Communications for Statistical Applications and Methods
2024, Vol. 31, No. 2, 203–212

https://doi.org/10.29220/CSAM.2024.31.2.203
Print ISSN 2287-7843 / Online ISSN 2383-4757

L1-penalized AUC-optimization with a surrogate loss

Hyungwoo Kim[1,a], Seung Jun Shin[b]

[a]Department of Statistics and Data Science, Pukyong National University, Korea;
[b]Department of Statistics, Korea University, Korea

Abstract

The area under the ROC curve (AUC) is one of the most common criteria used to measure the overall performance of binary classifiers for a wide range of machine learning problems. In this article, we propose a L_1-penalized AUC-optimization classifier that directly maximizes the AUC for high-dimensional data. Toward this, we employ the AUC-consistent surrogate loss function and combine the L_1-norm penalty which enables us to estimate coefficients and select informative variables simultaneously. In addition, we develop an efficient optimization algorithm by adopting k-means clustering and proximal gradient descent which enjoys computational advantages to obtain solutions for the proposed method. Numerical simulation studies demonstrate that the proposed method shows promising performance in terms of prediction accuracy, variable selectivity, and computational costs.

Keywords: AUC-optimization, AUC consistency, variable selection, L_1-norm penalty, clustering and proximal gradient descent

1. Introduction

Classifying the binary data is a fundamental task in the machine learning community, where the primary goal is to categorize data into one of two classes. Most existing classification algorithms are designed to maximize the accuracy (or equivalently minimize the error rate), however, it may be a misleading performance metric for imbalanced data with few samples in the minority class since they classify all examples into the majority class by achieving a high classification accuracy.

A receiver operation characteristic (ROC) analysis originated from signal detection theory (Egan, 1975) can be a great alternative for imbalanced classification problems and it has become the standard methodology to evaluate the model performance of the binary classifier in various fields such as psychology, medicine, bioinformatics, and machine learning. The ROC curve is a graphical plot that features a true positive rate (TPR) on the Y axis and a false positive rate (FPR) on the X axis. It is particularly useful for imbalanced data due to its diagnostic ability for evaluating the classifier's performance over a variety of decision thresholds.

As a summary measure of the ROC curve, the area under the ROC curve (AUC) is the most popular scalar metric in classification. It equals to the probability that a classifier will rank a randomly drawn positive example higher than a randomly drawn negative example, which is closely related to the

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00242528).

[1]Corresponding author: Department of Statistics and Data Science, Pukyong National University, 45 Yongso-ro, Nam-gu, Busan 04853, Korea. E-mail: khw7682@pknu.ac.kr

Published 31 March 2024 / journal homepage: http://csam.or.kr

bipartite ranking problem (Agarwal *et al.*, 2005; Clémençon *et al.*, 2008; Menon and Williamson, 2016). Namely, the population AUC of a classifier f is defined as

$$\mathrm{AUC}(f) = \mathbb{P}\left(f(\mathbf{X}^{+}) > f(\mathbf{X}^{-})\right), \tag{1.1}$$

where $\mathbf{X}^{+}$ and $\mathbf{X}^{-}$ denote a p-dimensional input vector that belongs to the binary class $Y = +1$ and $Y = -1$, respectively. The perfect classifier f takes the value of 1 for AUC while the random guess does 0.5.

It is noteworthy that a high AUC value represents the better classification performance of the classifier, and thus a huge amount of studies have been devoted to directly maximizing the AUC in the two past decades. They include various forms such as linear models (Brefeld and Scheffer, 2005; Ying *et al.*, 2016; Norton and Uryasev, 2019), kernel models (Rakotomamonjy, 2004; Ataman *et al.*, 2006), and neural networks (Liu *et al.*, 2019; Yuan *et al.*, 2021). In addition, several AUC-optimization algorithms have been developed under different loss functions (Rakotomamonjy, 2004; Zhang *et al.*, 2012; Natole *et al.*, 2018; Yang *et al.*, 2020; Lei and Ying, 2021). Due to the non-continuity of AUC (1.1) in sample level, most learning algorithms for AUC maximization resort to a convex relaxation of the loss function such as hinge (Rakotomamonjy, 2004) for support vector machine (SVM), exponential (Clémençon *et al.*, 2013) for boosting, least square (Natole *et al.*, 2018), and logistic surrogate (Yang *et al.*, 2020). A typical choice for AUC maximization is hinge loss because of its computational advantage via quadratic or linear programming. After Rakotomamonjy (2004) introduced a SVM-type approach that directly maximizes the AUC, called the ROC-SVM, several variants of the ROC-SVM have been studied (Arzhaeva *et al.*, 2006; Tian *et al.*, 2011; Zhao *et al.*, 2011; Ying *et al.*, 2016).

