}{3}\left(1 + (w^*)^\top(\epsilon, 2 - \epsilon)\right)^2 + \tfrac{1}{3}\left(1 + (w^*)^\top(1 + \epsilon, 1 - \epsilon)\right)^2$$

which evaluates approximately to 2/3. At the same time, the actual optimum w' *of this surrogate measure is around* $(-1/2, -1/2)$, *where it takes the value:*

$$\mathrm{Risk}^{\ell_2}(h^{w'}) = \mathbf{E}[\ell_2(h; X, X')|Y > Y'] \approx 1/3 \ ,$$

whereas $\mathrm{AUC}^*(h^{w'}) \approx 2/3$, *which is very far from the true optimum.*

That is, when $\mathcal{H}$ *consists of the linear hypotheses (as is the case in [14] and [13]), then*

$$\mathrm{Regret}^{\ell_2, \mathcal{H}}(h^{w'}) = 0 \ ,$$

implying that

$$\mathrm{Regret}^{\mathcal{H}}(h^{w'}) = \mathrm{Regret}(h^{(-1/2, -1/2)}) \approx 1/3 \ .$$

Furthermore, adding a term $\|w\|^2$ *to regularize the surrogate measure does not change on this.*

Note that this does not contradict the consistency of the square loss. The reason is that consistency requires Regret to vanish as Regret^ℓ approaches 0, but as Regret^ℓ is huge for all linear hypotheses, this sets no restrictions on how Regret should behave in this example.

4 Conditional Probability as Rank Function

In this section we start the investigation of finding alternative algorithmic solutions. With that in mind, we reach back to the fundaments of AUC, and show that good estimates of the conditional probability function $\eta(x) = \mathbf{P}[Y = 1|X = x]$ perform well at AUC maximization too.

The particular estimates that we consider here are of the form $\widehat{\eta} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ for some domain $\mathcal{Z}$. Here $\mathcal{Z}$ is the domain of a variable that is used to encode prior information (e.g., random samples) and internal randomization (used e.g., for tie breaking) of the learner. (In accordance with that, in some cases it will be more convenient to use the notation $\widehat{\eta}_z$ for $\widehat{\eta}(\,\cdot\,, z)$.) For example, given some series $\{k_n\}_n$ of stepsizes, the k_n-NN estimate of [12] makes use of some i.i.d. samples $U_1, \ldots, U_n, U$ drawn from the uniform distribution over $[0, 1]$. Putting $Z_n = (X_1, Y_1, \ldots, X_n, Y_n, U_1, \ldots, U_n, U)$, their k_n-NN estimate maps an instance x to

$$\widehat{\eta}^{\mathrm{DGKL}}_{Z_n}(x) = \frac{1}{k_n} \sum_{i=1}^{k_n} Y_{\sigma(Z_n,x,i)} \ , \tag{4}$$

where $\sigma(Z_n, x, \,\cdot\,)$ is the permutation for which $(\|X_{\sigma(Z_n,x,1)} - x\|, \|U_{\sigma(Z_n,x,1)} - U\|), \ldots, (\|X_{\sigma(Z_n,x,n)} - x\|, \|U_{\sigma(Z_n,x,n)} - U\|)$ is in lexicographic order.

Given such an estimate, we show the following result (for the proof see Appendix A).

Theorem 2 *Let Z be some random variable over some domain $\mathcal{Z}$, and let $\widehat{\eta} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ be an estimate of the conditional distribution function $\eta(x) := \mathbf{E}[Y|X = x]$ as described above. Then $\mathbf{E}_Z\left[\mathrm{Regret}(\widehat{\eta}_Z)\right] \leq \frac{3\sqrt{\epsilon}}{\mathbf{P}[Y=1]\mathbf{P}[Y=0]}$, where $\epsilon = \mathbf{E}_{X,Z}[|\widehat{\eta}(X, Z) - \eta(X)|]$.*

Similar result has also appeared in [10, 1, 19]. However, this particular estimator requires some small but essential differences in the analysis. Most importantly, kernel estimators are completely determined by the samples, whereas k_n-NN needs tie breaking. This requires additional randomness and complicates the analysis slightly.

5 AUC Maximization Using $k-$NN

In the previous section we have shown guarantees for the AUC performance of estimators of the conditional probability function η. In this section we review some of the results on estimating η using k_n-NN, and show what they give combined with Theorem 2.

First of all, Devroye et al. [12] have shown that the k_n-NN version presented as Algorithm 1 converges under any distribution, assuming some standard restrictions on k_n. [6]

[6] They actually show an even stronger equivalence result.

Theorem 3 (Theorem 1 in [12]) *If stepsize k_n satisfies $\lim_{n\to\infty} k_n = \infty$ and $\lim_{n\to\infty} k_n/n = 0$, then $\mathbf{E}\left[\left|\widehat{\eta}^{\mathrm{DGKL}}_{Z_n}(X) - \eta(X)\right|\right] \to 0$, where $\widehat{\eta}^{\mathrm{DGKL}}_{Z_n}$ is defined as in (4).*

Plugging this into Theorem 2 we immediately obtain the following result on $\mathrm{AUC}(\widehat{\eta}^{\mathrm{DGKL}}_{Z_n})$.

