Proximal Stochastic AUC Maximization

Majdi Khalid
Computer Science Department
Umm Al-Qura University
Makkah, KSA
mknfiai@uqu.edu.sa

Hamidreza Chitsaz
Computer Science Department
Colorado State University
Fort Collins, USA
chitsaz@chitsazlab.org

Indrakshi Ray
Computer Science Department
Colorado State University
Fort Collins, USA
Indrakshi.Ray@colostate.edu

***Abstract*—This work considers a stochastic optimization problem for maximizing the AUC (area under the ROC curve). The AUC metric has proven to be a reliable performance measure for evaluating a model learned on imbalanced data. The batch pairwise learning methods (e.g., rankSVM) can achieve a quadratic convergence to the optimal solution. However, the batch learning paradigm hinders the scalability of these methods. Recently different online and stochastic AUC maximization algorithms are developed. While these can scale well for large-scale data, they either cannot generalize as good as the batch AUC methods or suffer from slow convergence, which minimizes their scalability. A recent stochastic pairwise learning algorithm for AUC maximization suggests to schedule both the regularization and the averaging steps to improve the generalization capability and the convergence speed. Building on this algorithm, we develop a simple proximal stochastic AUC maximization algorithm. The proposed algorithm uses a proximal operator of the pairwise hinge loss function, which encourages small update steps. Averaging these adjacent weights has a significant improvement on the converges rate of the final model. Experiments on several benchmark data sets show that the proposed algorithm can achieve AUC classification accuracy on par with that of the batch method while being considerably efficient. The proposed algorithm also outperforms state-of-the-art online and stochastic algorithms in terms of generalization performance and convergence rate.**

***Index Terms*—AUC maximization, Imbalanced learning, stochastic gradient**

I. Introduction

The area under the ROC Curve (AUC) [12] is a measure of interest in a wide range of machine learning and data mining applications such as information retrieval, bioinformatics, and anomaly detection [1], [19], [22], [24], [27]. The interest in using the AUC as an evaluation measure stems from its reliability in evaluating a classifier trained on a heavily imbalanced dataset [7]. Unlike accuracy, the AUC measures the classification capacity of a classifier over all possible thresholds. What an AUC score denotes is the probability of scoring a random positive instance higher than a randomly drawn negative instance.

The objective function for maximizing the AUC optimizes a sum of pairwise loss functions. Therefore, any bipartite ranking algorithm can be utilized to optimize the AUC measure directly. The batch ranking algorithms [6], [17] have a powerful generalization capability as they are able to reach the optimal solution. However, the computational complexity of each iteration of the batch algorithms is linear in the data size $\mathcal{O}(nd)$ at best, where n is the number of instances and d is the dimension of the data. Though the number of iterations could be small due to the quadratic convergence, the complexity of each iteration compromises the scalability of the batch algorithms when applied to sizable datasets.

Online learning [4], [8] and Stochastic gradient methods [3], [23], [25] are appealing learning paradigm for large-scale setting. However, the first-order online and stochastic methods suffer from suboptimal convergence, while second-order methods have an expensive updating step, which hinders their applicability for high dimensional data.

For AUC maximization, several first and second-order online and stochastic algorithms are devised to optimize the AUC objective functions [11], [28], [30]. However, these AUC maximization algorithms are either prone to a suboptimal solution or require a large number of iterations to yield a high AUC classification accuracy.

Recently, a few methods [14], [18], [20] are designed to improve the convergence rate of some of the precedent AUC maximization algorithms. The confidence-weighted AUC maximization method [14] attempts to improve the convergence of the first-order method by exploiting the second-order information. The work [20] applies a proximal operator to the gradient of the primal variable to improve the rate of convergence of the original algorithm [28]. An adaptive variant of [28] is proposed by [18] to achieve a faster convergence rate. However, the faster convergence rate obtained by these algorithms either is at the expense of the per iteration complexity, or it does not enhance the generalization capability of the original algorithm. Therefore, the development of a scalable algorithm for AUC maximization that is able to achieve a better generalization and a faster convergence rate is still a challenging problem.

In this work, we propose a proximal stochastic AUC maximization algorithm, which we abbreviate to PSAM, to promote the convergence rate of the accelerated stochastic AUC maximization (ASAM) [15]. The proposed algorithm PSAM borrows the acceleration techniques of ASAM and applying it to the weight vector updated by the proximal operator of a pairwise hinge loss function. In a nutshell, the contributions of our work are (I) We design a proximal stochastic algorithm for AUC maximization. (II) We experiment on several data sets to prove the superiority of the proposed algorithm PSAM over the state-of-the-art stochastic AUC maximization algorithms. (III) We also evaluate the proposed algorithms on approximate

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

nonlinear space computed via k-means Nyström methods, which improve the generalization capability of almost any learning algorithm. The results show that PSAM achieves comparable generalization capability if not better than the batch AUC maximization method while its training time is significantly faster than that of the batch method. Besides, the proposed algorithm PSAM is able to converge faster than the other competing methods from the first epoch, where each epoch consists of n iterations.