Theoretical analysis to make use of surrogate loss in the AUC-maximization context has recently been examined. Gao and Zhou (2015) first introduced the notion of AUC consistency that guarantees the solution of the AUC-optimization with surrogate loss yields the one with original loss in the asymptotic sense. Moreover, Gao and Zhou (2015); Uematsu and Lee (2017) provided a sufficient condition for AUC consistency and proved that hinge loss is inconsistent with AUC because of its non-differentiability at one. To our knowledge, there lacks of study on variable selection for AUC-optimization with the AUC-consistent surrogate loss. In this work, we try to this gap by employing a logistic surrogate loss function which is statistically AUC consistent (Gao and Zhou, 2015) and combining the L_1-norm penalty (Tibshirani, 1996). The proposed method performs coefficient estimation and automatic variable selection simultaneously due to the L_1 penalty, which is very useful for the prediction and interpretability of the classifier f in high-dimensional data analysis.

It is well known that AUC optimization is very computationally expensive because it has a form of summation of pairwise losses over pairs of examples, while most machine learning algorithms that focus on maximizing accuracy (e.g. SVM) take the sum of pointwise losses over individual examples. As mentioned earlier above, AUC-optimization with hinge loss has the advantage of computation by using solvers via quadratic programming (Rakotomamonjy, 2004) or linear programming (Arzhaeva *et al.*, 2006) or even regularization path algorithms (Kim and Shin, 2020; Kim *et al.*, 2021), but such techniques are inapplicable to the proposed AUC-optimization with logistic loss. To mitigate such computational burden for the proposed method, we develop an efficient optimization algorithm by adopting k-means clustering (Duda *et al.*, 1973; Brefeld and Scheffer, 2005) and proximal gradient descent (PGD) (Rockafellar, 1976; Combettes and Wajs, 2005), which we call CPAUC. Through the k-means clustering algorithm, we approximate the objective function from a double summation formulation to a single one, which drastically reduces the computational costs. Also, the PGD algorithm enables us to solve the non-differentiable optimization problem with a fast convergence rate.

By combining two techniques, the CPAUC algorithm successively reduces the running time for the L_1-penalized AUC-optimization with the logistic loss.

The remainder of this article is organized as follows. In Section 2, we briefly describe the AUC consistency and give the formulation of L_1-penalized AUC optimization with the surrogate loss. We present an efficient algorithm to find the parameter of interest for the proposed method in Section 3. Numerical simulation studies and real data analysis results are presented in Sections 4 and 5, respectively. The concluding remarks follow in Section 6.

2. L_1-penalized AUC optimization with logistic loss

Suppose we are given n observations with $(\mathbf{x}_i, y_i)_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$ is a p-dimensional predictor and $y_i \in \{-1, +1\}$ is a binary outcome. Let us define the index set $I_+ = \{i : y_i = +1\}$ and $I_- = \{i : y_i = -1\}$, and the corresponding number of positive and negative samples are $n_+ = |I_+|$ and $n_- = |I_-|$, respectively. Considering a linear classifier $f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T\mathbf{x}$ with $\boldsymbol{\beta}^T = (\beta_1, \dots, \beta_p)$, the empirical AUC of (1) is denoted by

$$\frac{1}{n_+n_-}\sum_{i\in I_+}\sum_{j\in I_-}\mathbb{1}\left\{f(\mathbf{x}_i) - f\left(\mathbf{x}_j\right) \geq 0\right\}, \tag{2.1}$$

where $\mathbb{1}\{A\}$ is an indicator function that takes value 1 if A is true and 0 otherwise. To maximize (2.1), it is equivalent to minimize $1 - \mathrm{AUC}(f)$ and thus we solve the minimization problem as follows: $\min(1/n_+n_-)\sum_{i\in I_+}\sum_{j\in I_-}\mathbb{1}\{f(\mathbf{x}_i) - f(\mathbf{x}_j) < 0\}$. However, the direct optimization problem of (2.1) is ill-posed (Feldman *et al.*, 2012) since the objective function (2.1) is not continuous and the solution may not be unique. To make it well-posed, one may consider the convex surrogate loss and employ the regularization term as follows:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ \frac{1}{n_+n_-}\sum_{i\in I_+}\sum_{j\in I_-}\phi\left\{f(\mathbf{x}_i) - f\left(\mathbf{x}_j\right)\right\} + \lambda J(\boldsymbol{\beta}), \tag{2.2}$$