Corollary 4 *Let $\widehat{\eta}^{\mathrm{DGKL}}_{Z_n}$ be defined as in (4). Then $\mathbf{E}_{Z_n}\left[\mathrm{Regret}(\widehat{\eta}^{\mathrm{DGKL}}_{Z_n})\right] \to 0$ if $\lim_{n\to\infty} k_n = \infty$ and $\lim_{n\to\infty} k_n/n = 0$, where $\widehat{\eta}^{\mathrm{DGKL}}_{Z_n}$ is defined as in (4).*

KNNOAM is thus guaranteed to converge in case of i.i.d. samples.

One can, in fact, derive results also for the rate of convergence based on the work by Chaudhuri et al. [8]. This would not hold uniformly though, only for some restricted distributions.

Algorithm 1 KNNOAM($\{k_t\}_t$)

```
Draw a random sample $U_1$ uniformly at random from $(0, 1)$
for round $t = 1, \ldots, n$ do
    Observe sample $(x_t, y_t)$
    Draw a random sample $U_{t+1}$ uniformly at random from $(0, 1)$
    $Z_t = (x_1, y_1, x_2, y_2, \ldots, x_t, y_t, U_1, \ldots, U_t, U_{t+1})$
    Construct hypotheses $h_t : \mathcal{X} \to \mathbb{R}$, mapping $x$ to $h_t(x) = \widehat{\eta}^{\mathrm{DGKL}}_{Z_t}(x)$  {As in (4)}
end for
```

Efficient Implementation
An important feature of $k-$NN-methods is that it can be implemented efficiently. For example, the Cover Tree structure [6] makes it possible to insert a new instance into an existing cover tree or to remove an old one from it in time $O(\log t)$, and also to find the k_n nearest neighbor of some arbitrary point in time $O(k_n \log t)$.

Choosing k
Choosing the right k for $k-$NN is a hard question. $k > \log\log n$ is recommended for pointwise convergence (see Remark 1 in 4), but the common practice is to use $logn < k < n^{1/2}$. One can also think about using k that changes with context; i.e., depends on the particular instance that is queried. See [4] for further details.

For a given dataset, one can also use cross validation or some Bayesian approach to find the best k. This was used by all the linear methods mentioned in the Introduction, but in a real online setting this is not applicable.

6 Dimensionality Reduction

In general, any learning method can be applied that approximates η with arbitrary accuracy. Note, however, that all these methods suffer from dimensionality issues; including k-NN. One way to deal with it is to apply first dimensionality reduction methods. More specifically, the idea is to first feed the obtained sample into some online PCA algorithm (like SGA or CCIPCA—see [7] for a thorough discussion), and then use its output as input for $k-$NN, Parzen-Rosenblatt kernels, etc. This way one maintains the good AUC performance guaranteed by the learning algorithms but prevents dimensionality-originated run-time issues thanks to the guarantees of the PCA methods.

The property this task requires from the aforementioned techniques to preserve the good AUC performance guarantees of $k-$NN is a kind of stability. More precisely, denoting by Φ_t the mapping they induce from the data from the first t rounds, it should fulfill the property

$$\|\Phi_t(x_t) - \Phi_T(x_t)\| \leq \epsilon(t) \qquad \forall\ T \geq t$$

for some $\epsilon(t)$ converging to 0 as t goes to infinity. Maintaining the convergence of $k-$NN does not seem possible otherwise.

7 Experimental Results

In this section, we evaluate the empirical performance of the proposed KNN Online AUC Maximization (KNNOAM) algorithm on benchmark datasets.

Compared Algorithms

We compare the proposed KNNOAM algorithm with state-of-the-art online AUC optimization algorithms. Specifically, the compared algorithms in our experiments include:

- **OAM**$_{\text{seq}}$: the OAM algorithm with reservoir sampling and sequential updating [22];
- **OAM**$_{\text{gra}}$: the OAM algorithm with reservoir sampling and online gradient updating method [22];
- **OPAUC**: the one-pass algorithm AUC optimization algorithm proposed in [14]
- **AdaOAM**: the adaptive gradient AUC optimization algorithm proposed in [13];
- **KNNOAM**: our proposed KNN based algorithm.

General Experimental Setup

We conduct our experiments on sixteen benchmark datasets that have been used in previous studies on AUC optimization. The details of the datasets are summarized in Table 1. All these datasets can be downloaded from LIBSVM [7] and UCI Machine Learning Repository [8]. Note that half of the datasets (segment, satimage, vowel, letter, poker, usps, connect-4, acoustic and vehicle) are originally multi-class. These multi-class datasets have been converted into class-imbalanced binary datasets by choosing one class, setting its label to +1 and the rest to -1. This class has been chosen so that the ratio T_-/T_+ is below 50

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

[8] www.ics.uci.edu/~mlearn/MLRepository.html

and its cardinality has been minimized. (The ratio is kept below 50 to obtain conclusive results.) In case two or more classes have the same size, one has been chosen randomly. Previous studies use similar conversion methods. In addition, the features have been rescaled linearly to [-1, 1] for all datasets. All experiments are performed with Matlab on a computer workstation with 3.40GHz CPU and 32GB memory.

