PAUCInt: Rank-free pAUC Optimization with Approximated Integrals

Huiyang Shao and Weigang Zhang*

***Abstract*—The Partial Area Under the ROC Curve (pAUC) measures the average performance of a binary classifier within a specific false positive rate interval. It has attracted rising attention since it is closer to the actual requirements of many applications such as medical diagnosis and financial fraud detection requiring false positive rates to be lower than certain thresholds. Existing pAUC optimization methods face the following challenges: 1) the earlier deterministic algorithms are hard to optimize in an efficient manner since they rely on the score ranking operation; 2) the latest stochastic algorithms have large approximation error or slow convergence rate. To eliminate these problems, under the normality assumption, this paper presents the first trial to optimize pAUC based on approximated integral in an efficient way. We propose an approximated pAUC estimator based on integral, which escapes from the pairwise comparison and ranking. Meanwhile, we propose two efficient algorithms for both full-batch and mini-batch settings. According to stochastic composition optimization theory, our proposed stochastic algorithm has the convergence rate of $\mathcal{O}(1/\sqrt{k})$. Furthermore, we prove that when the number of samples is large enough, the empirical version of our estimator is an unbiased estimation of the population version under the limited distribution. Finally, extensive experiments show that our algorithm performs better on datasets of various scales.**

***Index Terms*—partial AUC, machine learning, ROC**

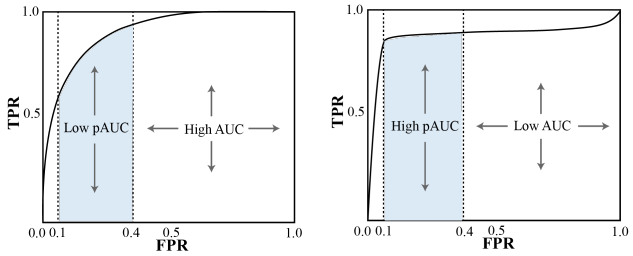
Receiver Operating Characteristics (ROC) is a popular tool to study the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) of a scoring function under different thresholds. The Area Under ROC Curve (AUC) [?], [?], [?] naturally measures the average classification performance under different thresholds and is widely used as a classification metric [?]. Compared with accuracy, AUC has some advantageous properties in that it is insensitive to class distribution and costs. Therefore, AUC is preferred in many applications, especially in faced with imbalanced distribution, *e.g.*, disease prediction [?] and anomaly detection [?]. In the past two decades, the remarkable success of AUC optimization has been witnessed [?], [?], [?].

Nonetheless, researchers try to step further on AUC to study the area under ROC curve within a given FPR interval. It is important and probably deserves more attention in some applications. For example, in medical diagnosis, maintaining a low FPR is essential to avoid unnecessary and expensive biopsies [?]. In email detection systems, a low FPR is necessary to prevent legitimate emails from being identified as spams [?]. For these applications, there is a favor of using pAUC [?] within a specific FPR range compared to the standard AUC.

Huiyang Shao is with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy, Beijing, 100190, China (email: shaohuiyang21@mails.ucas.ac.cn).

Weigang Zhang is with the School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China (email: wgzhang@hit.edu.cn).

*Corresponding authors.



(a) Entire AUC optimization (b) pAUC optimization

Fig. 1. Comparisons between the entire AUC optimization and pAUC optimization.

As is illustrated in Fig. 1, entire AUC optimization may result in an entire high AUC but a low pAUC value, showing that entire AUC optimization does not necessarily ensure a good pAUC. Therefore, directly optimizing pAUC is both important and necessary and has attracted growing attention recently [?], [?], [?], [?], [?], [?], [?].

In order to optimize pAUC, one typically faces the following two challenges. First, for each iteration, negative instances need to be sorted according to prediction scores and then selected to calculate pAUC in a given FPR interval. The ranking operation makes it unable to optimize model parameters based on gradient (ranking operation is not differentiable) [?], [?], [?], [?]. Although some studies propose a novel formulation for pAUC to solve the non-differentiable problem [?], [?], [?], it will impose an approximation error or slow convergence rate (excessive auxiliary variables). Second, some methods are based on the unbiased estimator [?], [?], [?] that pAUC is equivalent to the probability that a random positive instance is scored higher than negative instances in a specified FPR interval. In this formulation, one has to calculate positive/negative sample pairs to optimize approximate pAUC, resulting in a high time complexity ($O(N_+ \cdot N_-)$ where N_+/N_- is the number of positive/negative instances). This causes unacceptable running time for large-scale datasets.

Existing pAUC optimization methods, which could be roughly divided into parametric and non-parametric methods, fail to tackle the above two challenges. Some earlier non-parametric algorithms [?], [?], [?], [?], [?] are typically based on the unbiased estimator of pAUC. This line of methods suffers from both the above two issues. By contrast, the parametric methods [?], [?] which make assumptions on the distribution of sample scores, need not calculate sample pairs,

and can avoid the high complexity. However, this line of methods remains less explored for pAUC. To the best of our knowledge, the current parametric methods still do not allow efficient training due to the ranking operation.

In this paper, we assume that all the data follow a normal distribution. We give a simple explicit form of pAUC based on approximated integral of score distributions to eliminate the problems of both ranking operation and pairwise computation so that we can efficiently train the model. An efficient optimization algorithm that allows stochastic training is also proposed.

Specifically, the main contributions of this paper are as three-fold:

- We present the first trial to optimize pAUC based on approximated integral in an efficient way, effectively solving the problems of ranking and pairs calculation. Specifically, we develop a parametric loss function for cumulative distribution to approximate pAUC.
- We propose two efficient algorithms to optimize pAUC for full-batch and mini-batch setting. Theoretically, our algorithm is well supported by asymptotic normality analysis. Meanwhile, we prove that the mini-batch learning algorithm has a convergence rate of $\mathcal{O}(1/\sqrt{k})$.
- We conduct extensive experiments on multiple imbalanced binary classification tasks. The experimental results speak to the effectiveness and stability of our proposed methods.

I. Related Work

The partial area under the ROC curve (pAUC) was first proposed by [**?**]. In the past two decades, there has been much work on developing algorithms to optimize the pAUC, mainly in the context of ranking and pairs. Depending on the type of optimization method, most of the studies on pAUC fall into two main categories:

Earlier Studies. In the early research phase of pAUC optimization, most studies adopt a linear model to optimize pAUC. [**?**] propose a non-parametric rank-baed pAUC estimator and consider finding the optimal linear combinations of features as the final classifier to maximize the partial region of AUC. [**?**] present a robust estimator for pAUC without any parametric assumptions. They propose a regression modeling framework to analyze covariate effects on the pAUC. Additionally, they show that there is only a little loss in efficiency while obtaining robustness. [**?**] take the false positive rate into objective function and develop the concept of Asymmetric Support Vector Machine (ASVM), which aims to obtain the optimal classifier with a lower false positive rate. They achieve this by maximizing the core-margin (*i.e.*, the margin between the top score subset of the positive class and the negative class) and class-margin (*i.e.*, the margin of traditional SVM). However, these earlier methods have large approximation errors in estimating pAUC.

In a boosting way, [**?**] continuously adds weak classifiers to optimize pAUC and updates coefficients with a one-step Newton-Raphson iteration. [**?**] expand weak classifiers and calculate objective function based on Support Vector Machine (SVM). However, boosting algorithm performs poorly on large datasets with high variance and highly depends on weak classifiers. By breaking down a single optimization problem into more minor, single feature selection problems, [**?**] propose an algorithm to get the best feature and coefficient based on the empirical pAUC for each iteration. Finally, a linear combination of features is output as the classifier. [**?**] aims to decompose the problem into sub-problems and optimize them by the cutting plane method. [**?**] overcome the disadvantage of sensitivity to outliers in the data by using a structural SVM and a non-convex ramp loss as the objective function. Nevertheless, the main problem comes from the algorithmic complexity of the structural SVM, which requires calculating data pairs and solving quadratic programming with massive constraints.

In [**?**], they present the first trial to optimize pAUC in an online learning way. Their algorithm enables incremental learning based on batch data. However, this heuristic method still can not solve the ranking problem. Meanwhile, their algorithm is not guaranteed to converge to optimal pAUC. [**?**] introduce two types of nonlinear scoring functions to fit different models. Combining with approximated pAUC objective function, they show that classifiers trained with these nonlinear scoring functions have better performance, which surpasses other competitors significantly. In recent years, there has been an increasing number of pAUC studies; a partial list includes [**?**], [**?**], [**?**], [**?**], [**?**]. Nevertheless, there is no convergence analysis guarantee for their algorithms. In addition, ranking operations still can not be eliminated.

Deep pAUC Maximization. Since the ranking operation is not differentiable, the traditional pAUC maximization methods can not be applied to deep learning. In order to eliminate the FPR constraints in pAUC maximization, employing the Implicit Function Theorem, [**?**] modeled an implicit function that takes the model parameters as the input and FPR quantile threshold as the output. In this way, they can calculate the gradient of quantile and update it by gradient descent. Therefore, the original constrained problem is reformulated as a single-level optimization problem. However, extensive constraints make optimization very difficult.

As a milestone study, [**?**] converts the pAUC maximization into a bi-level optimization problem. The inner level optimization achieves top-ranked instances selection, and the outer level optimization focuses on minimizing the objective function. They present the first trial to design a formulation for pAUC which applies to deep learning. However, their proposed estimator may suffer from an unacceptable approximation error. Recently, [**?**] propose two estimators of pAUC which can be applied to deep learning and give a sound theoretical convergence analysis guarantee. Nevertheless, due to the fact that their proposed exact pAUC estimator needs $O(N^+)$ auxiliary variables, their algorithm can not update these variables in time. Additionally, their proposed soft estimator need suitable hyper-parameters to approximate pAUC, which means it has an approximation error in most situation. [**?**] consider casting the pAUC optimization into a non-smooth difference-of-convex (DC) problem, which allowed them to use an approximated gradient descent method to solve it. However, their algorithms have a slow convergence rate ($\tilde{O}(\epsilon^{-6})$ where

ϵ is the tolerance).

Distinguish from most existing studies, this paper focuses on designing an efficient pAUC algorithm based on parametric assumptions. To summarize, The existing algorithms prefer to use the non-parametric estimator of pAUC to optimize the classifier. Unlike these, we propose an approximate integral as an objective function for optimizing pAUC based on data distribution assumption. Our proposed estimator enjoys faster convergence with acceptable approximation error.