where $\phi\{\cdot\}$ is a convex surrogate function, $J(\boldsymbol{\beta})$ is a regularization term that prevents overfitting, and $\lambda > 0$ is a regularization strength parameter. The most popular choice of ϕ in the AUC maximization context is the hinge loss for SVM, which takes the form $\phi(z) = \max\{0, 1 - z\}$. Rakotomamonjy (2004) proposed the AUC-maximizing SVM with L_2-norm penalty $J(\boldsymbol{\beta}) = \boldsymbol{\beta}^T\boldsymbol{\beta}$. Furthermore, for the effective variable selection, Ataman *et al.* (2006) and Tian *et al.* (2011) proposed AUC-maximizing SVM with L_1-norm and L_q-norm $(0 < q < 1)$ penalty, respectively.

Gao and Zhou (2015) introduced the AUC consistency that guarantees the optimal solution of surrogate loss for (2.2) without the regularization term (i.e., $\lambda = 0$) yields an optimal solution for (2.1) in the asymptotic sense. Moreover, Gao and Zhou (2015) showed that the hinge loss for SVM is not AUC-consistent because of its non-differentiability at $z = 1$. In this work, we propose a AUC-optimization classifier by adopting the AUC-consistent surrogate loss and combining the L_1-norm penalty as follows:

$$\hat{\boldsymbol{\beta}}_n = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ Q_n(\boldsymbol{\beta}) := \frac{1}{n_+n_-}\sum_{i\in I_+}\sum_{j\in I_-}\phi\left\{\boldsymbol{\beta}^T\left(\mathbf{x}_i - \mathbf{x}_j\right)\right\} + \lambda\sum_{k=1}^{p}|\beta_k|, \tag{2.3}$$

which we call L_1-AUC surrogate estimator. For the choice of AUC-consistent surrogate function ϕ, we consider a popular logistic loss function that takes the form $\phi(z) = \log(1 + \exp(-z))$. Notice that AUC-optimizing classification (2.3) does not involve an intercept β_0 and thus it is unidentifiable. However,

β_0 becomes identifiable if the TPR or FPR is provided in practice. We remark that $\hat{\boldsymbol{\beta}}_n$ naturally becomes a sparse estimate due to the geometry of L_1-norm penalty in (2.3), which is particularly useful when the predictor is high-dimensional.

3. CPAUC algorithm

Notice that the AUC-optimization algorithm requires considerable computational expenses because the objective function $Q(\boldsymbol{\beta})$ in (2.3) takes a double sum loss over pairs of examples while most machine learning algorithms based on the accuracy (e.g. SVM) is a sum of loss over individual examples. Therefore, the running time for the AUC optimization will be acceptable for small to moderate samples but undesirable for large and high dimensional data.

To find an optimal solution (2.3) with a relatively low time complexity, we propose a clustering proximal AUC maximization (CPAUC) algorithm by adopting the k-means clustering and PGD algorithm. Since the optimization (2.3) is only involved in the predictors as a pair of observations $\delta_{ij} = \mathbf{x}_i - \mathbf{x}_j$ with n_+n_- vectors, we can apply the k-means clustering to reduce the number of examples by generating $N(\ll n_+n_-)$ new cluster centers, denoted by a p-dimensional vector $\mathbf{c}_m$ where $m = 1, \ldots, N$. Here, the choice of the number of clusters N is crucial for a given dataset and there are several techniques such as elbow curve (Thorndike, 1953) and silhouette (Rousseeuw, 1987) for finding an optimal N. In this work, we performed small Monte-Carlo simulations and chose sufficiently large N as the number of sample size, i.e., $N = n$. Then, we solve the following approximated AUC-optimization problem using n clustered samples:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \tilde{Q}_n(\boldsymbol{\beta}) := \frac{1}{n}\sum_{m=1}^{n} \phi\left\{\boldsymbol{\beta}^T \mathbf{c}_m\right\} + \lambda \sum_{k=1}^{p} |\beta_k|, \tag{3.1}$$

which drastically reduce the time complexity of (2.3).