Datasets	# instances	# features	T_-/T_+	Datasets	# instances	# features	T_-/T_+
fourclass	862	2	1.8078	segment	2310	19	6.0000
svmguide1	3089	4	1.8365	ijcnn1	141691	22	9.4453
magic04	19020	10	1.8439	connect-4	67557	126	9.4756
german	1000	24	2.3333	satimage	4435	36	9.6867
a9a	32561	123	3.1527	vowel	528	10	10.0000
svmguide3	1243	22	3.1993	usps	9298	256	12.1328
vehicle	846	18	3.2513	letter	20000	16	25.7380
acoustic	78823	50	3.3028	poker	25010	10	47.7524

Table 1. Details of the benchmark datasets used in the experiments. $T_+ = |\mathcal{X}^+|$ and $T_- = |\mathcal{X}^-|$.

Experimental Setup Of The Online Setting

Our main goal is to compare the performance of the above algorithms in the online setting. In this setting, each algorithm receives a random sample from the dataset, suffers loss and updates its classifier according to this sample. Note that our proposed KNNOAM does not require any parameter tuning, as opposed to the other four existing algorithms. Clearly, parameter tuning requires multiple passes over the data, which is inconsistent with this setting. Although KNNOAM must store at time t all the samples up to time $t-1$, the rest of the algorithms must store all of the samples in advance for the parameter tuning. Despite the fact that it gives the rest of the algorithms an unfair advantage, we follow the parameter tuning procedures for each algorithm suggested in [22, 14, 13] and use the best obtained parameters for each dataset and algorithm. We apply five-fold cross-validation on the training set to find the best learning rate $\eta \in 2^{[-10,10]}$ and the regularization parameter $\lambda \in 2^{[-10,2]}$ for both OPAUC and AdaOAM. For $\mathrm{OAM}_{\mathrm{seq}}$ and $\mathrm{OAM}_{\mathrm{gra}}$, we apply five-fold cross-validation to tune the penalty parameter $C \in 2^{[-10:10]}$, and fix the buffer at 100 as suggested in [22]. For KNNOAM we only had to choose k_n that goes to infinity and is of order $o(n)$. We choose $k_n = 2\log_2 n$. [9] For every dataset, we average over 20 runs and plot a graph showing the experimental AUC loss as a function of the number of samples. This gives us an opportunity to examine the evolution of the performance of the algorithms. The results of this experiment are presented in Appendix B. The code is available at `https://bitbucket.org/snir/auc2017`.

[9] This choice guarantees low time complexity and also turned out to result in a competitive method. (We did try other choices; the results were similar.)

Experimental Setup Of The One-Pass Setting
We also compare the performance of the algorithm in the one-pass setting. We follow the experimental setup suggested in previous studies [22, 14, 13]. The performance of the compared algorithms is evaluated by four trials of five-fold cross-validation using the parameters received by the parameter tuning procedure explained in the previous section about the online experimental setup. The AUC values are the average of these 20 runs. The results are summarized in Fig. 1.

7.1 Evaluation on Benchmark Datasets

In the online setting, KNNOAM outperforms the four other, state-of-the-art online AUC algorithms considered in our experiments in 12 out 16 datasets. What is more, in 10 out of those 12 datasets, the improvement by KNNOAM is significant. For example, in ijcnn1 and acoustic datasets, KNNOAM converges to a much lower empirical AUC loss than the rest of the algorithms. These are outstanding results for this setting, especially when recalling that the compared algorithms have been tuned before running the experiments. Our graphs demonstrate that KNNOAM has a much smoother convergence. The rest of the algorithms' performance is unstable: after receiving new samples it may improve but might also significantly deteriorate. This is a highly undesirable property.

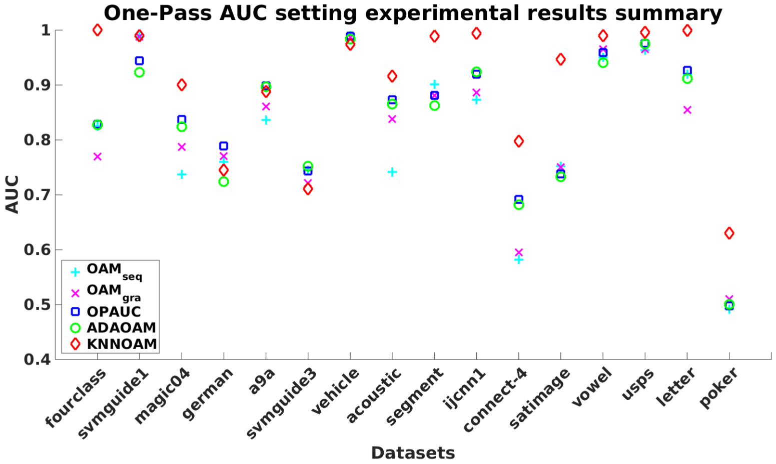


Fig. 1. Experimental results comparing the one-pass AUC performance summary

In the german, svmguide3 and vehicle datasets, KNNOAM does not outperform all the compared algorithms. However, these datasets are very small and therefore the results are inconclusive.