II. Preliminary

In the binary classification setting, denote $\mathcal{X} = \{\mathcal{X}^+, \mathcal{X}^-\} \subseteq \mathbb{R}^d$ be an d-dimensional instance space, where positive sample $\boldsymbol{x}^+ \in \mathcal{X}^+$ and negative sample $\boldsymbol{x}^- \in \mathcal{X}^-$. Let $\mathcal{D}^+$, $\mathcal{D}^-$ be the distributions of positive instances and negative instances, respectively. The goal is to learn a scoring function $f : \mathcal{X} \mapsto \mathbb{R}$ such that instances are classified into positive/negative based on $\mathrm{sign}(f(\boldsymbol{x}) - t)$. Herein, t is certain classification threshold, $\mathrm{sign}(x) = 1$ if $x > 0$ and $\mathrm{sign}(x) = 0$ if $x \leq 0$.

Now, we review the formal definitions of TPR, FPR and give the integral forms of AUC and pAUC. For specified threshold t, the TPR of a classifier derived from $f(\boldsymbol{x})$ measures the likelihood that it accurately predicts a positive instance when getting a random positive instance from $\mathcal{D}^+$. Formally, we have:

$$\mathrm{TPR}_f(t) = \mathbb{P}_{\boldsymbol{x}^+ \sim \mathcal{D}^+}\Big[f\left(\boldsymbol{x}^+\right) > t\Big]. \tag{1}$$

In a similar spirit, the classifier's FPR on threshold t refers to the probability that it predicts positive when it gets a negative instance from $\mathcal{D}^-$. That is,

$$\mathrm{FPR}_f(t) = \mathbb{P}_{\boldsymbol{x}^- \sim \mathcal{D}^-}\Big[f\left(\boldsymbol{x}^-\right) > t\Big]. \tag{2}$$

As AUC is the area under the ROC curve, we have that AUC is then computed by the integral of $\mathrm{TPR}_f(t)$ against the $\mathrm{FPR}_f(t)$ for various values of threshold t as follows:

$$\mathrm{AUC}_f = \int_0^1 \mathrm{TPR}_f\left(\mathrm{FPR}_f^{-1}(u)\right) du. \tag{3}$$

where $\mathrm{FPR}_f^{-1}(u) = \{t \in \mathbb{R} | \mathrm{FPR}_f(t) = u\}$ is the inverse function of $\mathrm{FPR}_f(t)$.

In this paper, we are interested in the area under the curve between FPR in the range $[\alpha, \beta]$. In this integral form, the pAUC of f is then calculated by:

$$\mathrm{pAUC}_f(\alpha, \beta) = \frac{1}{\beta - \alpha} \int_\alpha^\beta \mathrm{TPR}_f\left(\mathrm{FPR}_f^{-1}(u)\right) du. \tag{4}$$

III. Methodology

From the above-mentioned definition of pAUC, it is obvious that if TPR and FPR do not have explicit functions, pAUC is not integrable. In our approach, we propose to formulate TPR and FPR based on score distributions.

Considering a linear scoring function $f = \boldsymbol{w}^T\boldsymbol{x}$, where $\boldsymbol{w} \in \mathbb{R}^d$ is the model parameters, according to our assumption that $\mathcal{D}^+$ and $\mathcal{D}^-$ are multivariate normal, positive scores $s_f^+ = \boldsymbol{w}^T\boldsymbol{x}^+$ (*resp*, negative scores $s_f^- = \boldsymbol{w}^T\boldsymbol{x}^-$) satisfy $s_f^+ \sim \mathcal{D}_f^+ = \mathcal{N}(\mu_p, \sigma_p)$ (*resp*, $s_f^- \sim \mathcal{D}_f^- = \mathcal{N}(\mu_n, \sigma_n)$). Herein, μ_p and σ_p (*resp*, μ_n and σ_n) are the mean and standard deviation of the normal distribution for positive scores (*resp*, negative scores). When the probability density distribution of score is known, TPR can be understood as the cumulative probability of positive scores from threshold t to ∞. Denote $L_1 = \frac{s_f^+ - \mu_p}{\sqrt{2}\sigma_p}$, we have:

$$\begin{aligned}
\mathrm{TPR}_f(t) &= \mathbb{P}_{s_f^+ \sim \mathcal{D}_f^+}\left[s_f^+ > t\right] \\
&= \int_t^\infty \frac{1}{\sqrt{2\pi}\sigma_p} \exp\left(-L_1^2\right) ds_f^+ \\
&= \int_0^\infty -\frac{1}{\sqrt{2\pi}\sigma_p} 2L_1 \exp\left(-L_1^2\right) dL_1 \\
&\quad - \int_0^t -\frac{1}{\sqrt{2\pi}\sigma_p} 2L_1 \exp\left(-L_1^2\right) dL_1 \\
&= \frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\left(\frac{t - \mu_p}{\sqrt{2}\sigma_p}\right),
\end{aligned} \tag{5}$$

According to definition of cumulative distribution function of normal, $\mathrm{erf}(x)$ should be

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{6}$$

Similarly, FPR can be understood as the cumulative probability of negative scores from threshold $-\infty$ to t:

$$\begin{aligned}
\mathrm{FPR}_f(t) &= \mathbb{P}_{s^n \sim \mathcal{D}_f^-}\left[s^n > t\right] \\
&= \frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\left(\frac{t - \mu_n}{\sqrt{2}\sigma_n}\right).
\end{aligned} \tag{7}$$

So far, explicit formulas of TPR and FPR are provided. However, pAUC in Eq. 4 is still hard to calculate since TPR and FPR rely on erf, which do can not be determined directly with elementary functions. Hence, the inverse function FPR^{-1} is hard to calculate, let alone the integral of $\mathrm{TPR}_f\left(\mathrm{FPR}_f^{-1}(u)\right)$. Escaping the restriction that the erf function is not differentiable and computable is essential to our algorithm. In the next subsection, we will proceed to solve this problem.

A. Surrogate Error Function

On top of the explicit formulas of TPR and FPR, these two functions still cannot be calculated directly using the elementary functions. An intuitive solution is to use surrogate functions instead of erf to make TPR and FPR differentiable and computable. In this section, we propose to use a $\hat{\mathrm{erf}}(x)$ instead of $\mathrm{erf}(x)$ to make pAUC easy to calculate. Specifically, we propose the following surrogate error function to estimate $\mathrm{erf}(x)$:

$$\hat{\mathrm{erf}}(x) = \begin{cases} -1, & x < -\frac{\pi}{2} \\ \sin(x), & -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\ 1, & x > \frac{\pi}{2} \end{cases}. \tag{8}$$

This surrogate error function has left and right limits at $x = -\frac{\pi}{2}$ and $x = \frac{\pi}{2}$, so the function is continuous. In addition, this surrogate error function is easier to compute

and is differentiable. Meanwhile, we can easily get the inverse function FPR^{-1}. For fixed FPR u and distribution parameters $\boldsymbol{\theta} = \{\mu_p, \sigma_p, \mu_n, \sigma_n\}$, $\mathrm{FPR}_f^{-1}(u, \boldsymbol{\theta})$ (we denote it as $r(u, \boldsymbol{\theta})$ here) can be calculated as:

$$r(u,\boldsymbol{\theta}) = \begin{cases} \mu_n + \frac{\sqrt{2}\pi\sigma_n}{2} & u = 0 \\ \mu_n + \sqrt{2}\sigma_n \arcsin(1-2u) & 0 < u < 1 \\ \mu_n - \frac{\sqrt{2}\pi\sigma_n}{2} & u = 1 \end{cases}. \quad (9)$$

There is a cost to using the surrogate error function, and if the surrogate function differs too much from true $\mathrm{erf}(x)$, the final pAUC may be biased. To prove that the cost of our proposed surrogate error function is acceptable, we prove the error bound between our surrogate error function and the original $\mathrm{erf}(x)$ is small enough in Appendix.B.

B. Objective Function

In this section, we will establish an approximation of pAUC. We can propose a loss function that optimizes the pAUC to facilitate efficient training by using this approximation. On top of the explicit form of TPR and FPR and the proposed surrogate error function, we are ready to calculate approximate pAUC in Eq. (4) as follows:

$$\begin{aligned} &\mathrm{pAUC}_f(\alpha,\beta) \\ &= \frac{1}{\beta-\alpha}\int_{\alpha}^{\beta} \mathrm{TPR}_f\left(\mathrm{FPR}_f^{-1}(u)\right) du \\ &= \frac{1}{\beta-\alpha}\int_{r(\alpha,\boldsymbol{\theta})}^{r(\beta,\boldsymbol{\theta})} \mathrm{TPR}_f\left(r\right) d\mathrm{FPR}_f(r) \\ &= \frac{-1}{\beta-\alpha}\int_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})} \mathrm{TPR}_f\left(r\right)\cdot\left(\mathrm{FPR}_f(r)\right)' dr \\ &= \frac{-1}{4(\beta-\alpha)}\int_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})} \mathrm{erf}\left(\frac{r-\mu_n}{\sqrt{2}\sigma_n}\right)'\left[\mathrm{erf}\left(\frac{r-\mu_p}{\sqrt{2}\sigma_p}\right)-1\right] dr \\ &\approx \frac{-1}{4(\beta-\alpha)}\int_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})} \hat{\mathrm{erf}}\left(\frac{r-\mu_n}{\sqrt{2}\sigma_n}\right)'\left[\hat{\mathrm{erf}}\left(\frac{r-\mu_p}{\sqrt{2}\sigma_p}\right)-1\right] dr \\ &= \frac{-1}{4(\beta-\alpha)}\int_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})} \psi(r) dr, \end{aligned} \quad (10)$$

where

$$\psi(r) = \hat{\mathrm{erf}}\left(\frac{r-\mu_n}{\sqrt{2}\sigma_n}\right)'\left[\hat{\mathrm{erf}}\left(\frac{r-\mu_p}{\sqrt{2}\sigma_p}\right)-1\right], \quad (11)$$

Note that $r(u, \boldsymbol{\theta})$ is a monotonically decreasing function so the lower limit of integral starts from $r(\beta, \boldsymbol{\theta})$. From the above equations, we see that the remaining task is to calculate the integral of $\psi(r)$ from $r(\beta, \boldsymbol{\theta})$ to $r(\alpha, \boldsymbol{\theta})$. To do this, we first calculate the primitive funtions of $\psi(r)$. Based on Prosthaphaeresis formulas, we derive that $\int \psi(r)du$ is a piecewise function as follows:

- When r satisfies that $-\frac{\pi}{2} \le \frac{r-\mu_n}{\sqrt{2}\sigma_n} \le \frac{\pi}{2}$ and $\frac{r-\mu_p}{\sqrt{2}\sigma_p} < -\frac{\pi}{2}$, we have:

$$\begin{aligned} \int \psi(r)dr &= \int -\frac{\sqrt{2}}{\sigma_n}\cos\left(\frac{r-\mu_n}{\sqrt{2}\sigma_n}\right) dr \\ &= -2\sin\left(\frac{r-\mu_n}{\sqrt{2}\sigma_n}\right) + C, \end{aligned}$$

where C is certain constant.