To solve (3.1) effectively, we employ the PGD algorithm that has recently attracted much attention because of its fast convergence rate and ability to handle the non-differentiable objective function. For this, we represent (3.1) as the sum of smooth and non-smooth convex function, i.e., $\tilde{Q}_n(\boldsymbol{\beta}) = \tilde{Q}^s_n(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|$, where $\tilde{Q}^s_n(\boldsymbol{\beta}) = (1/n)\sum_{m=1}^{n} \phi\{\boldsymbol{\beta}^T \mathbf{c}_m\}$. The PGD algorithm iteratively updates the parameter by solving the following problem at the t^{th} iteration:

$$\hat{\boldsymbol{\beta}}^{(t+1)} = \operatorname{Prox}_\kappa \left\{\hat{\boldsymbol{\beta}}^{(t)} - \kappa \triangledown \tilde{Q}^s_n\left(\hat{\boldsymbol{\beta}}^{(t)}\right)\right\}, \tag{3.2}$$

where $\hat{\boldsymbol{\beta}}^{(t)} = (\hat{\beta}^{(t)}_1, \ldots, \hat{\beta}^{(t)}_p)^T$ denotes the t^{th} current solution and $\kappa(> 0)$ plays the role of learning rate. The proximal operator in (3.2) is defined as $\operatorname{Prox}_\kappa\{\boldsymbol{\beta}\} = \operatorname{argmin}_{\mathbf{s}\in\mathbb{R}^p}(1/2\kappa)\|\boldsymbol{\beta}-\mathbf{s}\|_2^2 + \lambda\|\mathbf{s}\|_1$ and $\triangledown$ denotes the first partial derivative operator with respect to $\boldsymbol{\beta}$. To summarize, the proposed algorithm CPAUC is presented in Algorithm 1.

4. Simulation

In this section, we carry out numerical experiments to investigate the superior performance of the proposed method in terms of prediction accuracy, variable selection, and computing time. All experiments are implemented with 3.4GHz Intel Core(TM), 64GB RAM and 64-bit R version 4.3.1.

Algorithm 1 : Clustering proximal AUC maximization (CPAUC)

```
    Input: $y, \kappa, \lambda$, and standardized $\mathbf{x}$.
    Output: An approximated solution (3.1)
1: Clustering: Apply the $k$-means clustering for $\delta_{ij}$ with $n_+n_-$ examples to generate $n$ centered clusters.
2: Proximal gradient descent:
    (a) Initialize $\hat{\boldsymbol{\beta}}^{(0)} = \mathbf{0}$ with $t = 0$
    (b) repeat
            Compute $\hat{\boldsymbol{\beta}}^{(t+1)} \leftarrow \text{Prox}_\kappa \left\{\hat{\boldsymbol{\beta}}^{(t)} - \kappa \nabla Q_n^s\left(\hat{\boldsymbol{\beta}}^{(t)}\right)\right\}$
            $t \leftarrow t + 1$
        until convergence
```

4.1. Sample performance

To examine the prediction accuracy and variable selection ability of the proposed method, we compare it with the popular binary classification methods: L$_2$-SVM (Cortes and Vapnik, 1995), L$_1$-SVM (Zhu *et al.*, 2003), L$_2$-ROCSVM (Rakotomamonjy, 2004), L$_1$-ROCSVM (Kim *et al.*, 2021), and L$_2$-ROCLOG. In this work, we denote the L$_2$-penalized AUC-optimization with logistic loss by L$_2$-ROCLOG and the proposed method using all examples and n-clustered samples by L$_1$-ROCLOG^1 and L$_1$-ROCLOG^2, respectively.

We consider the imbalanced scenario with the proportion of positive class as 0.9, i.e., $\mathbb{P}(Y = +1) = 1 - \mathbb{P}(Y = -1) = 0.9$. For given the class label Y, we generate the predictor $\mathbf{X}$ from multivariate normal (MN) distribution with MN$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for the positive class and MN$(-\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for the negative class, where a mean vector $\boldsymbol{\mu} = (\mathbf{z}_k^T, \mathbf{0}_{p-k}^T)^T$ and covariance structure $\boldsymbol{\Sigma}$. The data is generated from the following four models:

(M1) $\boldsymbol{\mu}^T = (0.8, -0.8, 0.8, \mathbf{0}_{p-3}^T)$ and $\boldsymbol{\Sigma} = 2I_p$.

(M2) $\boldsymbol{\mu}^T = (0.8, -0.8, 0.8, \mathbf{0}_{p-3}^T)$ and $\boldsymbol{\Sigma} = (\sigma_{ij})$ with its non-zero elements $\sigma_{ii} = 2$ for $i = 1, \dots, p$ and $\sigma_{ij} = 0.3^{|i-j|}$ for $1 \leq i \neq j \leq 3$.

(M3) $\boldsymbol{\mu}^T = (1, -0.8, 0.6, -0.4, 0.2, \mathbf{0}_{p-5}^T)$ and $\boldsymbol{\Sigma} = 2I_p$.