In the a9a dataset, OPAUC and AdaOAM outperform KNNOAM. Although this dataset cannot be considered small, KNNOAM still behaves as if it were, because the samples are sparse and the dimensionality is relatively big. To strengthen this claim, we can examine the performance of KNNOAM on connect-4 dataset. This dataset has roughly the same dimensionality as the a9a dataset and the samples are as sparse, but has more than twice the samples, and KNNOAM outperforms the rest of the algorithms significantly.

Many of the graphs show that $\mathrm{OAM}_{\mathrm{seq}}$, $\mathrm{OAM}_{\mathrm{gra}}$ are not guaranteed to converge at all. [10] Some of them also show that OPAUC and AdaOAM does not converge smoothly. This behavior is observed when the learning rate η is too high and a single sample could change the classifier drastically and cause a significant local deterioration of the performance as can be seen from the performance of OPAUC on svmguide1 dataset for example. On the other hand, if the chosen learning rate η is too small, the performance might be poor, as can be seen from the performance of AdaOAM on the same dataset.

The results for the one-pass setting are similar to the ones for the online setting: KNNOAM is better, or at least competitive in comparison with its competitors. For this setting we also present the running times (see Appendix C) KNNOAM is usually the fastest algorithm and is never the slowest among the compared algorithms.

8 Related Work

Reducing Ranking to Binary Classification

[3] and [5] considers the problem of ranking a finite random sample, and reduced this to a related binary classification problem. In particular, they bound the risk of the ranking (which is closely related to AUC of the ranker over this sample) in terms of the classification performance. However, the risk and the classification performance they use is only representative to that particular sample, and they do not consider prediction or generalization bounds. ([5] comment on the case when the rankings are drawn from some distribution, but does not imply any result in our setting.)

Offline Algorithmic Solutions

[9] aims to minimize (3), and actually obtain asymptotic optimality. Their algorithm constructs a tree in an iterative fashion by solving subsequent challenging optimization problems.

[19] uses kernel estimates of the conditional probability function $\eta(x) = \mathbf{E}[Y|X = x]$ based on the Parzen-Rosenblatt kernel, and have shown fast and superfast convergence rates assuming Tsybakov-style noise. They also complement their results by showing lower bounds on the best convergence rate in some situations.

Consistency of Surrogate Measures

The investigation of consistency with respect to AUC was initiated by [18] showing consistency of a balanced version of the exponential and the logistic loss. Later on [20, 1, 14, 15] investigated the consistency of other loss functions like the

[10] It should be mentioned that the theoretical gurantees of OAM is also doubtful, according to [17].

exponential, logistic, squared, and q-normed hinge loss, and variants of them. Finally, [15] shows that hinge loss is not consistent.

One-pass and online solutions

As mentioned, [22] was the first to analyze online AUC maximization. They have defined the setup, presented algorithmic solutions optimizing for the hinge loss, and provided regret bounds. [21] also uses the hinge loss with a perceptron-like algorithm which, in round t, achieves regret $O(1/\sqrt{t})$. [17] works with the same setting as [21], and achieves several improvements in terms of different parameters.

[14] uses square loss in the one-pass setting, and obtains a convergence rate of order $O(1/t)$ for the linearly separable case and $O(\sqrt{1/t})$ for the gereal one. [13] obtains similar results, but using the Adaptive Gradient method.

All these papers work with the set of linear hypothesis.

Uniform Convergence Bounds

Uniform convergence bounds like the ones in [2] show how fast the empirical AUC-risk converges to the actual AUC over a given class of hypothesis. Consequently, they do not provide any practical guidance on how one could acquire some hypothesis with small risk.

9 Concluding Remarks

We have shown that existing methodology for maximizing AUC in an online or one-pass setting can fail already in very simple situations. To remedy this, we have proposed to reach back to the fundaments of AUC, and suggested an algorithmic solution based on the celebrated $k-$NN-estimate of the conditional probability function. This has guarantees in the stochastic setting, has efficient implementations, and outperforms previous methods on several real datasets. The latter is even more surprising in view of the fact that, unlike its competitors, it requires no parameter tuning.

Nevertheless, we feel that this should not be considered as an ultimate solution, but rather as an encouragement for future research to explore further alternative solutions. To mention a few:

- Combining KNNOAM with metric learning arises naturally, and could extend its applicability to more exotic domains.
- Maximizing the objective function AUC_n in the adversarial setting is another important question, which existing results do not tell anything about.

Acknowledgements

This research was supported in part by the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement 306638 (SUPREL).