II. Related Work

In AUC maximization literature, different learning approaches have been considered to optimize the AUC objective function efficiently. In what follow, we divide the AUC maximization methods based on their learning paradigm (i.e., batch, online, stochastic) and briefly review most recent works in each category.

Batch Methods. Most ranking algorithms can be employed to solve the AUC maximization problem. Several linear and nonlinear RankSVM methods have been developed to speed up the training time of optimizing a pairwise loss function [2], [5], [6], [13], [16], [17]. However, these methods cannot scale well for large-scale datasets.

Online and Stochastic Methods. Unlike batch methods, online AUC maxmization methods [14], [30] makes a single pass over the training instances while using a buffering scheme (i.e., reservoir sampling or first-in-first-out) to deal with the pairwise nature of the loss function. The first-order online AUC maximization algorithm [30] has per iteration complexity of $\mathcal{O}(Bd)$, while the second-order [14] has per iteration complexity of $\mathcal{O}(Bd^2)$, where d is the dimension of the data and B is the size of the buffer. To circumvent the need for storing some historical instances when updating the model, the work [11] maintains covariance matrices for storing the second-order information of each class, which require $\mathcal{O}(d^2)$ operation for updating the model. Recently, the work [28] proposes a stochastic online AUC maximization algorithm that optimizes a pairwise least square loss function as a min-max saddle point problem. While the AUC algorithm suggested by [28] has a pure online complexity $\mathcal{O}(d)$, it suffer from slow rate of convergence. Based on the saddle point formulation for the AUC maximization, the work [20] proposes a proximal stochastic AUC maximization algorithm, which can be applied to a non-smooth regularizer. An adaptive variant of the saddle point formulation for AUC maximization is suggested by [18] to improve the convergence rate. The authors of the two precedent algorithms [18], [20] claim a convergence rate of $\mathcal{O}(\frac{1}{n})$ up to a logarithmic factor, where n is the number of training instances. Based on classical stochastic learming, Khalid et al., [15] propose an accelerated stochastic method for AUC maximization and achieve a better rate of convergence and generalization capability compared to previous methods. However, existing online and stochastic AUC maximization algorithms experience suboptimal generalization capability compared to batch methods.

III. Proposed Algorithm

A. Preliminaries

Given a sequence of training instances $(x_1, y_1), \ldots, (x_n, y_n)$ independently drawn from unknown distribution $\mathcal{D}$ on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ represents an instance with d dimensional features and $y \in \{1, -1\}$ represents the label. Let $h(x) = (w^T x)$ denotes a linear classifier, then the AUC score is defined as:

$$AUC(w) = Pr(h(x^+) \geq h(x^-)) = \mathbb{E}[\mathbb{I}_{h(x^+) \geq h(x^-)}],$$

where $\mathbb{I}(\cdot)$ is an indicator function. In practice, the indicator function, which is discontinuous, is replaced by a convex surrogate loss function. In our algorithm, we define the AUC loss function using the hinge loss, which has the form $\ell(w) = max\{0, 1 - w^T z\}$, where $z = (x^+ - x^-)$. The derivative of the hinge loss can be written as: $\ell'(w)z$, where the subgradient ℓ' is defined as follows:

$$\ell'(w) = \begin{cases} -1 & 1 - w^T z \geq 1 \\ 0 & 1 - w^T z \leq 0 \\ w^T z - 1 & \text{otherwise} \end{cases}.$$

The optimization problem for maximizing the AUC objective function is defined as:

$$\min_{w \in \mathbb{R}^d} F(w) \triangleq \frac{1}{n^+ n^-} \sum_{i=1}^{n^+} \sum_{j=1}^{n^-} f(w) + \lambda \psi(w), \qquad (1)$$

where $f(w)$ is a convex differentiable function and $\psi(w)$ is a convex regularizer, which could be non-differentiable. The popular and scalable approach to solve such an optimization problem is stochastic gradient descent (SGD) [23], which enjoys a low per-iteration complexity. However, the convergence rate of the vanilla SGD is slower than that of the gradient method. The accelerated stochastic AUC maximization algorithm [15] improves the convergence rate by combining both the scheduled regularization and the scheduled iterate averaging techniques.

In some cases with complex datasets, this accelerated algorithm [15] turns out to be inefficient in terms of iteration complexity, meaning a large number of iterations is required to achieve an AUC performance comparable to the batch AUC method. How to make a first-order SGD converges to the optimal solution from the first few iterations is a challenging problem.