(M4) $\boldsymbol{\mu}^T = (1, -0.8, 0.6, -0.4, 0.2, \mathbf{0}_{p-5}^T)$ and $\boldsymbol{\Sigma} = (\sigma_{ij})$ with its non-zero elements $\sigma_{ii} = 2$ for $i = 1, \dots, p$ and $\sigma_{ij} = 0.3^{|i-j|}$ for $1 \leq i \neq j \leq 5$.

Notice that (M1)–(M4) present the sparse models where the first k components are only informative. For (M2) and (M4), we set the pairwise correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$ to be $0.3^{|i-j|}$ for $1 \leq i \neq j \leq k$. We consider a different combination of the training sample size and the predictor dimension with $(n, p) \in \{100, 200\} \times \{20, 40\}$. For a fair comparison with tuning parameter selection, we implement a 5-fold cross-validation (CV) over the interval $1.8^{\{-8,-7,\dots,1,2\}}$ and then evaluate each classifier's performance based on the 500 independent test samples generated from (M1)–(M4). We also examine the variable selection ability by identifying the number of correctly selected relevant variables (CS) and incorrectly selected (ICS) irrelevant variables, respectively.

Table 1: The average of test AUC over 100 independent repetitions are reported for (M1)–(M4)

Model	n	p	L_2-SVM	L_1-SVM	L_2-ROCSVM	L_1-ROCSVM	L_2-ROCLOG	L_1-ROCLOG[1]	L_1-ROCLOG[2]
(M1)	100	20	0.830 (0.05)	0.852 (0.05)	0.820 (0.05)	0.857 (0.06)	0.851 (0.04)	0.870 (0.04)	0.868 (0.04)
		40	0.776 (0.05)	0.828 (0.05)	0.759 (0.06)	0.824 (0.07)	0.809 (0.05)	0.850 (0.05)	0.849 (0.04)
	200	20	0.869 (0.04)	0.879 (0.04)	0.854 (0.05)	0.886 (0.05)	0.884 (0.04)	0.891 (0.04)	0.893 (0.04)
		40	0.827 (0.03)	0.861 (0.03)	0.804 (0.05)	0.886 (0.04)	0.856 (0.03)	0.889 (0.04)	0.885 (0.04)
(M2)	100	20	0.856 (0.04)	0.876 (0.04)	0.833 (0.05)	0.878 (0.05)	0.869 (0.04)	0.889 (0.04)	0.886 (0.04)
		40	0.803 (0.05)	0.865 (0.05)	0.798 (0.05)	0.876 (0.06)	0.836 (0.04)	0.882 (0.05)	0.879 (0.06)
	200	20	0.896 (0.03)	0.907 (0.03)	0.884 (0.03)	0.919 (0.02)	0.906 (0.02)	0.923 (0.02)	0.923 (0.02)
		40	0.855 (0.03)	0.895 (0.03)	0.845 (0.04)	0.909 (0.03)	0.870 (0.04)	0.911 (0.03)	0.909 (0.03)
(M3)	100	20	0.854 (0.04)	0.887 (0.04)	0.856 (0.05)	0.869 (0.05)	0.874 (0.04)	0.885 (0.04)	0.882 (0.04)
		40	0.807 (0.03)	0.854 (0.03)	0.797 (0.05)	0.865 (0.04)	0.843 (0.03)	0.875 (0.04)	0.874 (0.04)
	200	20	0.890 (0.03)	0.895 (0.03)	0.871 (0.03)	0.901 (0.03)	0.902 (0.03)	0.908 (0.03)	0.905 (0.03)
		40	0.857 (0.03)	0.881 (0.03)	0.816 (0.04)	0.898 (0.04)	0.882 (0.03)	0.901 (0.03)	0.898 (0.03)
(M4)	100	20	0.894 (0.02)	0.903 (0.02)	0.887 (0.02)	0.896 (0.03)	0.902 (0.03)	0.910 (0.03)	0.909 (0.04)
		40	0.835 (0.03)	0.886 (0.03)	0.827 (0.03)	0.888 (0.04)	0.865 (0.03)	0.891 (0.03)	0.888 (0.04)
	200	20	0.919 (0.02)	0.921 (0.02)	0.901 (0.03)	0.928 (0.02)	0.926 (0.02)	0.931 (0.02)	0.929 (0.02)
		40	0.886 (0.03)	0.915 (0.03)	0.852 (0.04)	0.922 (0.02)	0.899 (0.03)	0.925 (0.02)	0.920 (0.02)