References

1. Agarwal, S.: Surrogate regret bounds for bipartite ranking via strongly proper losses. Journal of Machine Learning Research 15(1), 1653–1674 (2014)
2. Agarwal, S., Graepel, T., Herbrich, R., Har-Peled, S., Roth, D.: Generalization bounds for the area under the ROC curve. JMLR 6 (Apr 2005)
3. Ailon, N., Mohri, M.: An efficient reduction of ranking to classification. In: COLT 2008, Helsinki, Finland, July 9-12, 2008. pp. 87–98 (2008)
4. Anava, O., Levy, K.: k^*-nearest neighbors: From global to local. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 4916–4924. Curran Associates, Inc. (2016)
5. Balcan, M.F., Bansal, N., Beygelzimer, A., Coppersmith, D., Langford, J., Sorkin, G.B.: Robust reductions from ranking to classification. Machine Learning 72(1), 139–153 (2008)
6. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. pp. 97–104. ICML, ACM, New York, NY, USA (2006)
7. Cardot, H., Degras, D.: Online principal component analysis in high dimension: Which algorithm to choose? CoRR abs/1511.03688 (2015), `http://arxiv.org/abs/1511.03688`
8. Chaudhuri, K., Dasgupta, S.: Rates of convergence for nearest neighbor classification. In: NIPS, 2014. pp. 3437–3445 (2014)
9. Clémençon, S., Vayatis, N.: Tree-based ranking methods. Information Theory, IEEE Transactions on 55(9), 4316–4336 (Sept 2009)
10. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and empirical minimization of U-statistics. Ann. Statist. 36(2), 844–874 (04 2008)
11. Cortes, C., Mohri, M.: Auc optimization vs. error rate minimization. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) NIPS, pp. 313–320. MIT Press (2004)
12. Devroye, L., Győrfi, L., Krżyzak, A., Lugosi, G.: On the strong universal consistency of nearest neighbor regression function estimates. The Annals of Statistics 22(3), 1371–1385 (1994)
13. Ding, Y., Zhao, P., Hoi, S.C.H., Ong, Y.: An adaptive gradient method for online AUC maximization. In: AAAI. pp. 2568–2574 (2015)
14. Gao, W., Jin, R., Zhu, S., Zhou, Z.: One-pass AUC optimization. In: ICML 2013. pp. 906–914 (2013)
15. Gao, W., Zhou, Z.: On the consistency of AUC pairwise optimization. In: IJCAI, 2015. pp. 939–945 (2015)
16. Hanley, J.A., Mcneil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29–36 (1982)
17. Kar, P., Sriperumbudur, B.K., Jain, P., Karnick, H.: On the generalization ability of online learning algorithms for pairwise loss functions. In: 30th ICML 2013, Atlanta, GA, USA, 16-21 June 2013. pp. 441–449 (2013)
18. Kotlowski, W., Dembczynski, K., Hüllermeier, E.: Bipartite ranking through minimization of univariate loss. In: ICML. pp. 1113–1120. Omnipress (2011)
19. Robbiano, S., Clémençon, S.: Minimax learning rates for bipartite ranking and plug-in rules. In: ICML, 2011. pp. 441–448 (2011)
20. Uematsu, K., Lee, Y.: On theoretically optimal ranking functions in bipartite ranking. Tech. Rep. 863, Dept. of Stat., The Ohio State Univ. (December 2011)
21. Wang, Y., Khardon, R., Pechyony, D., Jones, R.: Generalization bounds for online learning algorithms with pairwise loss functions. In: COLT. pp. 13.1–13.22 (2012)
22. Zhao, P., Hoi, S.C.H., Jin, R., Yang, T.: Online auc maximization. In: ICML. pp. 233–240 (2011)

A Proof of Theorem 2

Let us first introduce the notation $\mathcal{X}_z = \{x : |\widehat{\eta}(x,z) - \eta(x)| < \sqrt{\epsilon}\}$ for $z \in \mathcal{X}$ and define for $h : \mathcal{X} \to \mathbb{R}$ measurable and $x, x' \in \mathcal{X}$

$$a(h,x,x') = \tfrac{1}{2}\mathbb{I}\left[h(x) = h(x')\right]\left[\eta(x)(1-\eta(x')\right] + \mathbb{I}\left[h(x) > h(x')\right]\left[\eta(x)(1-\eta(x')\right]$$

and its symmetrization

$$b(h,x,x') = \tfrac{1}{2}a(h,x,x') + \tfrac{1}{2}a(h,x',x) \enspace .$$

It then holds that

$$\begin{aligned}
\mathbf{E}_{(X,Y)}\mathbf{E}_{(X',Y')}\Big[&\tfrac{1}{2}\mathbb{I}\left[h(X) > h(X')\right]\mathbb{I}\left[Y=1, Y'=0\right] \\
&+ \mathbb{I}\left[h(X) > h(X')\right]\mathbb{I}\left[Y=1, Y'=0\right]\Big] \\
=&\mathbf{E}_{X,X'}\Bigg[\tfrac{1}{2}\mathbb{I}\left[h(X) = h(X')\right]\mathbf{E}_{Y,Y'}\left[\mathbb{I}\left[Y=1,Y'=0\right]\right] \\
&+ \mathbb{I}\left[h(X) > h(X')\right]\mathbf{E}_{Y,Y'}\left[\mathbb{I}\left[Y=1,Y'=0\right]\right]\Bigg] \\
=&\mathbf{E}_{X,X'}\left[a(h,X,X')\right] \\
=&\mathbf{E}_{X,X'}\left[b(h,X,X')\right] \enspace ,
\end{aligned}$$

where the last equation follows because X and X' are i.i.d. This then gives

$$\mathrm{AUC}(h) = \frac{\mathbf{E}_{X,X'}[b(h,X,X')]}{\mathbf{P}[Y=1]\mathbf{P}[Y=0]} \enspace . \tag{5}$$