B. Proximal AUC Maximization

In this work we promote the convergence rate of the accelerated stochastic AUC maximization [15] using the proximal mapping of the hinge loss function. The minimization of the proximal variant of the objective function 1 using a stochastic algorithm comprises of drawing a random positive and negative instance at each iteration and compute the model as,

$$w_{t+1} = w_t - \frac{1}{(t+t_0)} \, M \, \text{prox}_{\lambda t}(w_t),$$

where the rescaling matrix M is defined as $M = \lambda^{-1} I$ when updating the weight vector using only the first-order information. The proposed algorithm is detailed in Algorithm 1. The main step in our algorithm is the use of the proximal mapping of the pairwise hinge loss function. The operator of the proximal mapping of $f(w_t)$ is defined as:

$$\text{prox}_{\lambda t}(w) = \underset{v \in \mathbb{R}^d}{\text{argmin}} \Big\{ \lambda f_t(v) + \frac{1}{2} ||v - w||^2 \Big\}. \qquad (2)$$

The solution of the proximal operator 2 can be derived analytically using its optimality condition. The derivation steps are detailed in Appendix A. The proposed proximal algorithm applies the scheduled regularization and averaging steps [15] to the weights of the model to speed up the convergence. These two steps are regulated to be performed each $rskip$ and $askip$ iterations respectively as follows,

$$w_{t+1} = w_{t+1} - rskip(t+t_0)^{-1} w_{t+1}$$

$$\tilde{w}_{q+1} = \frac{q\tilde{w}_q + w_{t+1}}{q \, + \, 1},$$

where $\tilde{w}$ is the averaged solution after q iterations with respect to the $askip$.

To show the difference between the proximal stochastic and the standard stochastic gradient methods for AUC maximization, we can rewrite the update step of our proximal algorithm as:

$$w_{t+1} = w_t - \frac{1}{\lambda(t+t_0)} \, g(w_{t+1}),$$

whereas the update step of the vanilla stochastic algorithm is written as:

$$w_{t+1} = w_t - \frac{1}{\lambda(t+t_0)} \, g(w_t).$$

We can see that the proximal stochastic algorithm can evaluate the hinge loss function at w_{t+1} instead of w_t without making an actual iteration. Moreover, the proximal operator makes small update steps during training iterations. In contrast to the accelerated stochastic method [15], our proximal algorithm averages a set of adjacent weights. The averaging of such weights yields an improvement in accelerating the convergence rate.

C. Comparison to Other Proximal Methods

Among several proximal stochastic methods [9], [20], [21], [26], the proposed algorithm PSAM is similar to the proximal stochastic AUC maximization (SPAM) [20] in terms of employing the proximal operator for AUC maximization. However, the proximal operator used in [20] is applied to the regularization term, while we apply the proximal operator to the hinge loss function. Also, our proposed method invites comparison with Point-SAGA [9]. However, Point-SAGA is designed to maximize accuracy, whereas our algorithm is devised to maximize AUC measure. Besides, Point-SAGA uses the proximal operator of the hinge loss within a variance reduction framework, while we employ this proximal operator in a different acceleration framework, which comprises of scheduling the regularization and averaging steps. Notice that our method can easily be used for maximizing the accuracy as well. The comparison with Point-SAGA in terms of accelerating the rate of convergence for maximizing the accuracy metric is beyond the scope of this work.

Algorithm 1: Proximal Stochastic AUC Maximization

```
Input: training dataset $X$, $\gamma$,$t_0$,$T$, $rskip$, $askip$
Set $rcount = rskip$, $acount = askip$, $q = 0$
Initialize $w_1 = 0 \in \mathbf{R}^d$ and $\tilde{w}_0 = 0 \in \mathbf{R}^d$
for $t = 1, \ldots, T$ do
    Randomly pick a pair $i_t \in 1, \ldots, n^+, j_t \in 1, \ldots, n^-$
    $x_t = x_{i_t} - x_{j_t}$
    $\lambda_t = \frac{1}{\gamma(t+t_0)}$
    $w_{t+1} = \text{prox}_{\lambda_t f_t}(w_t)$
    $rcount = rcount - 1$
    if $rcount \leq 0$ then
        $w_{t+1} = w_{t+1} - rskip \, (t+t_0)^{-1} \, w_{t+1}$
        $rcount = rskip$
    end if
    $acount = acount - 1$
    if $acount \leq 0$ then
        $\tilde{w}_{q+1} = \frac{q\tilde{w}_q + w_{t+1}}{q \, + \, 1}$
        $q = q + 1$
        $acount = askip$
    end if
end for
set $w = \tilde{w}_q$
return $w$
```

IV. Experiments

In this section, we evaluate the performance of our proposed method on several benchmark datasets. We compare our proximal stochastic AUC maximization algorithm with the state-of-the-art online and stochastic AUC maximization algorithms. The experiments are implemented in MATLAB, while the learning algorithms are written in C++ language via MEX files. The experiments were performed on a computer equipped with an Intel 4GHz processor with 32G RAM.

A. Benchmark Datasets

We use several datasets described in Table I. The datasets can be downloaded from LibSVM website[1] and UCI[2]. For datasets that are not split into training and test sets, we partition them into 80% for training and 20% for testing. The multi-class data are transformed into imbalanced binary data

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

[2]http://archive.ics.uci.edu/ml/index.php

by grouping roughly half of the classes into a label and the rest of classes into a different label.