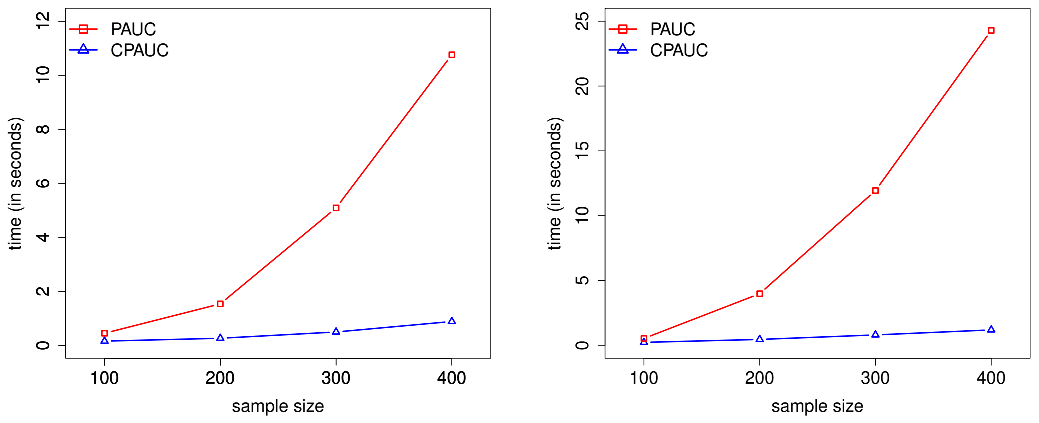
The numbers in the parentheses are standard errors.

Table 2: The average of CS and ICS over 100 independent repetitions are reported for (M1)–(M4)

Model	n	p	L_1-SVM		L_1-ROCSVM		L_1-ROCLOG[2]		Oracle	
			CS	ICS	CS	ICS	CS	ICS	CS	ICS
(M1)	100	20	2.92	8.66	2.78	4.76	2.70	4.96	3.00	0.00
		40	2.90	13.14	2.50	6.94	2.68	8.14		
	200	20	3.00	9.98	2.88	2.78	2.96	3.62		
		40	3.00	18.12	3.00	6.62	2.94	6.78		
(M2)	100	20	2.94	7.82	2.78	4.32	2.88	5.24		
		40	3.00	12.34	2.72	5.58	2.76	6.56		
	200	20	3.00	10.06	3.00	3.34	3.00	3.18		
		40	3.00	16.18	2.98	5.62	2.94	6.12		
(M3)	100	20	4.03	6.63	3.34	3.78	3.51	4.67	5.00	0.00
		40	3.83	12.09	2.98	4.98	3.36	8.59		
	200	20	4.48	9.00	3.85	3.75	4.02	4.58		
		40	4.50	16.39	3.43	4.27	3.89	6.67		
(M4)	100	20	4.16	7.05	3.59	4.25	3.94	5.13		
		40	3.93	10.68	3.08	4.30	3.41	6.41		
	200	20	4.57	8.92	3.90	3.70	4.06	4.24		
		40	4.55	15.18	3.78	5.17	4.05	7.26		

Table 1 reports the averaged test AUC computed from seven classifiers over 100 independent repetitions. It is shown that all classification methods get better as the sample size increases and worse the predictor dimension increases. One can observe that AUC-maximization algorithms consistently yield better results than those based on the accuracy within the same penalty function. It is not surprising that L_1-penalized classifiers lead to great prediction power compared to the L_2-penalized ones by removing noise variables, especially as the predictor dimension increases. We note that ROCLOG works slightly better than ROCSVM with hinge loss, which is known to be inconsistent with AUC. The results for L_1-ROCLOG[1] and L_1-ROCLOG[2] are comparable in terms of AUC, but the latter is significantly more computationally efficient compared with the former, which will be discussed in detail in Section 4.2.

Table 2 summarizes the variable selection results of L_1-penalized classifiers in terms of CS and ICS over 100 independent repetitions. It is observed that L_1-SVM does worst in terms of ICS for all



(a) $p = 20$ (b) $p = 40$

Figure 1: *Comparison of running time for PAUC and CPAUC.*

models. The variable selection performance of L_1-ROCSVM and L_1-ROCLOG[2] gets better as the sample size increases. We can see that the results of AUC-optimization classifiers are comparable, but L_1-ROCSVM slightly works better than L_1-ROCLOG[2] in terms of ICS.