Now, note that $x, x' \in \mathcal{X}_z$ implies $|\eta(x)(1-\eta(x')) - \eta(x')(1-\eta(x))| \le 2\sqrt{\epsilon}$ because of the $\alpha\beta - \alpha'\beta' = (\alpha-\alpha')\beta + \alpha'(\beta-\beta')$ equality. Combining this with the fact that $\mathbb{I}\left[h(x) = h(x')\right] + \mathbb{I}\left[h(x) < h(x')\right] + \mathbb{I}\left[h(x) > h(x')\right] = 1$ for any $h : \mathcal{X} \to \mathbb{R}$ and any $x, x' \in \mathcal{X}$, it follows that

$$b(\eta,x,x') - b(\widehat{\eta}_z,x,x')) \le \sqrt{\epsilon} \enspace , \quad z \in \mathcal{Z},\ x,x' \in \mathcal{X}_z \enspace .$$

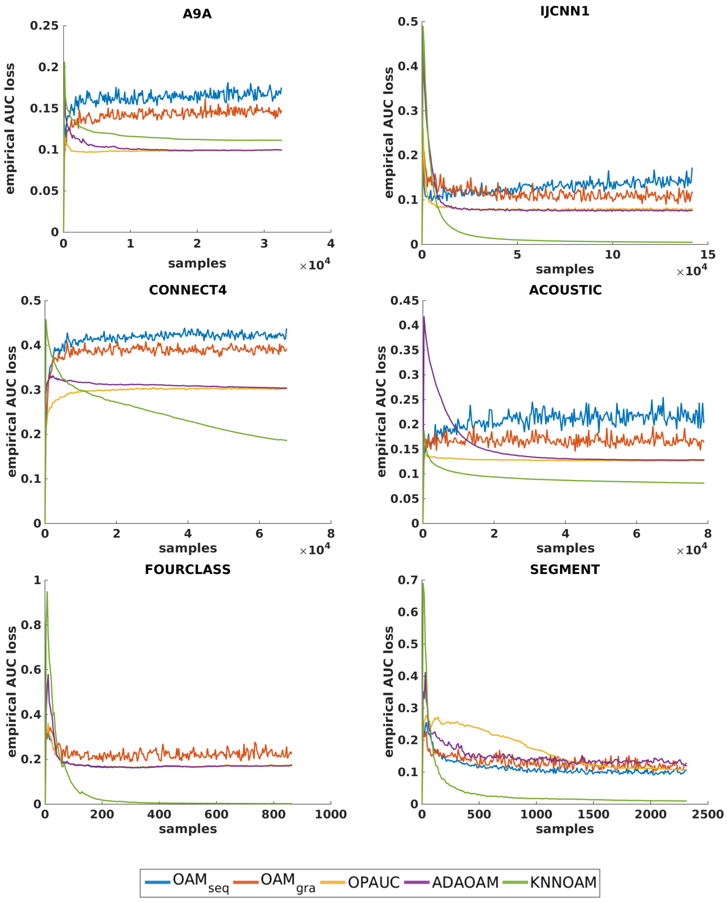
Accordingly,

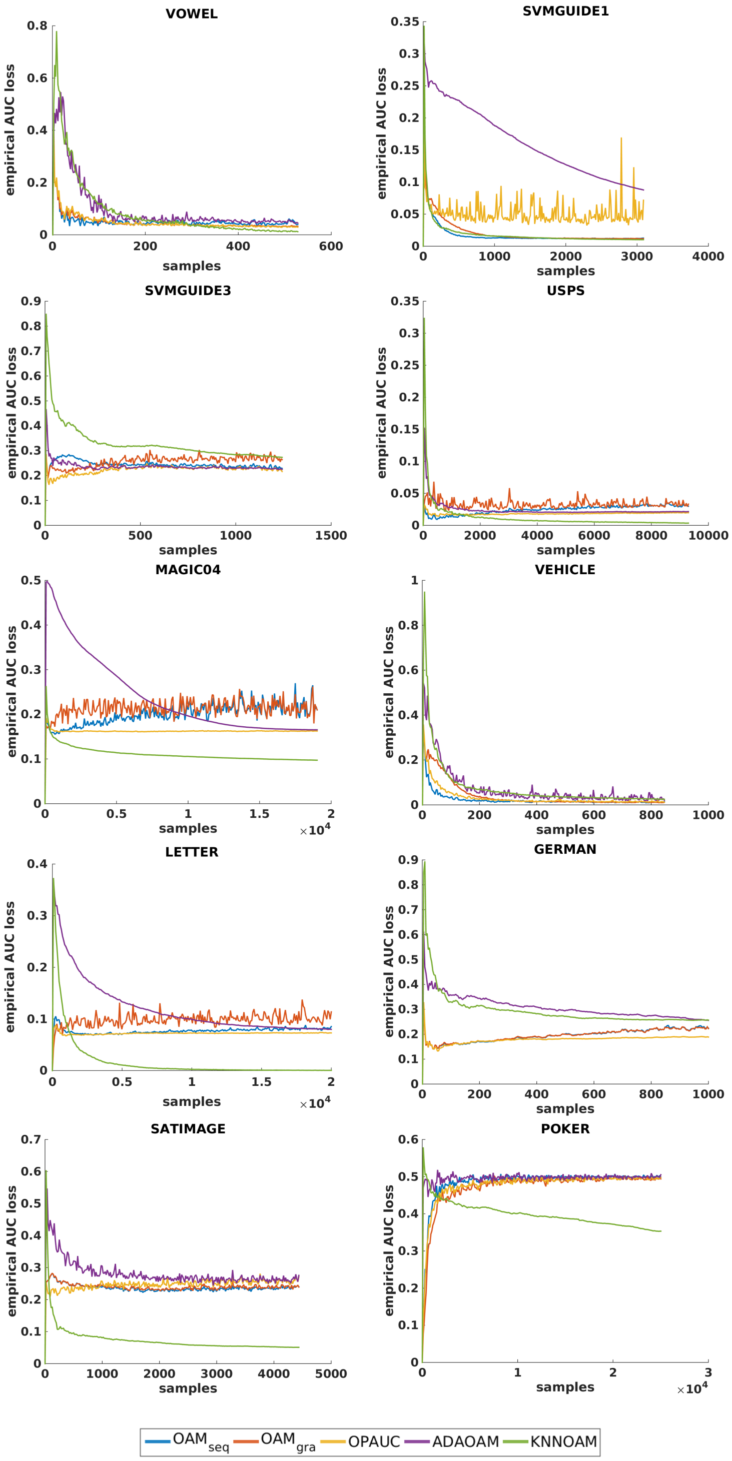
$$\begin{aligned}
\mathbf{E}_{X,X'}[b(\eta,X,X')] - \mathbf{E}_{X,X',Z}[b(\widehat{\eta}_Z,X,X')]& \\
&\le \sqrt{\epsilon} + \mathbf{P}_{X,X',Z}[X \notin \mathcal{X}_Z \text{ or } X' \notin \mathcal{X}_Z] \\
&\le 3\sqrt{\epsilon} \enspace ,
\end{aligned} \tag{6}$$

where the last inequality is true because X and X' are i.i.d. and because $\mathbf{P}_{X,Z}[X \notin \mathcal{X}_Z] = \mathbf{P}_{X,Z}[|\widehat{\eta}(X,Z) - \eta(X)| \ge \sqrt{\epsilon}] \le \sqrt{\epsilon}$ by the definition of ϵ.