TABLE I
BENCHMARK DATASETS

Data	#training	#test	#feat
spambase	3,680	921	57
a9a	26,048	6,513	123
ijcnn1	49,990	91,701	22
connect-4	54,045	13,512	126
mnist	60,000	10,000	784
acoustic	78,823	19,705	50
aloi	86,400	21,600	128
webspam	280,000	70,000	254
cod-rna	331,152	157,413	8
epsilon	400,000	100,000	2000
covtype	464,809	116,203	54
susy	4,500,000	500,000	18
farm-ads	3,314	829	54,877
sector	6,412	3,207	55,197
news20	12,748	3,187	62,061

B. Compared Methods and Model Selection

1) **OAM$_{\text{seq}}$ and OAM$_{\text{gra}}$** [30]: The sequential and gradient variants of online AUC maximization. The hyperparameters are chosen as suggested by [30] via 3-fold cross validation. The number of positive and negative buffers is set to 100.
2) **CBR$_{\text{FIFO}}$** [14]: The confidence-weighted bipartite ranking algorithms with the First-In-First-Out buffer updating policy. The size of the positive and negative buffers is fixed at 50. The hyperparameter η is set to 0.7, and the penalty hyperparameter C is tuned by 3-fold cross validation by searching in $2^{[-10:10]}$. We use the diagonal variant when experimenting on the high dimensional datasets.
3) **SOLAM** [28]: This is the stochastic online AUC maximization. The hyperparameters of the algorithm (i.e., the learning rate and the bound on the weight vector) are selected via 3-fold cross validation by searching in the grids $\{1 : 9 : 100\}$ and $\{10^{-1}, \ldots, 10^{5}\}$, respectively. The number of iterations is set to 15.
4) **BAM** [6]: This is the batch AUC maximization algorithm. This algorithm optimizes the squared hinge loss function using truncated Newton. The best regularization hyper-parameter C is chosen from the grid $\{2^{-15}, \ldots, 2^{10}\}$ via 3-fold cross validation.
5) **ASAM** [15]: This is the accelerated stochastic AUC maximization algorithm. The hyper-parameter λ is chosen from the grid $\{10^{-10}, \ldots, 10^{-7}\}$ via 3-fold cross validation. For the experiment with high dimensional data, we tune λ using 3-fold cross validation by searching in the grid $\{1 : 9 : 100\}$.
6) **PSAM**: This is the proposed proximal stochastic AUC maximization algorithm. The hyper-parameter λ is chosen from the grid $\{10^{-10}, \ldots, 10^{-7}\}$ via 3-fold cross validation. For the experiment with high dimensional data, we tune λ using 3-fold cross validation by searching in the grid $\{1 : 9 : 100\}$.

C. Results and Discussion

Results for Linear AUC Maximization Methods
The comparison in terms of AUC performance and training time on the benchmark datasets is shown in Table II. The reported AUC results is the average of 5 runs.

We observe that our algorithm PSAM outperforms the other online and stochastic methods in terms of AUC classification accuracy. Further, we see that the AUC performance of PSAM is comparable to the batch method, whereas the training of PSAM is faster than the batch method. PSAM is also able to achieve better AUC classification accuracy compared to its non-proximal counterpart ASAM, while its training time is on par with that of ASAM. CBR$_{\text{FIFO}}$ achieves a robust AUC performance on most datasets. However, its training is significantly slower than the other stochastic and online algorithms on most datasets, especially for datasets with a large number of features.

Results for Nonlinear AUC Maximization Methods
We compare the performance of the nonlinear variant of our proximal method with the other nonlinear batch and stochastic AUC maximization algorithms on six datasets (i.e., acoustic, aloi, cod-rna, webspam, covtype, and susy). The results comparing the performance are shown in Table III. We use the k-means Nyström method [29] to approximate the Gaussian kernel matrix. The bandwidth of the Gaussian function is set to the average square distance between the first 80k instances and the mean, which is computed over these instances. We set the number of landmark points to be 1600 for acoustic, aloi, cod-rna, webspam, and covtype, while susy has landmark points set to 400. The results of the stochastic methods are averaged over 3 runs.

We can see that our method NPSAM achieves a robust performance compared to NSOLAM and NASAM, while its AUC performance is on par with that of the batch method NBAM. However, the training time of our method NPSAM is significantly faster than NBAM. Among the stochastic algorithms, NSOLAM performs poorly compared to our method. The robust performance and the fast training of our proximal algorithm make it appealing for large-scale applications.

D. Study on the Convergence Rate

We study the convergence of PSAM with respect to the number of epochs. We also compare it with the other stochastic AUC maximization methods ASAM and SOLAM. The AUC results of these stochastic methods upon varying the number of epochs are depicted in Figure 1. We vary the number of epochs according to the grid $\{1, 2, 3, 4, 5, 10, 30, 60\}$, and run the stochastic algorithms using the same setup described in the preceding subsection. One epoch means n number of iterations, where n is the number of instances. For PSAM and ASAM, we pick a positive and negative instance at random in each iteration. In all subfigures, the x-axis represents the number of epochs, while the y-axis is the AUC classification accuracy averaged over 3 runs on the test set.

We observe that PSAM and its nonlinear variant NPSAM are able to reach the optimal solution from the first epoch in most datasets. We attribute this superior performance of our algorithms to the formulation of the proximal operator with scheduling both the regularization and the averaging steps. We also note that increasing the number of epochs improve the AUC performance of PSAM and NPSAM on some datasets. Notice that the number of iterations in the first epoch is much smaller than the number of pairs. This suggests that studying different sampling strategies is a possible research direction to boost the rate of convergence.