- When $-\frac{\pi}{2} \le \frac{r-\mu_n}{\sqrt{2}\sigma_n} \le \frac{\pi}{2}$ and $\frac{r-\mu_p}{\sqrt{2}\sigma_p} > \frac{\pi}{2}$, note that $\hat{\mathrm{erf}}\left(\frac{r-\mu_p}{\sqrt{2}\sigma_p}\right) = 1$. Hence, we have:

$$\begin{aligned} \int \psi(r)dr &= \int -\hat{\mathrm{erf}}\left(\frac{r-\mu_n}{\sqrt{2}\sigma_n}\right)'\left[\hat{\mathrm{erf}}\left(\frac{r-\mu_p}{\sqrt{2}\sigma_p}\right)-1\right] dr \\ &= C. \end{aligned}$$

- When $-\frac{\pi}{2} \le \frac{r-\mu_n}{\sqrt{2}\sigma_n} \le \frac{\pi}{2}$ and $-\frac{\pi}{2} \le \frac{r-\mu_p}{\sqrt{2}\sigma_p} \le \frac{\pi}{2}$, for simplicity, denote $L_2 = \frac{r-\mu_n}{\sqrt{2}\sigma_n}$, $\beta_1 = \frac{\sigma_p}{2(\sigma_n+\sigma_p)}$, $\beta_2 = \frac{\sigma_p}{2(\sigma_n-\sigma_p)}$, we will get:

$$\begin{aligned} &\int \psi(r)dr \\ &= \int \frac{1}{\sqrt{2}\sigma_n}\cos\left(L_2\right)\Big[\sin\left(L_1\right)-1\Big]dr \\ &= \frac{1}{2\sqrt{2}\sigma_n}\int \sin\left(L_1+L_2\right)+\sin\left(L_1-L_2\right)dr - \sin(L_1) \\ &= -\beta_1\cos\left(L_1+L_2\right)-\beta_2\cos\left(L_1-L_2\right)-\sin(L_1)+C. \end{aligned}$$

Since the constant C is irrelevant to optimization, we eliminate it in the remaining paper. According to the above calculation, we could easily get the approximation of $\mathrm{pAUC}_f(\alpha, \beta)$. Therefore, instead of maximizing $\mathrm{pAUC}_f(\alpha, \beta)$, we can minimize $-\mathrm{pAUC}_f(\alpha, \beta)$. Let $\mathcal{R}(u; \boldsymbol{\theta}) = \frac{1}{4(\beta-\alpha)}\int \psi(r)dr$, our goal then becomes:

$$\min_{\boldsymbol{\theta}} \mathcal{R}_{\alpha,\beta}(\boldsymbol{\theta}) := \mathcal{R}(u;\boldsymbol{\theta})|_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})} \quad (12)$$

Details of the partial derivatives of $\mathcal{R}(u; \boldsymbol{\theta})$ with respect to each parameter are given in Appendix.C.

IV. Optimization Algorithm

A. Weight Initialization Algorithm

A good initialization guesses for parameter $\boldsymbol{w}$ is very important for optimization. If the initial guess is good enough, we can get the optimal solution after a few iterations. For the initialization of $\boldsymbol{w}_0$, zero vectors or random vectors are not a good choice, so we propose a $\boldsymbol{w}_0$ initialization algorithm for our model.

Algorithm 1 Weight Initialization Algorithm

Input: Dataset $\boldsymbol{X}^+$, $\boldsymbol{X}^-$, feature num d
Output: Initial model parameters $\boldsymbol{w}_0$

```
Initialize the d-dimensional zero vector w_0.
Calculate the mean value of all features μ^+/μ^- of the dataset X^+/X^-.
for k = 1, 2, ··· , d do
    if μ^+[k] > μ^-[k] then
        w_0[k] = random() // random() means randomly sample a real number from the interval [0,1]
    else
        w_0[k] = -1*random()
    end if
end for
return w_0
```

Algorithm 2 Deterministic Algorithm For pAUC Optimization

Input: Dataset $\boldsymbol{X}$, α, the maximum iteration number K
Output: Final model parameter $\boldsymbol{w}^*$

1: Initialize model parameters $\boldsymbol{w}_0$, set $\hat{\boldsymbol{\theta}}_0$.
2: **for** $k = 1, 2, \cdots, K$ **do**
3: $\mu_p = \boldsymbol{w}_{k-1}^T \bar{\boldsymbol{x}^+}$
4: $\mu_n = \boldsymbol{w}_{k-1}^T \bar{\boldsymbol{x}^-}$
5: $\sigma_p = \boldsymbol{w}_{k-1}^T \Sigma_p \boldsymbol{w}_{k-1}$
6: $\sigma_n = \boldsymbol{w}_{k-1}^T \Sigma_n \boldsymbol{w}_{k-1}$
7: $\hat{\boldsymbol{\theta}}_k = \{\mu_p, \mu_n, \sigma_p, \sigma_n\}$
8: Calculate $\mathcal{R}_{\alpha,\beta}(\hat{\boldsymbol{\theta}}_k)$.
9: Update $\boldsymbol{w}$ with algorithm $\mathcal{A}$: $\boldsymbol{w}_k = \mathcal{A}(\mathcal{R}_{\alpha,\beta}(\hat{\boldsymbol{\theta}}_k))$
10: Loop until $\left\|\nabla \mathcal{R}_{\alpha,\beta}(\hat{\boldsymbol{\theta}}_k)\right\|_2 \leq \epsilon_\nabla$.
11: **end for**
12: **return** $\boldsymbol{w}^*$

B. Deterministic Algorithm

In traditional machine learning applications, deterministic algorithms do not suffer from gradient estimation error problems compared to stochastic ones. For small-scale datasets, deterministic algorithms are the first choice for optimization in practice. Therefore, we propose a deterministic algorithm for pAUC optimization in this subsection. As is illustrated in Alg.2, steps 3-6 aim to get an unbiased estimation of mean and variance, and steps 7-8 aim to calculate the loss function. For step 9, $\mathcal{A}$ can be replaced by any unconstrained optimization algorithm (*i.e.*, Gradient Descent, BFGS, L-BFGS-B). Using the unbiased gradient, we can update parameter $\boldsymbol{w}$ by $\mathcal{A}$ until step 10 is satisfied.

C. Stochastic Algorithm

The deterministic algorithm is not feasible for large-scale datasets or online learning scenarios. One effective way is to sample a batch of a dataset for optimization. In this section, we will introduce a novel stochastic algorithm with fast convergence for pAUC optimization.

Though the objective function Eq. (12) is continuous and differentiable, for mini-batch data, an optimization challenge here is that the objective function indicates a two-level compositional optimization problem. Since the parameters of score distribution (eg., mean and variance) are not known, one has to first estimate them, and then calculate the approximate pAUC. Specifically, let $\boldsymbol{\theta}$ take the form of expected values:

$$\boldsymbol{\theta} = \mathbb{E}[g_{\boldsymbol{x}^+,\boldsymbol{x}^-}(\boldsymbol{w})], \tag{13}$$

where the function $g_{\boldsymbol{x}^+,\boldsymbol{x}^-}(\boldsymbol{w})$ is parameterized by random samples $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$. Mathematically, the objective in Eq. (12) is equivalent to:

$$\min_{\boldsymbol{w}\in\mathcal{R}^n} \mathcal{R}_{\alpha,\beta}(\mathbb{E}[g_{\boldsymbol{x}^+,\boldsymbol{x}^-}(\boldsymbol{w})]). \tag{14}$$

This is exactly a stochastic compositional optimization problem [**?**], whose inner composition involves additional expectation. For solving this problem, a feasible attempt is to use sample average to estimate the expectation. However, this requires all training samples during each iteration, which is inefficient and unacceptable for large-scale datasets. Practically, we hope that $\boldsymbol{\theta}$ could be estimated accurately in the popular mini-batch setting.

Algorithm 3 Stochastic Compositional Gradient descent For pAUC Optimization

Input: Dataset $\boldsymbol{X}$, α, learning rate η_1, η_2 and hyper-parameters $\kappa \in (0, 1]$, the maximum iteration number K
Output: Final model parameter $\boldsymbol{w}^*$

1: Initialize model parameters $\boldsymbol{w}_0$, set $\hat{\boldsymbol{\theta}}_0, \boldsymbol{h}_0, \boldsymbol{v}_0 = \boldsymbol{0}$.
2: **for** $k = 1, 2, \cdots, K$ **do**
3: Sample mini-batch instances Ω_k.
4: Calculate $g_{\Omega_k}(\boldsymbol{w}_k)$ and $\nabla g_{\Omega_k}(\boldsymbol{w}_k)$ by Eq. (39) in Appendix.A.
5: Set $\beta_k = \frac{1}{\sqrt{k}}$ and update the estimated $\hat{\boldsymbol{\theta}}_k$ by: $\hat{\boldsymbol{\theta}}_k = (1-\beta_k)\hat{\boldsymbol{\theta}}_{k-1} + \beta_k g_{\Omega_k}(\boldsymbol{w}_{k-1})$
6: Update $\boldsymbol{w}_k$ by the following ADAM-type rules:
7: $\boldsymbol{\nabla}_k = \nabla g_{\Omega_{k-1}}(\boldsymbol{w}_k) \nabla \mathcal{R}_{\alpha,\beta}(\hat{\boldsymbol{\theta}}_k)$.
8: $\boldsymbol{h}_k = \eta_1 \boldsymbol{h}_{k-1} + (1-\eta_1)\boldsymbol{\nabla}_k$.
9: $\boldsymbol{v}_k = \eta_2 \hat{\boldsymbol{v}}_{k-1} + (1-\eta_2)\boldsymbol{\nabla}_k^2$.
10: $\hat{\boldsymbol{v}}_k = \max\{\boldsymbol{v}_k, \hat{\boldsymbol{v}}_{k-1}\}$.
11: $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \kappa \cdot \beta_k \cdot \frac{\boldsymbol{h}_k}{\sqrt{\epsilon + \hat{\boldsymbol{v}}_k}}$.
12: Loop until $\left\|\nabla \mathcal{R}_{\alpha,\beta}(\hat{\boldsymbol{\theta}}_k)\right\|_2 \leq \epsilon_\nabla$.
13: **end for**
14: **return** $\boldsymbol{w}^*$

Motivated by the stochastic compositional gradient descent (SCGD) algorithm, we propose a stochastic optimization algorithm for mini-batch learning. The key idea is that the new estimation is a weighted average of historical estimation and incremental estimation. Specifically, suppose the mini-bach of samples in k-th iteration is Ω_k, $\hat{\boldsymbol{\theta}}_k$ is calculated as:

$$\hat{\boldsymbol{\theta}}_k = (1-\beta_k)\hat{\boldsymbol{\theta}}_{k-1} + \beta_k g_{\Omega_k}(\boldsymbol{w}_k) \tag{15}$$

where $g_{\Omega_k}(\boldsymbol{w}_k)$ is incremental estimation of $\boldsymbol{\theta}$ and β_k is the weight. The specific formulation of $g_{\Omega_k}(\boldsymbol{w}_k)$ is introduced in Appendix.A.