Finally, according to [10], AUC(h) is maximized when $h = \eta$. The theorem thus follows by combining (5) and (6).

B Figures for Benchmark Datasets




VOWEL
empirical AUC loss
0
0.2
0.4
0.6
0.8
samples
0
200
400
600
SVMGUIDE1
empirical AUC loss
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
samples
0
1000
2000
3000
4000
SVMGUIDE3
empirical AUC loss
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
samples
0
500
1000
1500
USPS
empirical AUC loss
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
samples
0
2000
4000
6000
8000
10000
MAGIC04
empirical AUC loss
0
0.1
0.2
0.3
0.4
0.5
samples
0
0.5
1
1.5
2
$\times 10^4$
VEHICLE
empirical AUC loss
0
0.2
0.4
0.6
0.8
1
samples
0
200
400
600
800
1000
LETTER
empirical AUC loss
0
0.1
0.2
0.3
0.4
samples
0
0.5
1
1.5
2
$\times 10^4$
GERMAN
empirical AUC loss
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
samples
0
200
400
600
800
1000
SATIMAGE
empirical AUC loss
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
samples
0
1000
2000
3000
4000
5000
POKER
empirical AUC loss
0
0.1
0.2
0.3
0.4
0.5
0.6
samples
0
1
2
3
$\times 10^4$
OAM_{seq}
OAM_{gra}
OPAUC
ADAOAM
KNNOAM