V. Conclusion and Future Work

In this paper, we have proposed scalable batch and stochastic nonlinear AUC maximization algorithms. The proposed algorithms optimize linear classifiers on a finite-dimensional feature space constructed via the k-means Nyström approximation. We solve the proposed batch AUC maximization algorithm using truncated Newton optimization, which minimizes the pairwise squared hinge loss function. The proposed stochastic AUC maximization algorithm is solved using a first-order gradient descent that implements scheduled regularization update and scheduled averaging to accelerate the convergence of the classifier. We show via experiments on several benchmark datasets that the proposed AUC maximization algorithms are more efficient than the nonlinear kernel AUC machines, while their AUC performances are comparable or even better than the nonlinear kernel AUC machines. Moreover, we show experimentally that the proposed stochastic AUC maximization algorithm outperforms the state-of-the-art online AUC maximization methods in terms of AUC classification accuracy with a marginal increase in the training time for some datasets. We demonstrate empirically that the proposed stochastic AUC algorithm converges to the optimal solution in a few epochs, while other online AUC maximization algorithms are susceptible to suboptimal convergence. In the future, we plan to use the proposed algorithms in solving large-scale multiple-instance learning. Also, we will study the application of the proposed algorithm to solving non-convex cost function.

Appendix

The Proximal Operator of the Hinge Loss: Proximal operators for most loss functions have efficient or closed form solutions. In what follows, we derive the analytical solution for the proximal operator of the pairwise hinge loss function, which is similar to the solution presented in [10]. Let x_i^+ and x_j^- represent a random positive and negative instance, respectively. We assume that $x \in \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathbb{R}$ is a Euclidean space, meaning that the magnitude of any vector in $\mathbb{R}$ is obtained by l_2-norm. Let $x_t = (x_i - x_j)$ denotes the difference vector at the t-th iteration. The pairwise hinge loss function is defined as:

$$f(w_t) = max\{0, 1 - w_t^T x_t\}.$$

The proximal mapping of $f(w_t)$ is achieved by the optimality condition that implies the following minimization problem:

$$\text{prox}_{\lambda t}(w) = \underset{v \in \mathbb{R}^d}{\text{argmin}} \Big\{ \lambda f_t(v) + \frac{1}{2}||v - w||^2 \Big\}.$$

The precedent expression is a minimization problem that approximates to the vector v while taking into account the cost of this approximation $f(v)$. A closed form solution of this minimization problem can be attained by the optimality condition of the proximal operator. Recall that the gradient of the hinge loss function is defined as $f^{'}(w_t)x_t$, where $f^{'}$ is the subgradient of f, and the evaluation of the gradient is determined based on the projection onto the hyperplane or half-spaces. Consequently, the proximal operator of the hinge loss function can be redefined as an orthogonal projection [**?**]

$$\text{prox}_{\lambda t}(w) = \underset{v \in \mathbb{R}^d}{\text{argmin}} \Big\{ \lambda f_t(v) + \frac{1}{2}||v - w||^2 \Big\} = P_{\lambda H}(w),$$

where $H = \{v \in \mathbb{R}^d : v^T x = 1\}$. We can derive an explicit form for the problem of finding $P_H(w)$ as follows:

$$\begin{aligned} &\underset{v \in \mathbb{R}^d}{\text{argmin}} \, ||v - w||^2 \\ &s.t. \quad v^T x = 1 \ . \end{aligned}$$

The precedent constrained minimization problem can be solved using its optimality condition (i.e., KKT conditions), which yields the following solution:

$$P_H(w) = w - \frac{w^T x - 1}{||x||^2} x.$$

Therefore, the proximal operator of the hinge loss has the following closed-form solution,

$$\text{prox}(w) = w - \lambda g x_t,$$

where:

$$z = \frac{1 - w^T x_t}{\lambda ||x_t||^2}.$$

$$g = \begin{cases} 0 & \text{z} \ \leq \ 0 \\ -1 & z \ \geq \ 1 \\ -z & 0 < z < 1 \end{cases} .$$

References

[1] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth, *Generalization bounds for the area under the roc curve*, Journal of Machine Learning Research, 6 (2005), pp. 393–425.

[2] A. Airola, T. Pahikkala, and T. Salakoski, *Training linear ranking svms in linearithmic time using red–black trees*, Pattern Recognition Letters, 32 (2011), pp. 1328–1336.

[3] A. Bordes, L. Bottou, and P. Gallinari, *Sgd-qn: Careful quasi-newton stochastic gradient descent*, Journal of Machine Learning Research, 10 (2009), pp. 1737–1754.

[4] L. Bottou and Y. L. Cun, *Large scale online learning*, in Advances in neural information processing systems, 2004, pp. 217–224.

[5] T. Calders and S. Jaroszewicz, *Efficient auc optimization for classification*, in European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2007, pp. 42–53.

[6] O. Chapelle and S. S. Keerthi, *Efficient algorithms for ranking with svms*, Information Retrieval, 13 (2010), pp. 201–215.