4.2. Computing time

In this subsection, we show the advantage of the proposed CPAUC algorithm in terms of computing time. We compare it with the one based on the PGD algorithm using all observations (without clustering), which we call PAUC. The simulation settings are identical to (M1) in Section 4.1. We reported the running time in the setting of increasing sample size and predictor dimension with $(n, p) \in \{100, 200, 300, 400\} \times \{20, 40\}$.

Figure 1 denotes the comparison of running time (in seconds) between CPAUC and PAUC over 100 independent repetitions. The regularization parameter λ is chosen with the maximum AUC value computed from 5-CV. One can observe that the running time of CPAUC is significantly reduced after applying the k-means clustering algorithm. We also remark that CPAUC becomes more computationally efficient than PAUC as the sample size and the predictor dimension increase.

5. Real data analysis

In this section, we apply the proposed method to the Pima Indians Diabetes (PID) data set from the national institute of diabetes and digestive and kidney disease, which is available at Kaggle and UCI Machine Learning Repository. The main purpose of this study is to accurately predict whether a Pima Indian female has diabetes or not. This data set contains 768 participants (268 diabetic and 500 non-diabetic) with 8 real-valued input variables.

We first split PID data as 70% train and 30% test set, respectively. We applied L_2-SVM, L_1-SVM, L_2-ROCLOG, and L_1-ROCLOG described in Section 4.1 to the train set and then computed the AUC using test set. The optimal regularization parameter for each classifier is chosen by a 5-fold CV. In this work, we excluded ROCSVM classifiers because they fail to produce the solutions from both

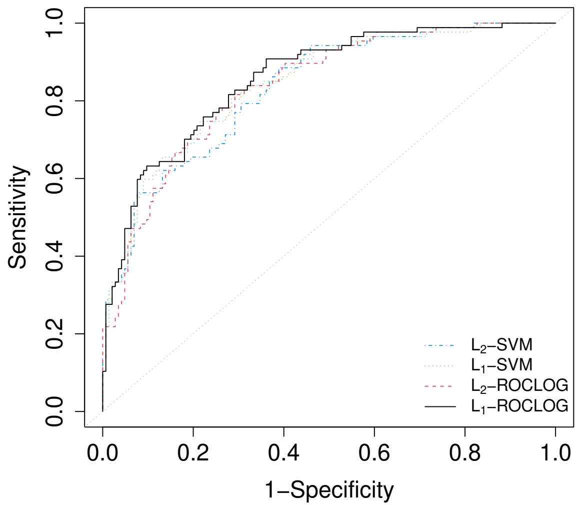


Figure 2: *The comparison of the ROC curve with four classifiers. The L_1-penalized ROCLOG outperforms all others.*

Table 3: The number of selected predictors and test AUC for employed four classifiers

	Number of selected variables	Test AUC
L_2-SVM	8	0.832 (0.025)
L_1-SVM	7	0.843 (0.023)
L_2-ROCLOG	8	0.837 (0.025)
L_1-ROCLOG[2]	5	0.856 (0.024)

Standard errors are reported in the parentheses over 100 independent repetitions.

quadratic programming and regularization paths in R-package.

Figure 2 presents the test ROC curves of the four classifiers and Table 3 reports the corresponding test AUC and the number of selected predictors. It is observed that the proposed method attains the highest test AUC with a minimal number of predictors.

6. Concluding remarks

In this paper, we consider the L_1-penalized AUC optimization with logistic loss that directly maximizes the AUC. We also develop an efficient algorithm by adopting the *k*-means clustering and PGD algorithm to find the solution of the proposed method. Through numerical experiments, we demonstrate the promising performance of the proposed method in terms of prediction accuracy, variable selection, and running time compared with the existing classifiers.

In addition to the linear classifier for AUC-maximization with the logistic loss, the extension to nonlinear classifiers such as the tree-based model, generalized additive model, kernel model, or even deep neural network model will be possible. It is very challenging due to its high computational complexity, but we believe it would be interesting for further research. Another challenging task is that most clustering algorithms suffer from the curse of dimensionality, which may deteriorate the performance of the CPAUC algorithm when the predictor dimension is very large. To avoid this, we will try to adopt dimension reduction clustering techniques (Bouveyron and Brunet-Saumard, 2014; Cohen *et al.*, 2015) for the proposed method, which requires further investigation.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00242528).

References

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

Arzhaeva Y, Duin RP, and Tax D (2006). Linear model combining by optimizing the area under the ROC curve, *Proceedings of 18th International Conference on Pattern Recognition (ICPR'06)*, Vol.4, IEEE, 119–122.

Ataman K, Street WN, and Zhang Y (2006). Learning to rank by maximizing AUC with linear programming, *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, IEEE, pp. 123–129.