C Experimental Results for the One-pass AUC Setup

Algorithm	fourclass		Algorithm	segment	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.8253 ± 0.0290	0.5125	OAM$_{\text{seq}}$	0.9010 ± 0.0205	1.4031
OAM$_{\text{gra}}$	0.7694 ± 0.0518	0.1797	OAM$_{\text{gra}}$	0.8811 ± 0.0262	0.4719
OPAUC	0.8281 ± 0.0308	0.0422	OPAUC	0.8808 ± 0.0237	0.1289
AdaOAM	0.8278 ± 0.0269	0.0641	AdaOAM	0.8629 ± 0.0337	0.1922
KNNOAM	**1.0000 ± 0.0000**	**0.0070**	KNNOAM	**0.9891 ± 0.0087**	**0.0227**
Algorithm	svmguide1		Algorithm	ijcnn1	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.9885 ± 0.0026	2.0953	OAM$_{\text{seq}}$	0.8733 ± 0.0253	101.9695
OAM$_{\text{gra}}$	0.9884 ± 0.0039	0.6922	OAM$_{\text{gra}}$	0.8862 ± 0.0345	34.1859
OPAUC	0.9440 ± 0.0175	0.1516	OPAUC	0.9201 ± 0.0080	**10.3078**
AdaOAM	0.9229 ± 0.0338	0.2336	AdaOAM	0.9237 ± 0.0036	14.2531
KNNOAM	**0.9900 ± 0.0041**	**0.0156**	KNNOAM	**0.9945 ± 0.0005**	66.9656
Algorithm	magic04		Algorithm	connect-4	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.7370 ± 0.0910	13.4914	OAM$_{\text{seq}}$	0.5817 ± 0.0205	50.7984
OAM$_{\text{gra}}$	0.7875 ± 0.0530	4.6008	OAM$_{\text{gra}}$	0.5948 ± 0.0232	**17.4992**
OPAUC	0.8372 ± 0.0063	1.1969	OPAUC	0.6918 ± 0.0077	117.2625
AdaOAM	0.8240 ± 0.0085	1.7000	AdaOAM	0.6824 ± 0.0166	144.8133
KNNOAM	**0.9003 ± 0.0061**	**0.3875**	KNNOAM	**0.7977 ± 0.0053**	77.9141
Algorithm	german		Algorithm	satimage	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.7603 ± 0.0395	0.6086	OAM$_{\text{seq}}$	0.7520 ± 0.0250	2.9531
OAM$_{\text{gra}}$	0.7704 ± 0.0368	0.2172	OAM$_{\text{gra}}$	0.7504 ± 0.0262	1.1398
OPAUC	**0.7890 ± 0.0274**	0.0578	OPAUC	0.7380 ± 0.0166	0.4508
AdaOAM	0.7238 ± 0.0691	0.0859	AdaOAM	0.7328 ± 0.0267	0.5719
KNNOAM	0.7454 ± 0.0268	**0.0086**	KNNOAM	**0.9466 ± 0.0108**	**0.1172**
Algorithm	a9a		Algorithm	vowel	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.8368 ± 0.0199	25.3867	OAM$_{\text{seq}}$	0.9503 ± 0.0329	0.1195
OAM$_{\text{gra}}$	0.8611 ± 0.0108	**8.3141**	OAM$_{\text{gra}}$	0.9654 ± 0.0157	0.0547
OPAUC	**0.8989 ± 0.0036**	54.7289	OPAUC	0.9592 ± 0.0245	0.0273
AdaOAM	0.8967 ± 0.0036	67.2898	AdaOAM	0.9410 ± 0.0290	0.0422
KNNOAM	0.8879 ± 0.0040	17.4187	KNNOAM	**0.9895 ± 0.0092**	**0.0047**
Algorithm	svmguide3		Algorithm	usps	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.7453 ± 0.0410	0.7477	OAM$_{\text{seq}}$	0.9634 ± 0.0121	7.3531
OAM$_{\text{gra}}$	0.7217 ± 0.0428	0.2703	OAM$_{\text{gra}}$	0.9650 ± 0.0097	**2.3188**
OPAUC	0.7433 ± 0.0384	0.0703	OPAUC	0.9759 ± 0.0066	35.9055
AdaOAM	**0.7519 ± 0.0442**	0.1047	AdaOAM	0.9752 ± 0.0081	43.8656
KNNOAM	0.7112 ± 0.0339	**0.0102**	KNNOAM	**0.9957 ± 0.0029**	3.0055
Algorithm	vehicle		Algorithm	letter	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	**0.9898 ± 0.0067**	0.4680	OAM$_{\text{seq}}$	0.9200 ± 0.0088	13.1813
OAM$_{\text{gra}}$	0.9872 ± 0.0084	0.1617	OAM$_{\text{gra}}$	0.8548 ± 0.0554	4.3805
OPAUC	0.9886 ± 0.0060	0.0469	OPAUC	0.9267 ± 0.0062	1.2367
AdaOAM	0.9838 ± 0.0101	0.0711	AdaOAM	0.9120 ± 0.0315	1.7828
KNNOAM	0.9745 ± 0.0074	**0.0063**	KNNOAM	**0.9993 ± 0.0003**	**1.0063**
Algorithm	acoustic		Algorithm	poker	
	AUC	Time(s)		AUC	Time(s)
OAM$_{\text{seq}}$	0.7421 ± 0.0587	58.3867	OAM$_{\text{seq}}$	0.4916 ± 0.0266	15.6133
OAM$_{\text{gra}}$	0.8379 ± 0.0277	19.5711	OAM$_{\text{gra}}$	0.5099 ± 0.0318	5.6508
OPAUC	0.8729 ± 0.0036	**8.0313**	OPAUC	0.4982 ± 0.0368	1.4664
AdaOAM	0.8653 ± 0.0120	10.4055	AdaOAM	0.5001 ± 0.0321	2.1391
KNNOAM	**0.9166 ± 0.0023**	42.7477	KNNOAM	**0.6299 ± 0.0309**	**1.1422**

Table 2. Experimental results comparing the one-pass AUC performance (i.e., AUC(h_n)) of OAM$_{\text{seq}}$, OAM$_{\text{gra}}$, OPAUC, ADAOM and KNNOAM