TABLE II
AUC CLASSIFICATION ACCURACY (%) AND TRAINING TIME (IN SECONDS) FOR DIFFERENT AUC MAXIMIZATION ALGORITHMS.

Algorithm	spambase		ijcnn1		a9a	
	AUC	Training time	AUC	Training time	AUC	Training time
OAM_{seq}	96.236 ± 0.473	0.018	87.498 ± 1.282	0.113	81.565 ± 0.869	0.286
OAM_{gra}	95.995 ± 0.913	0.017	86.617 ± 1.850	0.113	81.456 ± 1.878	0.282
CBR_{FIFO}	97.573 ± 0.093	0.533	91.591 ± 0.048	2.224	89.900 ± 0.013	17.25
SOLAM	94.204 ± 0.143	0.017	90.527 ± 0.087	0.071	89.540 ± 0.047	0.225
ASAM	97.356 ± 0.100	0.011	91.503 ± 0.197	0.116	89.637 ± 0.104	0.287
PSAM	97.508 ± 0.143	0.015	92.218 ± 0.024	0.188	90.111 ± 0.011	0.367
BAM	97.72	0.110	91.56	0.570	90.43	1.7459

Algorithm	connect-4		mnist		acoustic	
	AUC	Training time	AUC	Training time	AUC	Training time
OAM_{seq}	79.737 ± 0.179	0.618	92.176 ± 0.748	3.928	82.116 ± 2.264	0.368
OAM_{gra}	78.501 ± 1.504	0.602	92.097 ± 0.656	3.950	78.000 ± 8.433	0.354
CBR_{FIFO}	88.151 ± 0.029	37.75	95.753 ± 0.119	17.24	88.573 ± 0.076	11.251
SOLAM	87.491 ± 0.062	0.433	94.866 ± 0.046	3.024	87.083 ± 0.177	0.250
ASAM	87.771 ± 0.057	0.701	95.911 ± 0.047	2.164	88.393 ± 0.024	0.543
PSAM	88.200 ± 0.005	0.851	96.051 ± 0.018	2.768	88.557 ± 0.010	0.672
BAM	88.20	3.429	96.05	27.74	87.38	1.880

Algorithm	webspam		cod-rna		epsilon	
	AUC	Training time	AUC	Training time	AUC	Training time
OAM_{seq}	95.432 ± 0.399	6.571	94.956 ± 3.567	0.343	88.201 ± 0.412	67.44
OAM_{gra}	95.331 ± 0.334	6.356	97.632 ± 0.121	0.332	87.375 ± 0.614	68.19
CBR_{FIFO}	97.234 ± 0.024	37.75	98.893 ± 0.003	5.913	95.591 ± 0.074	689.4
SOLAM	96.615 ± 0.025	4.536	98.770 ± 0.006	0.250	95.961 ± 0.006	53.50
ASAM	97.197 ± 0.026	4.870	98.900 ± 0.012	0.896	95.814 ± 0.031	37.46
PSAM	97.250 ± 0.004	6.289	98.687 ± 0.002	1.203	95.964 ± 0.001	50.35
BAM	97.37	17.56	98.86	2.046	95.97	837.8

Algorithm	covtype		susy		aloi	
	AUC	Training time	AUC	Training time	AUC	Training time
OAM_{seq}	78.683 ± 1.952	2.378	71.957 ± 0.669	8.973	73.226 ± 1.521	1.088
OAM_{gra}	80.760 ± 1.613	2.281	69.810 ± 7.131	8.590	74.128 ± 2.219	1.007
CBR_{FIFO}	86.760 ± 0.895	72.60	85.953 ± 0.001	202.7	81.576 ± 0.243	80.14
SOLAM	86.425 ± 0.114	1.662	83.525 ± 0.015	5.723	73.846 ± 1.636	0.719
ASAM	86.851 ± 0.048	4.183	85.820 ± 0.060	21.06	80.311 ± 0.145	0.789
PSAM	87.059 ± 0.001	5.344	85.950 ± 0.001	28.44	80.993 ± 0.020	1.248
BAM	87.18	15.02	85.81	63.75	81.64	12.36

Algorithm	farm-ads		sector		news20	
	AUC	Training time	AUC	Training time	AUC	Training time
OAM_{seq}	89.260 ± 0.091	19.43	98.258 ± 0.240	39.108	97.610 ± 0.100	90.90
OAM_{gra}	90.080 ± 0.397	20.15	98.161 ± 0.206	37.94	97.647 ± 0.046	89.40
CBR_{FIFO}	93.959 ± 1.572	226.8	98.720 ± 0.245	541.6	98.619 ± 0.074	1576
SOLAM	91.738 ± 0.385	19.39	96.836 ± 0.186	29.39	97.605 ± 0.021	65.73
ASAM	95.690 ± 0.071	7.877	97.719 ± 0.039	20.537	97.904 ± 0.006	36.67
PSAM	95.839 ± 0.037	12.45	98.875 ± 0.013	25.03	98.182 ± 0.395	56.73

[7] C. CORTES AND M. MOHRI, *Auc optimization vs. error rate minimization*, Advances in neural information processing systems, 16 (2004), pp. 313–320.