On top of $\hat{\boldsymbol{\theta}}_k$, the algorithms then calculates $\mathcal{R}_{\alpha,\beta}(\hat{\boldsymbol{\theta}}_k)$ and update model parameters. The detailed procedure is depicted in Alg. 3. [**?**] have studied how fast the non-stationary metric $\nabla \mathcal{R}_{\alpha,\beta}$ approaches to zero. Following their studies, under our algorithm setting (inner function and outer function are smooth and differentiable), our proposed algorithm is guaranteed with a convergence rate of $\mathcal{O}(1/\sqrt{k})$, where k is the number of iterations.

V. Theoretical analysis

This section will theoretically analyze the convergence property of the Algorithm. We first give the following definition:

Definition 1. *Let $\boldsymbol{X}^+ \subseteq \mathbb{R}^{N_+ \times d}, \boldsymbol{X}^- \subseteq \mathbb{R}^{N_- \times d}$ be positive instances and negative instances, where N_+ and N_- are the number of positive samples and negative samples, respectively, d is dimension. Let*

$$\bar{S}_+ = \frac{1}{N_+}\sum_{i=1}^{N_+} \boldsymbol{w}^T \boldsymbol{x}_i^+, \bar{S}_- = \frac{1}{N_-}\sum_{j=1}^{N_-} \boldsymbol{w}^T \boldsymbol{x}_j^-, \tag{16}$$

$$\bar{V}_+ = \frac{1}{N_+}\sum_{i=1}^{N_+}(\boldsymbol{w}^T\boldsymbol{x}_i^+)^2, \bar{V}_- = \frac{1}{N_-}\sum_{j=1}^{N_-}(\boldsymbol{w}^T\boldsymbol{x}_j^-)^2. \quad (17)$$

Then we have the following theorem.

Theorem 1. *Let $\boldsymbol{\theta}_S = \{\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-\}$ and $\boldsymbol{\theta}_S^* = \mathbb{E}[\boldsymbol{\theta}_S]$, $\tilde{\boldsymbol{\Sigma}}$ be the covariance matrix of $\boldsymbol{\theta}_S$. Due to $\boldsymbol{\theta}$ can be derive from $\boldsymbol{\theta}_S$, we construct a function $\phi : \mathbb{R}^4 \to \mathbb{R}$ such that has the same output as $\mathcal{R}(u;\boldsymbol{\theta})|_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})}$ with $\boldsymbol{\theta}_S$ as input. So we have estimated pAUC like: $-\hat{\text{pAUC}} = \phi(\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-)$. When the input is $\boldsymbol{\theta}_S^*$, we can obtain the optimal approximated pAUC: $-\text{pAUC}^* = \phi(\boldsymbol{\theta}_S^*)$. There exists a linear map $\nabla\phi_{\boldsymbol{\theta}_S^*} : \mathbb{R}^4 \to \mathbb{R}$ such that*

$$\nabla\phi_{\boldsymbol{\theta}_S^*} = \left[\frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\bar{S}_+}, \frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\sigma_p}\frac{\partial\sigma_p}{\partial\bar{V}_+}, \frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\bar{S}_-}, \frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\sigma_n}\frac{\partial\sigma_n}{\partial\bar{V}_-}\right],$$

*according to the delta method [**?**], we have:*

$$[\phi(\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-) - \phi(\boldsymbol{\theta}_S^*)] \xrightarrow{D} \mathcal{N}\left(0, \nabla\phi_{\boldsymbol{\theta}_S^*}^T\tilde{\boldsymbol{\Sigma}}\nabla\phi_{\boldsymbol{\theta}_S^*}\right).$$

Thm 1 shows that $\phi(\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-)$ is an unbiased estimation of $\phi(\boldsymbol{\theta}_S^*)$ under the limited distribution. Please see Appendix.D for the proof. Additionally, Appendix.A shows details of $\nabla\phi_{\boldsymbol{\theta}_S^*}$.

VI. Experiment

In this section, we conduct experiments on different datasets under both full-batch and mini-batch settings.

A. Experimental Setup

1) Dataset Description: Note that pAUC is aimed at processing the binary classification problems, hence we filter 19 binary datasets diverse in terms of imbalance ratio and sample size from UCI[1] and libsvm[2]. Tab. III lists the statistics for them. We randomly divided the dataset into a training set and a testing set 15 times with a ratio of 0.8:0.2. We apply python package *sklearn.preprocessing.MinMaxScaler* to standardize original data.

2) Competitors.: We compare our algorithm with both earlier deterministic pAUC optimization algorithms: PAUCBoost [**?**]; SvmPAUC [**?**]; MSPAUC [**?**]; GMMPAUC [**?**], and the lateset stochastic pAUC algorithms: pAUC-poly (poly calibrated weighting function) [**?**]; pAUC-exp (exp calibrated weighting function) [**?**]; the DRO formulation of PAUC, which are denoted as SOPA (exact estimator) [**?**] and SOPA-S (soft estimator) [**?**].

3) General Implementation Details.: All experiments are conducted on a Ubuntu 16.04.1 server equipped with an Intel(R) Xeon(R) Silver 4110 CPU and a TITAN RTX GPU, and all codes are developed in Python 3.8 environment. For our proposed deterministic algorithm PAUCInt, we use `scipy.optimize.minimize` as an optimizer to optimize the objective function. For the stochastic algorithm PAUCIntm, we use Alg. 3 to optimize the model. We set the batch size to 64 and the maximum number of iterations to $40 \cdot \frac{\#\text{samples}}{\#\text{batch size}}$. Let $\eta_1 = 0.1$, $\eta_2 = 0.2$, $\kappa = 0.2$. All experiments for method SvmPAUC are conducted under python package `qpsolvers`. MSpAUC and GMMPAUC are optimized by python package `scipy` with limited memory `BFGS` method. The maximum allowed runtime of all algorithms on a single dataset is two days, and if this time is exceeded, the algorithm is forced to stop and calculate the pAUC. With a `Sigmoid` function, the output is scaled into $[0, 1]$. For each dataset, all methods are warmed up with the same weight $\boldsymbol{w}$ which is initialized by Alg. 1.

For the sake of fairness, we use 5-fold cross-validation to choose the best parameters for all algorithms. Furthermore, in order to avoid randomness, all methods are repeated five times on the same training and testing set. The final result takes the average value based on all the divided datasets. We present two sets of experiments, a comparison of our proposed deterministic algorithm with earlier work and a comparison of our proposed stochastic algorithm with the latest pAUC optimization algorithm.

B. Parameter Tuning

For our proposed algorithms PAUCInt and PAUCIntm, we set the tolerance $\epsilon_\nabla = 0.00001$, the maximum iteration number $K = 200$, learning rate $\eta_1 = 0.1 \cdot k^{-1/2}$, $\eta_2 = k^{-1/2}$ and $\kappa = 0.2$. For SvmPAUC, we set the trade-off parameter $C \in 2^{[-2:1:4]}$, soft margin $\xi \in [0.1 : 0.05 : 0.3]$, maximum iteration number for each cutting plane process $M = 100$. For MSPAUC, we set control parameter $\lambda = 0.5$, smooth parameter $h_1 \in [0.1 : 0.05 : 0.5]$, $h_2 \in [0.2 : 0.05 : 0.7]$. For PAUCBoost, we set $\lambda \in [0.05 : 0.05 : 0.5]$ and the maximum order $m = 6$. For the stochastic optimization experiment, the learning rate of all methods is tuned in $[10^{-2}, 10^{-5}]$, and the number of epochs is set as 200. For pAUC-poly and pAUC-exp, E_k is searched in $\{1, 5, 7, 9, 12, 15\}$. Specifically, for pAUC-poly, γ is searched in $\{0.01, 0.05, 0.09, 0.3, 0.9, 1, 3, 5\}$. For AUC-exp, γ is searched in $\{0.02, 0.06, 0.1, 0.4, 0.8, 2, 4, 6\}$. For SOPA-S, we tune the KL-regularization parameter λ in $\{0.1, 1.0, 10\}$, and we fix $\beta_0 = \beta_1 = 0.9$.