Bouveyron C and Brunet-Saumard C (2014). Model-based clustering of high-dimensional data: A review, *Computational Statistics & Data Analysis*, **71**, 52–78.

Brefeld U and Scheffer T (2005). AUC maximizing support vector learning, *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*.

Clémençon S, Depecker M, and Vayatis N (2013). An empirical comparison of learning algorithms for nonparametric scoring: The treerank algorithm and other methods, *Pattern Analysis and Applications*, **16**, 475–496.

Clémençon S, Lugosi G, and Vayatis N (2008). Ranking and empirical minimization of U-statistics, *The Annals of Statistics*, **36**, 844–874.

Cohen MB, Elder S, Musco C, Musco C, and Persu M (2015). Dimensionality reduction for *k*-means clustering and low rank approximation, *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, 163–172.

Combettes PL and Wajs VR (2005). Signal recovery by proximal forward-backward splitting, *Multiscale Modeling & Simulation*, **4**, 1168–1200.

Cortes C and Vapnik V (1995). Support-vector networks, *Machine Learning*, **20**, 273–297.

Duda RO, Hart PE, and Stork DG (1973). *Pattern Classification and Scene Analysis*, Vol.3, Wiley New York.

Egan JP (1975). Signal detection theory and ROC analysis, (No Title), Available from: https://www.amazon.com/Detection-Analysis-Academic-Cognition-Perception/dp/0122328507

Feldman V, Guruswami V, Raghavendra P, and Wu Y (2012). Agnostic learning of monomials by halfspaces is hard, *SIAM Journal on Computing*, **41**, 1558–1590.

Gao W and Zhou Z-H (2015). On the consistency of AUC pairwise optimization, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Kim D and Shin SJ (2020). The regularization paths for the ROC-optimizing support vector machines, *Journal of the Korean Statistical Society*, **49**, 264–275.

Kim H, Sohn I, and Shin SJ (2021). Regularization paths of L1-penalized ROC curve-optimizing support vector machines, *Stat*, **10**, e400.

Lei Y and Ying Y (2021). Stochastic proximal AUC maximization, *The Journal of Machine Learning*

Research, **22**, 2832–2876.

Liu M, Yuan Z, Ying Y, and Yang T (2019). Stochastic auc maximization with deep neural networks, Available from: arXiv preprint arXiv:1908.10831

Menon AK and Williamson RC (2016). Bipartite ranking: A risk-theoretic perspective, *The Journal of Machine Learning Research*, **17**, 6766–6867.

Natole M, Ying Y, and Lyu S (2018). Stochastic proximal algorithms for AUC maximization, *International Conference on Machine Learning, PMLR*, **80**, 3710–3719.

Norton M and Uryasev S (2019). Maximization of auc and buffered auc in binary classification, *Mathematical Programming*, **174**, 575–612.

Rakotomamonjy A (2004). Optimizing area under Roc curve with SVMs, *ROCAI*, 71–80.

Rockafellar RT (1976). Monotone operators and the proximal point algorithm, *SIAM Journal on Control and Optimization*, **14**, 877–898.

Rousseeuw PJ (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *Journal of Computational and Applied Mathematics*, **20**, 53–65.

Thorndike RL (1953). Who belongs in the family?, *Psychometrika*, **18**, 267–276.

Tian Y, Shi Y, Chen X, and Chen W (2011). AUC maximizing support vector machines with feature selection, *Procedia Computer Science*, **4**, 1691–1698.

Tibshirani R (1996). Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **58**, 267–288.

Uematsu K and Lee Y (2017). On theoretically optimal ranking functions in bipartite ranking, *Journal of the American Statistical Association*, **112**, 1311–1322.

Yang Z, Shen W, Ying Y, and Yuan X (2020). Stochastic AUC optimization with general loss, *Communications on Pure & Applied Analysis*, **19**, 4191–4212.

Ying Y, Wen L, and Lyu S (2016). Stochastic online AUC maximization, *Advances in Neural Information Processing Systems*, **29**.

Zhang X, Saha A, and Vishwanathan S (2012). Smoothing multivariate performance measures, *The Journal of Machine Learning Research*, **13**, 3623–3680.

Zhao P, Hoi SC, Jin R, and Yang T (2011). Online AUC maximization, Available from: https://icml.cc/2011/papers/198_icmlpaper.pdf

Zhu J, Rosset S, Tibshirani R, and Hastie T (2003). 1-norm support vector machines, *Advances in Neural Information Processing Systems*, **16**.

Received January 4, 2024; Revised February 22, 2024; Accepted February 23, 2024