[8] K. CRAMMER, O. DEKEL, J. KESHET, S. SHALEV-SHWARTZ, AND Y. SINGER, *Online passive-aggressive algorithms*, Journal of Machine Learning Research, 7 (2006), pp. 551–585.

[9] A. DEFAZIO, *A simple practical accelerated method for finite sums*, in Advances in Neural Information Processing Systems, 2016, pp. 676–684.

[10] A. DEFAZIO, F. BACH, AND S. LACOSTE-JULIEN, *Saga: A fast incremental gradient method with support for non-strongly convex composite objectives*, in Advances in neural information processing systems, 2014, pp. 1646–1654.

[11] W. GAO, R. JIN, S. ZHU, AND Z.-H. ZHOU, *One-pass auc optimization.*, in ICML (3), 2013, pp. 906–914.

[12] J. A. HANLEY AND B. J. MCNEIL, *The meaning and use of the area under a receiver operating characteristic (roc) curve.*, Radiology, 143 (1982), pp. 29–36.

[13] V. KAKKAR, S. SHEVADE, S. SUNDARARAJAN, AND D. GARG, *A sparse nonlinear classifier design using auc optimization*, in Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, 2017, pp. 291–299.

[14] M. KHALID, I. RAY, AND H. CHITSAZ, *Confidence-weighted bipartite ranking*, in Advanced Data Mining and Applications: 12th International Conference, ADMA 2016, Gold Coast, QLD, Australia, December 12-15, 2016, Proceedings 12, Springer, 2016, pp. 35–49.

[15] M. KHALID, I. RAY, AND H. CHITSAZ, *Scalable nonlinear auc maximization methods*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 292–307.

[16] T.-M. KUO, C.-P. LEE, AND C.-J. LIN, *Large-scale kernel ranksvm*, in Proceedings of the 2014 SIAM international conference on data mining, SIAM, 2014, pp. 812–820.

TABLE III

COMPARISON OF AUC CLASSIFICATION ACCURACY AND TRAINING TIME (IN SECONDS) FOR THE NONLINEAR VARIANTS OF THE BATCH AND THE STOCHASTIC AUC MAXIMIZATION ALGORITHMS. THE TRAINING TIME EXCLUDES THE EMBEDDING STEPS.

Algorithm	acoustic		aloi		cod-rna	
	AUC	Training time	AUC	Training time	AUC	Training time
NSOLAM	92.826 $\pm$ 0.279	8.25	92.450 $\pm$ 0.212	8.99	99.108 $\pm$ 0.000	34.93
NASAM	93.316 $\pm$ 0.063	5.61	98.992 $\pm$ 0.025	5.75	99.163 $\pm$ 0.007	23.65
NPSAM	94.073 $\pm$ 0.028	7.36	99.507 $\pm$ 0.002	8.22	99.195 $\pm$ 0.000	31.15
NBAM	94.173	142.4	99.742	171.8	99.182	419.5

Algorithm	webspam		covtype		susy	
	AUC	Training time	AUC	Training time	AUC	Training time
NSOLAM	99.594 $\pm$ 0.001	29.34	94.324 $\pm$ 0.241	49.07	86.933 $\pm$ 0.000	303.44
NASAM	99.629 $\pm$ 0.014	19.45	95.201 $\pm$ 0.081	34.26	87.200 $\pm$ 0.024	603.08
NPSAM	99.759 $\pm$ 0.000	26.74	96.150 $\pm$ 0.021	45.40	87.301 $\pm$ 0.006	589.48
NBAM	99.740	295.65	96.650	2025.1	87.280	5512.3

[17] C.-P. LEE AND C.-J. LIN, *Large-scale linear ranksvm*, Neural computation, 26 (2014), pp. 781–817.

[18] M. LIU, X. ZHANG, Z. CHEN, X. WANG, AND T. YANG, *Fast stochastic auc maximization with o (1/n)-convergence rate*, in International Conference on Machine Learning, 2018, pp. 3195–3203.

[19] T.-Y. LIU, *Learning to rank for information retrieval*, Foundations and Trends in Information Retrieval, 3 (2009), pp. 225–331.

[20] M. NATOLE, Y. YING, AND S. LYU, *Stochastic proximal algorithms for auc maximization*, in International Conference on Machine Learning, 2018, pp. 3707–3716.

[21] A. NITANDA, *Stochastic proximal gradient descent with acceleration techniques*, in Advances in Neural Information Processing Systems, 2014, pp. 1574–1582.

[22] S. RENDLE, L. BALBY MARINHO, A. NANOPOULOS, AND L. SCHMIDT-THIEME, *Learning optimal ranking with tensor factorization for tag recommendation*, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 727–736.

[23] H. ROBBINS AND S. MONRO, *A stochastic approximation method*, in Herbert Robbins Selected Papers, Springer, 1985, pp. 102–109.

[24] J. ROOT, J. QIAN, AND V. SALIGRAMA, *Learning efficient anomaly detectors from k-nn graphs*, in Artificial Intelligence and Statistics, 2015, pp. 790–799.