C. Results and Analysis

Tab. I (*resp*, Tab. II) shows the scores of all deterministic (*resp*, stochastic) algorithms on different benchmark datasets. For the HIGGS dataset, SvmPAUC, MSPAUC, GMMPAUC, and PAUCBoost exceed the maximum run time limit on a single iteration. Hence we set their scores as -. Fig. 2 (*resp*, Fig. 3) records the convergence curves of all deterministic algorithms (*resp*, stochastic) on training set. Fig. 4 and Fig. 5 demonstrates all results on the testing set of each algorithm on large-scale datasets, respectively. Due to the page limitation, please see the Appendix.A for the entire convergence and stability experiment results. From the results, we make the following remarks:

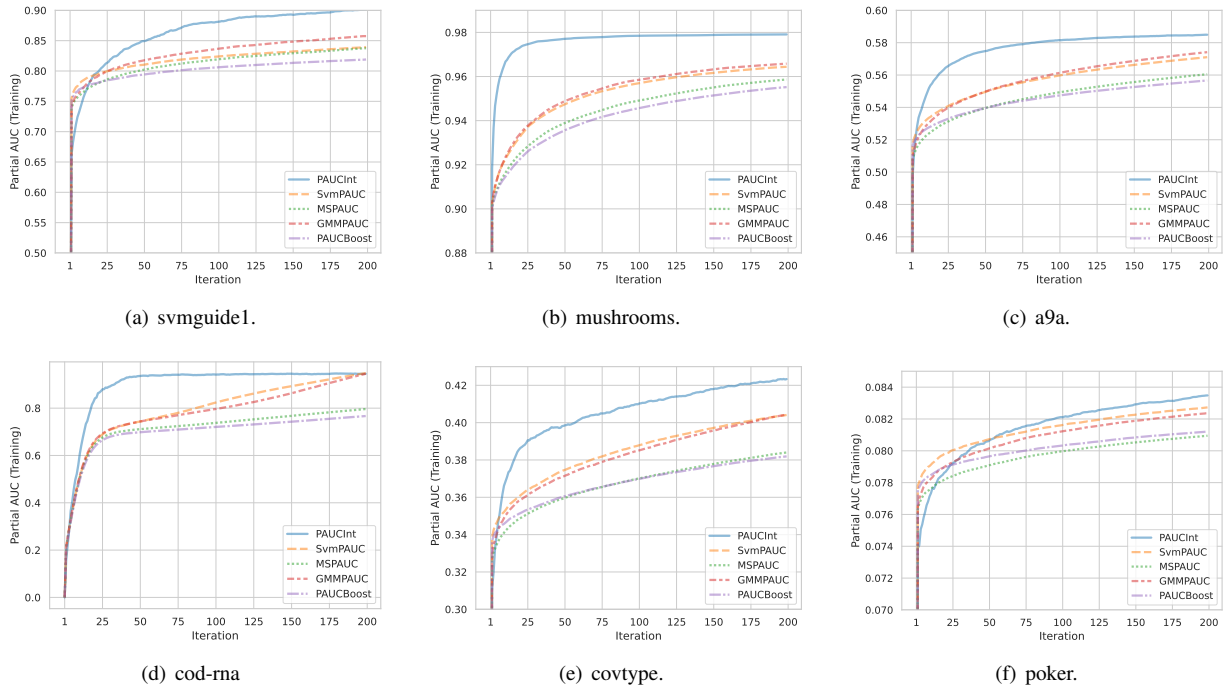
- From Tab. I and Tab. II, we can observe that PAUCInt and PAUCIntm achieve the highest scores on most datasets, confirming that our algorithm is effective and our data normality assumption is reasonable. Even for failure cases, our methods attain fairly competitive results compared with other methods. competitors

[1]https://archive.ics.uci.edu/ml/datasets.php

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

TABLE I
pAUC (MEAN±STD), WHERE FPR IN THE RANGE OF 0.05 TO 0.1. **ALL COMPETITORS ARE SELECTED FROM THE EARLIER STUDIES IN RELATED WORK.** FOR EACH DATASET, THE BEST VALUE IS BOLD, UNDERLINED IS APPLIED THE SECOND.

Scale	Dataset	PAUCInt(ours)	SvmPAUC	MSPAUC	GMMPAUC	PAUCBoost
Small Scale	breast_cancer	**1.000±0.000**	0.984±0.026	0.975±0.017	0.999±0.001	0.990±0.015
	sonar	0.478±0.171	0.493±0.111	0.509±0.108	**0.523±0.102**	0.480±0.097
	ionosphere	**0.551±0.227**	0.509±0.228	0.497±0.199	0.550±0.224	0.452±0.184
	liver-disorders	**0.490±0.163**	0.431±0.146	0.430±0.103	0.447±0.112	0.414±0.140
	diabetes	**0.575±0.065**	0.496±0.075	0.470±0.087	0.513±0.067	0.531±0.068
	fourclass	**0.499±0.050**	0.484±0.053	0.486±0.045	0.495±0.046	0.491±0.044
	australian	**0.740±0.093**	0.696±0.128	0.712±0.110	0.730±0.102	0.722±0.123
Middle Scale	german	**0.377±0.072**	0.317±0.068	0.312±0.079	0.329±0.079	0.304±0.073
	splice	**0.684±0.065**	0.612±0.075	0.617±0.084	0.665±0.088	0.559±0.067
	svmguide3	**0.518±0.063**	0.479±0.050	0.465±0.076	0.486±0.152	0.493±0.067
	heart	**0.690±0.108**	0.619±0.160	0.574±0.143	0.683±0.111	0.576±0.091
	a1a	**0.545±0.052**	0.522±0.041	0.497±0.051	0.509±0.052	0.509±0.042
	svmguide1	**0.904±0.036**	0.849±0.060	0.827±0.069	0.850±0.049	0.868±0.045
Large Scale	mushrooms	0.981±0.007	**0.983±0.010**	0.973±0.013	0.975±0.017	0.983±0.014
	a9a	**0.586±0.011**	0.561±0.020	0.550±0.015	0.564±0.013	0.546±0.022
	cod-rna	**0.950±0.004**	0.891±0.101	0.737±0.190	0.885±0.243	0.709±0.114
	covtype	**0.457±0.003**	0.438±0.026	0.418±0.044	0.439±0.038	0.415±0.028
	poker	**0.084±0.009**	0.066±0.004	0.052±0.007	0.079±0.007	0.079±0.005
	HIGGS	**0.184±0.002**	-	-	-	-



(a) svmguide1. (b) mushrooms. (c) a9a.

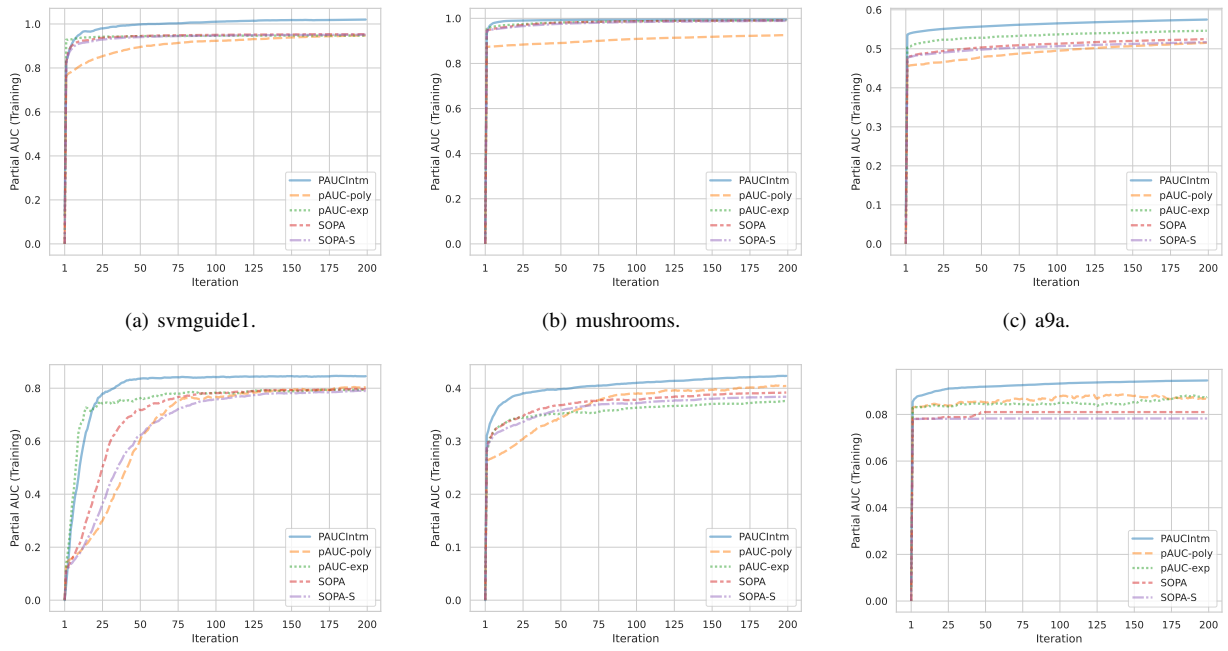
(d) cod-rna (e) covtype. (f) poker.

Fig. 2. Comparison of the convergence rate of all earlier methods on large-scale datasets.

- From Fig. 2 and Fig. 3, we can see that our proposed algorithm PAUCInt and PAUCIntm converge faster than other state-of-the-art methods on large-scale datasets, which is consistent with our theory.
- From Fig. 4 and Fig. 5, we can conclude that our algorithm has higher performance and better stability

TABLE II
pAUC (MEAN±STD), WHERE FPR IN THE RANGE OF 0.05 TO 0.1. **ALL COMPETITORS ARE SELECTED FROM THE DEEP PAUC MAXIMIZATION IN RELATED WORK.** FOR EACH DATASET, THE BEST VALUE IS BOLD, UNDERLINED IS APPLIED THE SECOND.

Scale	Dataset	PAUCIntm(ours)	pAUC-poly	pAUC-exp	SOPA	SOPA-S
Small Scale	breast_cancer	**0.997±0.006**	0.994±0.009	0.991±0.009	0.993±0.010	0.992±0.010
	sonar	**0.588±0.121**	0.508±0.161	0.536±0.156	0.505±0.111	0.546±0.118
	ionosphere	**0.561±0.156**	0.544±0.095	0.522±0.077	0.531±0.101	0.533±0.116
	liver-disorders	0.471±0.141	0.490±0.147	**0.497±0.153**	0.490±0.144	0.485±0.141
	diabetes	**0.584±0.059**	0.548±0.067	0.513±0.061	0.582±0.063	0.578±0.065
	fourclass	0.488±0.062	**0.509±0.043**	0.488±0.060	0.501±0.045	0.507±0.042
	australian	**0.745±0.099**	0.722±0.067	0.733±0.077	0.721±0.093	0.710±0.093
Middle Scale	german	**0.373±0.073**	0.337±0.081	0.352±0.084	0.353±0.085	0.343±0.081
	splice	**0.680±0.066**	0.578±0.079	0.656±0.066	0.627±0.090	0.602±0.096
	svmguide3	**0.541±0.057**	0.500±0.065	0.521±0.077	0.516±0.061	0.527±0.068
	heart	**0.709±0.102**	0.688±0.106	0.708±0.097	0.679±0.082	0.671±0.079
	a1a	**0.556±0.050**	0.522±0.057	0.485±0.066	0.526±0.047	0.515±0.051
	svmguide1	0.909±0.041	0.916±0.016	0.909±0.018	**0.919±0.016**	0.916±0.017
Large Scale	mushrooms	**0.960±0.026**	0.920±0.027	0.934±0.014	0.958±0.006	0.953±0.009
	a9a	**0.583±0.011**	0.549±0.018	0.553±0.015	0.554±0.018	0.548±0.020
	cod-rna	**0.949±0.004**	0.940±0.011	0.936±0.011	0.937±0.012	0.922±0.018
	covtype	**0.435±0.010**	0.382±0.024	0.394±0.016	0.397±0.015	0.386±0.018
	poker	**0.090±0.004**	0.078±0.006	0.0834±0.006	0.0813±0.005	0.0804±-0.005
	HIGGS	**0.100±0.013**	0.092±0.009	0.095±0.011	0.091±0.008	0.098±0.009



(a) svmguide1. (b) mushrooms. (c) a9a.