[25] S. SHALEV-SHWARTZ, Y. SINGER, N. SREBRO, AND A. COTTER, *Pegasos: Primal estimated sub-gradient solver for svm*, Mathematical programming, 127 (2011), pp. 3–30.

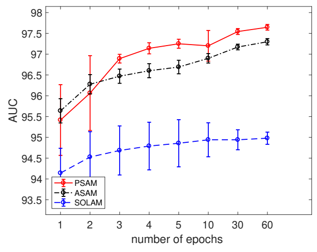
[26] L. XIAO AND T. ZHANG, *A proximal stochastic gradient method with progressive variance reduction*, SIAM Journal on Optimization, 24 (2014), pp. 2057–2075.

[27] Z. XIE AND M. LI, *Cutting the software building efforts in continuous integration by semi-supervised online auc optimization.*, in IJCAI, 2018, pp. 2875–2881.

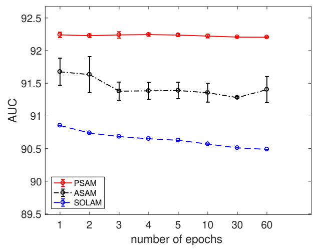
[28] Y. YING, L. WEN, AND S. LYU, *Stochastic online auc maximization*, in Advances in Neural Information Processing Systems, 2016, pp. 451–459.

[29] K. ZHANG, I. W. TSANG, AND J. T. KWOK, *Improved nyström low-rank approximation and error analysis*, in Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1232–1239.

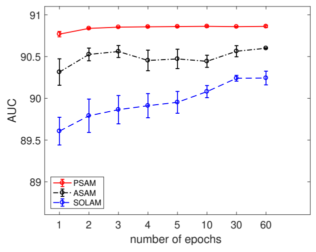
[30] P. ZHAO, R. JIN, T. YANG, AND S. C. HOI, *Online auc maximization*, in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 233–240.



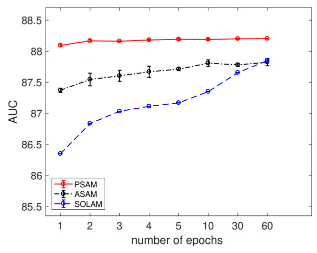
(a) spambase



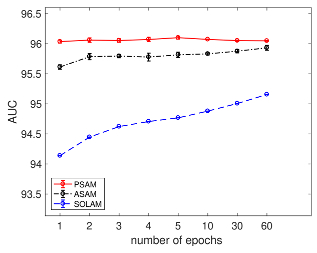
(b) ijcnn1



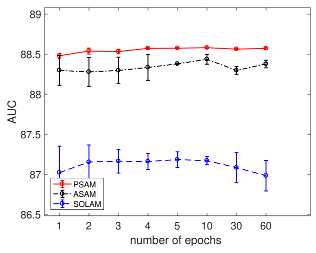
(c) a9a



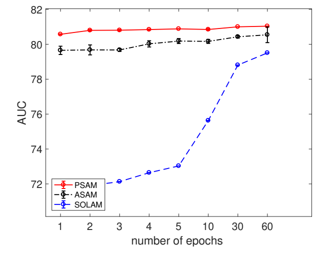
(d) connect-4



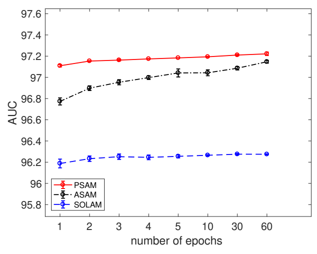
(e) mnist



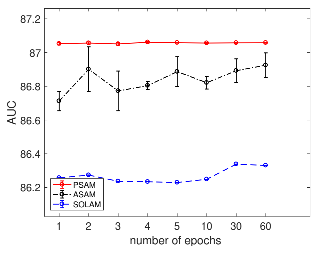
(f) acoustic



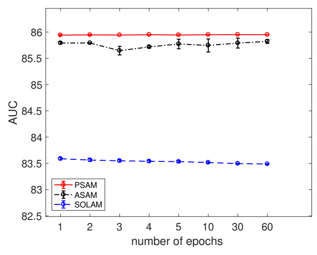
(g) aloi



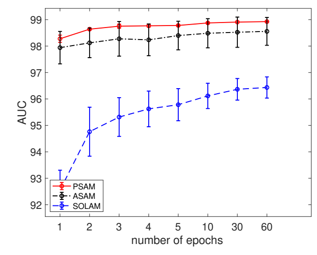
(h) webspam



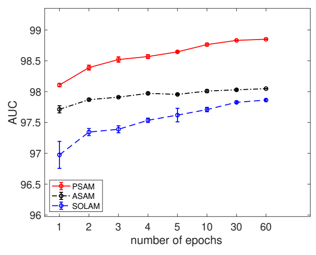
(i) covtype



(j) susy



(k) sector



(l) news20

Fig. 1. AUC classification accuracy with respect to the number of epochs for stochastic linear AUC methods. We randomly pick a positive and negative instance in each iteration for PSAM and ASAM, where n iterations correspond to one epoch.