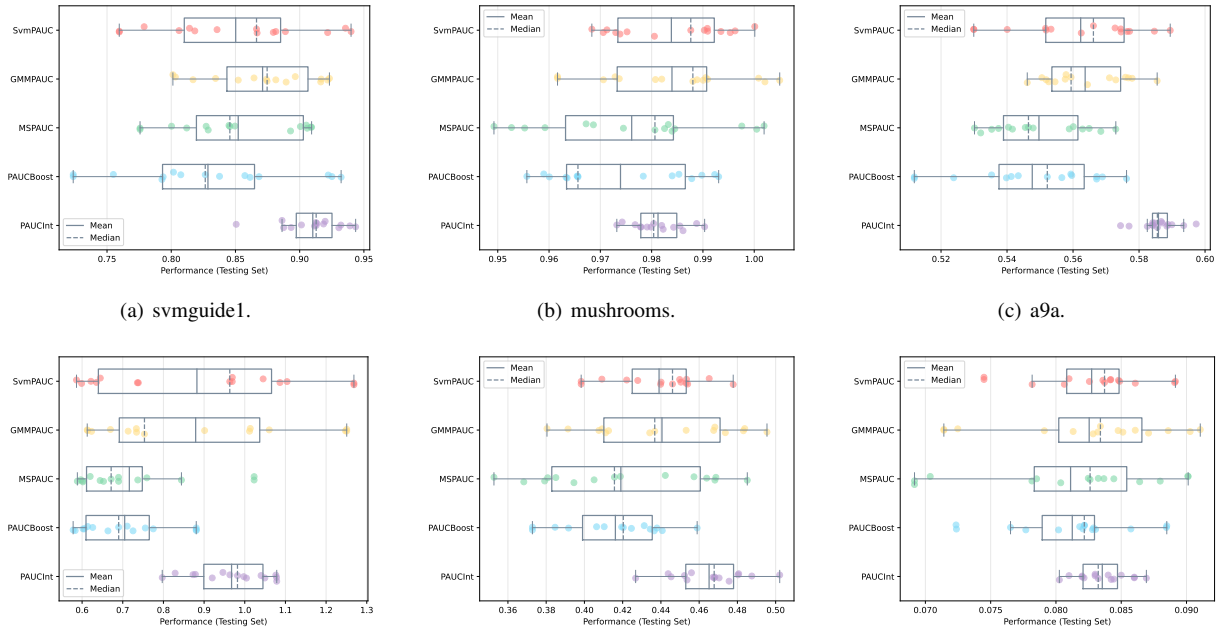
(d) cod-rna (e) covtype. (f) poker.

Fig. 3. Comparison of the convergence speed of all stochastic methods on large-scale datasets (Time unit: seconds).

than other algorithms. That means our proposed method has a lower variance, which can be applied to practice situations.

VII. CONCLUSION

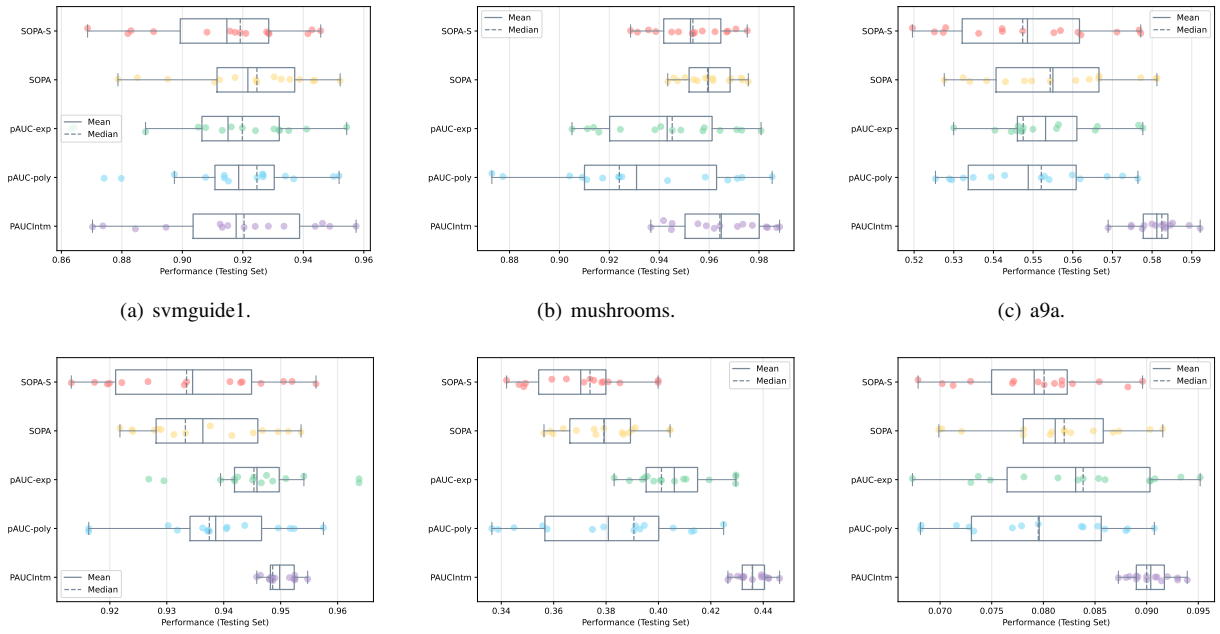
This paper presents a novel formulation to optimize pAUC based on approximate integral in an efficient way. For pAUC optimization, our proposed estimator's main advantage lies in



(a) svmguide1. (b) mushrooms. (c) a9a.

(d) cod-rna. (e) covtype. (f) poker.

Fig. 4. Comparison of the stability of all earlier methods on large-scale datasets.



(a) svmguide1. (b) mushrooms. (c) a9a.

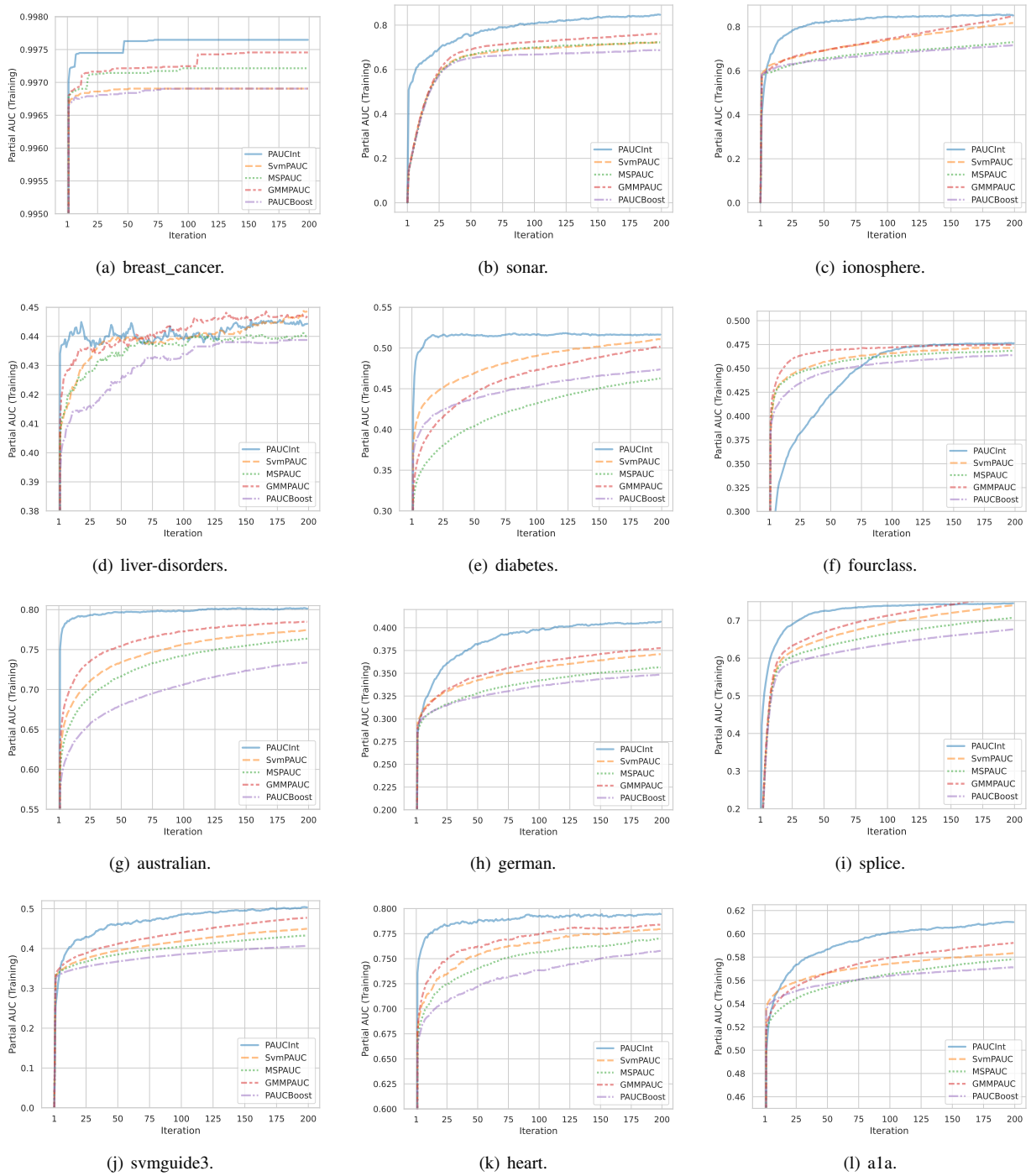
(d) cod-rna. (e) covtype. (f) poker.

Fig. 5. Comparison of the stability of all stochastic methods on large-scale datasets.

eliminating pairwise comparison and ranking issues. Meanwhile, we propose two efficient pAUC optimization algorithms for both full-batch and mini-batch settings. According to stochastic composition optimization theory, our mini-batch algorithm has the convergence rate of $\mathcal{O}(1/\sqrt{k})$. Furthermore, we prove that when the number of samples is large enough, the empirical version of our estimator is an unbiased estimation of the population version under the limited distribution. This property provides theoretical guarantees for our proposed algorithm. Finally, We also verify the efficiency and stability of our algorithm by experiments on extensive benchmark datasets.

APPENDIX A
EXPERIMENT RESULT

We conduct additional experiments and the results of all experiments are shown in Fig .6, Fig .7, Fig .8 and Fig .9.



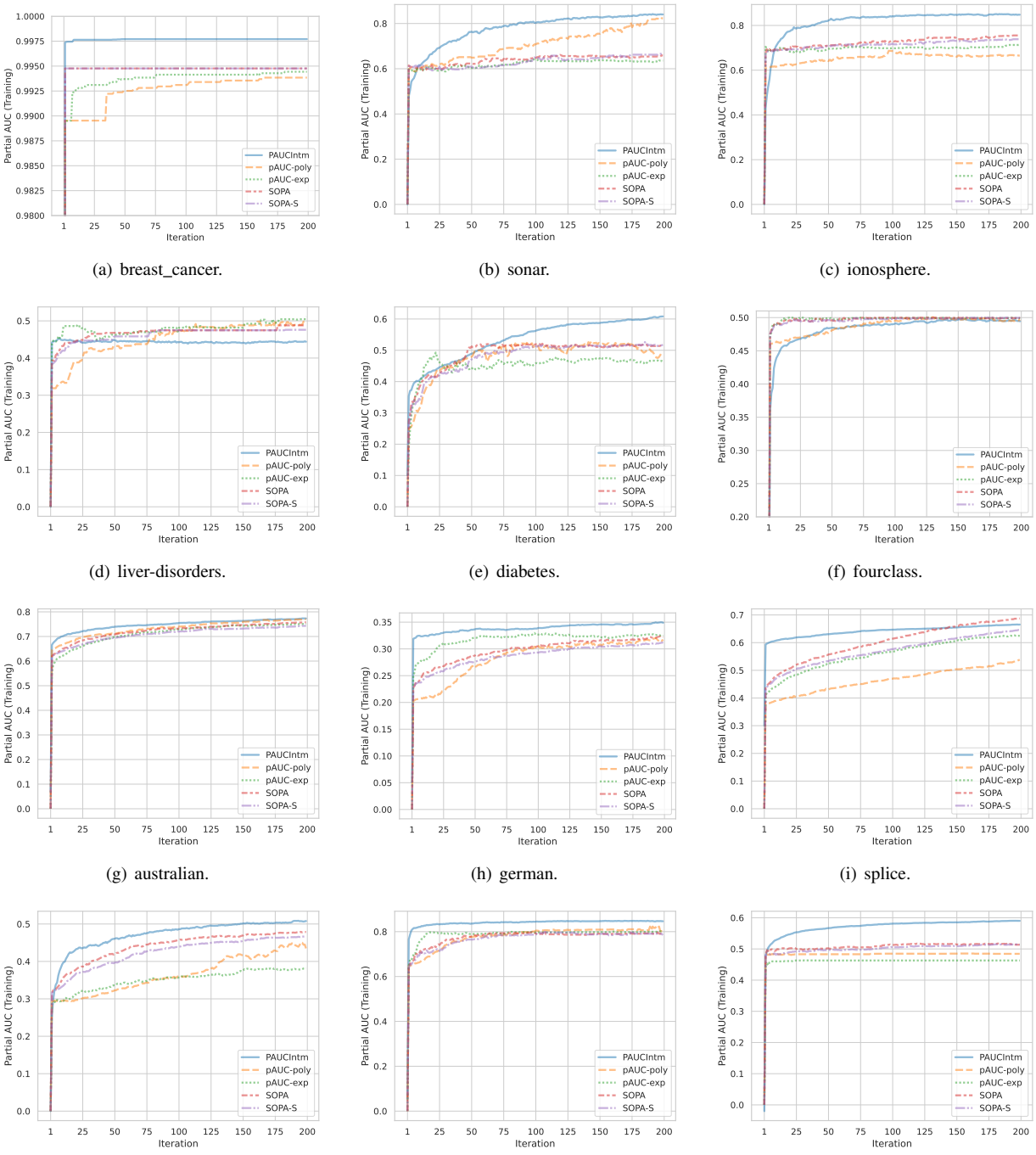
(a) breast_cancer. (b) sonar. (c) ionosphere.

(d) liver-disorders. (e) diabetes. (f) fourclass.

(g) australian. (h) german. (i) splice.

(j) svmguide3. (k) heart. (l) a1a.

Fig. 6. Comparison of the convergence rate of all earlier methods on small-scale and middle-scale datasets.



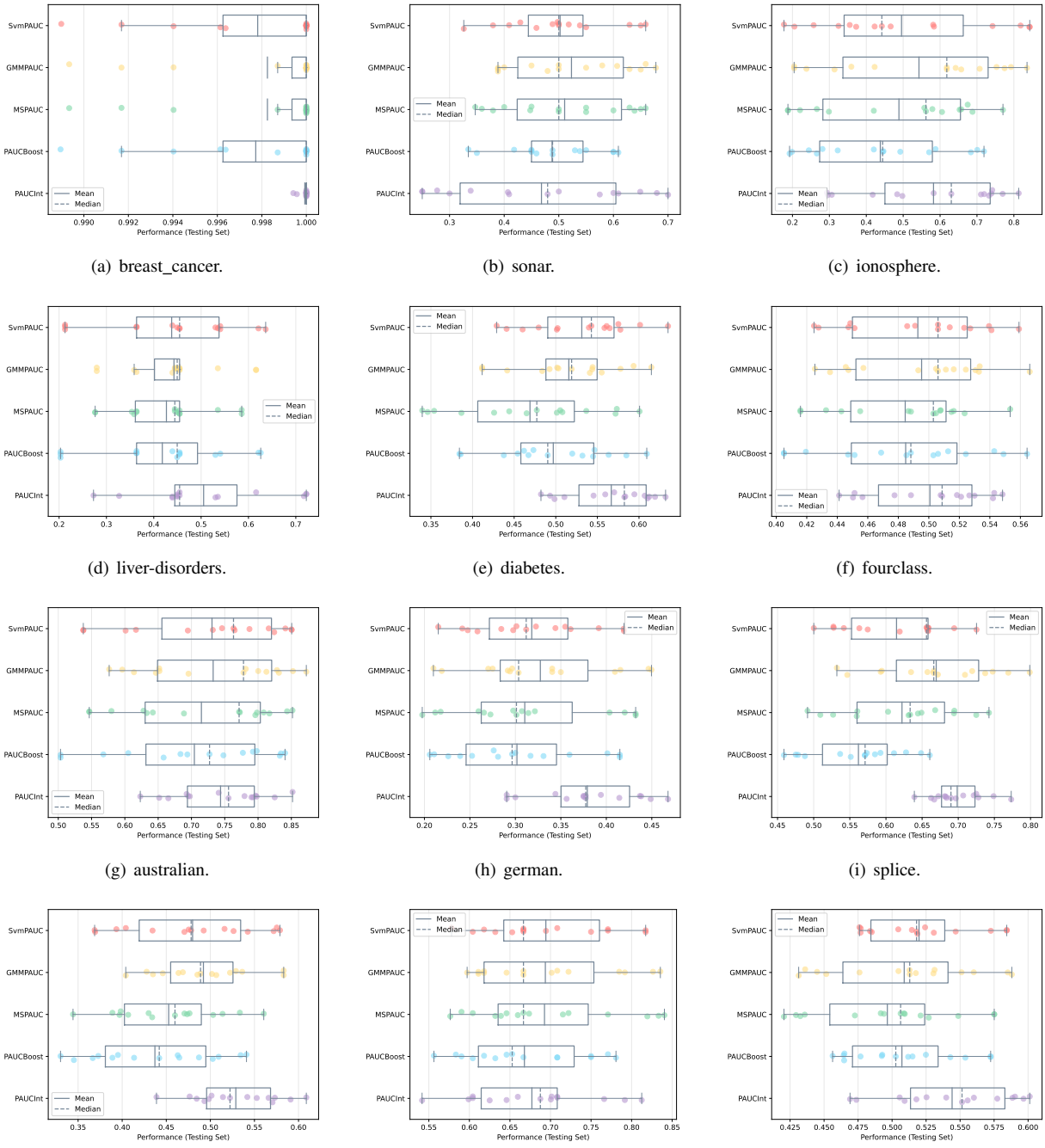
(a) breast_cancer. (b) sonar. (c) ionosphere.

(d) liver-disorders. (e) diabetes. (f) fourclass.

(g) australian. (h) german. (i) splice.

(j) svmguide3. (k) heart. (l) a1a.

Fig. 7. Comparison of the convergence rate of all stochastic methods on small-scale and middle-scale datasets.



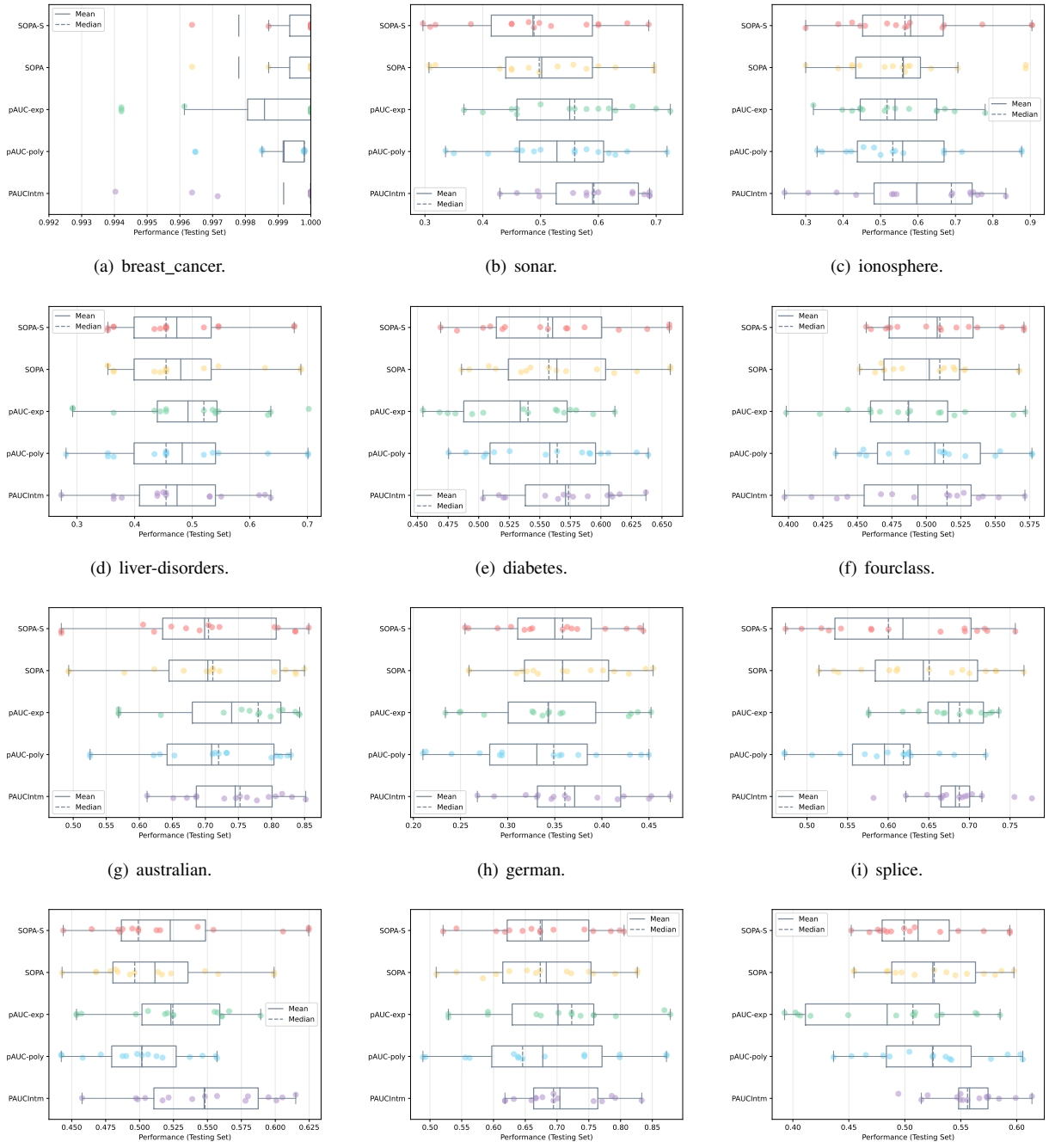
(a) breast_cancer. (b) sonar. (c) ionosphere.

(d) liver-disorders. (e) diabetes. (f) fourclass.

(g) australian. (h) german. (i) splice.

(j) svmguide3. (k) heart. (l) a1a.

Fig. 8. Comparison of the stability of all earlier methods on small-scale and middle-scale datasets.



(a) breast_cancer. (b) sonar. (c) ionosphere.

(d) liver-disorders. (e) diabetes. (f) fourclass.

(g) australian. (h) german. (i) splice.

(j) svmguide3. (k) heart. (l) a1a.

Fig. 9. Comparison of the stability of all stochastic methods on small-scale and middle scale datasets.

APPENDIX B
THE ERROR BOUND OF SURROGATE ERROR FUNCTION

Since the erf is an odd function and is symmetric about the origin. So we only need to calculate the error bound of the right half and multiply it by 2. First, we perform Taylor series on the actual error function and $sin(ax)$.

$$\begin{aligned}\sin(ax) &= \sum_{k=0}^{\infty} \frac{(-1)^k a^{1+2k} x^{1+2k}}{(1+2k)!} + R(n) \\ \operatorname{erf}(x) &= \frac{2}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k x^{1+2k}}{(1+2k)k!} + R(n)\end{aligned} \tag{18}$$

Since the surrogate error function is a piecewise function, the error bound is calculated in two parts. When $-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$, by subtracting and squaring the two formulas, the error of the two functions at point x is

$$\begin{aligned}(\operatorname{erf}(x) - \sin(ax))^2 &= \left(\sum_{k=0}^{\infty} \left(-\frac{2(-1)^k x^{1+2k}}{(1+2k)\sqrt{\pi}k!} + \frac{(-1)^k ax(ax)^{2k}}{(1+2k)!} \right) + R(n) \right)^2 \\ &= \left(\sum_{k=0}^{\infty} (-1)^k x^{1+2k} \left(-\frac{2}{(1+2k)\sqrt{\pi}k!} + \frac{a^{1+2k}}{(1+2k)!} \right) \right)^2\end{aligned} \tag{19}$$

We calculated the Taylor expansion of order $O(d)$, and set x to $0 : 0.01 : \frac{\pi}{2}$, $d = 25$, and use the optimal parameter $a = 1$.

$$\int_0^{\frac{\pi}{2}} (\operatorname{erf}(x) - \sin(x))^2 dx \approx 0.0012183 \tag{20}$$

When $x \geq \frac{\pi}{2}$, and let $erfc(x)$ be $1 - erf(x)$. We use approximation to the complementary error function to calculate this error bound.

$$\begin{aligned}\int_{\frac{\pi}{2}}^{\infty} (\operatorname{erf}(x) - 1)^2 dx &= \frac{2e^{-\pi^2/4} \operatorname{erfc}\left(\frac{\pi}{2}\right)}{\sqrt{\pi}} - \frac{1}{2}\pi \operatorname{erfc}\left(\frac{\pi}{2}\right)^2 - \sqrt{\frac{2}{\pi}} \operatorname{erfc}\left(\frac{\pi}{\sqrt{2}}\right) \\ &\approx 0.0000897761\end{aligned} \tag{21}$$

So we get the error bound between surrogate error function and true error function is $(0.0012183 + 0.0000897761) * 2 = 0.0026161522$.

APPENDIX C
GRADIENT CALCULATION

Gradient of Objective function

When $-\frac{\pi}{2} \leq \frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n} \leq \frac{\pi}{2}$ and $-\frac{\pi}{2} \leq \frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p} \leq \frac{\pi}{2}$:

$$\frac{\partial \mathcal{R}(u;\theta)}{\partial \mu_p} = \frac{\sigma_p \sin\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right) \cos\left(\frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p}\right) - \sigma_n \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right) \sin\left(\frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p}\right)}{\sqrt{2}(\sigma_n^2 - \sigma_p^2)}, \tag{22}$$

$$\frac{\partial \mathcal{R}(u;\theta)}{\partial \mu_n} = \frac{\sigma_p^2 \sin\left(\frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p}\right) \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right) - \sigma_n\sigma_p \cos\left(\frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p}\right) \sin\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right)}{\sqrt{2}(\sigma_n^2 - \sigma_p^2)\sigma_n} - \frac{\sqrt{2}}{2\sigma_n} \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_p}\right), \tag{23}$$

$$\begin{aligned}\frac{\partial \mathcal{R}(u;\theta)}{\partial \sigma_p} &= \frac{\sqrt{2}\sigma_p\sigma_n \cos(L) - (\mu_p - r(u,\theta))\left(\sigma_n \sin(L) + \sigma_p \sin(L)\right)}{2\sqrt{2}(\sigma_n + \sigma_p)^2\sigma_p} \\ &\quad + \frac{\sqrt{2}\sigma_p\sigma_n \cos(L) - (r(u,\theta) - \mu_p)\left(\sigma_n \sin(L) - \sigma_p \sin(L)\right)}{2\sqrt{2}(\sigma_n - \sigma_p)^2\sigma_p},\end{aligned} \tag{24}$$

$$\begin{aligned}\frac{\partial \mathcal{R}(u;\theta)}{\partial \sigma_n} &= \frac{\sqrt{2}\sigma_n^2 \cos(L) + \sigma_p(\mu_n - r(u,\theta))(\sigma_n + \sigma_p)\sin(L)}{2\sqrt{2}(\sigma_n + \sigma_p)^2\sigma_n^2} \\ &\quad + \frac{\sqrt{2}\sigma_n^2 \cos(L) + \sigma_p(\mu_n - r(u,\theta))(\sigma_n - \sigma_p)\sin(L)}{2\sqrt{2}(\sigma_n - \sigma_p)^2\sigma_n^2} - \frac{\sqrt{2}(r(u,\theta) - \mu_n)}{2\sigma_n^2} \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_p}\right),\end{aligned} \tag{25}$$

where

$$L = \frac{\sigma_p(r(u,\theta) - \mu_n) - \sigma_n(r(u,\theta) - \mu_p)}{\sqrt{2}\sigma_p\sigma_n}. \tag{26}$$

When $-\frac{\pi}{2} \leq \frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n} \leq \frac{\pi}{2}$ and $\frac{\pi}{2} < \frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p}$:

$$\frac{\partial \mathcal{R}(r(u,\theta);\theta)}{\partial \mu_p} = 0, \frac{\partial \mathcal{R}(r(u,\theta);\theta)}{\partial \sigma_p} = 0, \tag{27}$$

$$\frac{\partial \mathcal{R}(r(u,\theta);\theta)}{\partial \mu_n} = -\frac{1}{\sqrt{2}\sigma_n} \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right), \tag{28}$$

$$\frac{\partial \mathcal{R}(u;\theta)}{\partial \sigma_n} = -\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n^2} \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right). \tag{29}$$

When $-\frac{\pi}{2} \leq \frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n} \leq \frac{\pi}{2}$ and $\frac{r(u,\theta)-\mu_p}{\sqrt{2}\sigma_p} < -\frac{\pi}{2}$:

$$\frac{\partial \mathcal{R}(u;\theta)}{\partial \mu_p} = 0, \frac{\partial \mathcal{R}(u;\theta)}{\partial \sigma_p} = 0, \tag{30}$$

$$\frac{\partial \mathcal{R}(u;\theta)}{\partial \mu_n} = \frac{1}{\sqrt{2}\sigma_n} \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right), \tag{31}$$

$$\frac{\partial \mathcal{R}(u;\theta)}{\partial \sigma_n} = \frac{u-\mu_n}{\sqrt{2}\sigma_n^2} \cos\left(\frac{r(u,\theta)-\mu_n}{\sqrt{2}\sigma_n}\right). \tag{32}$$

Gradient of $\phi(\theta_S)$

where the calculation of ϕ'_{θ_S} as:

$$\frac{\partial \phi(\theta_S)}{\partial \bar{S_+}} = \frac{\partial \mathcal{R}(r(\beta,\theta_S);\theta_S)}{\partial \bar{S_+}} - \frac{\partial \mathcal{R}(r(\alpha,\theta_S);\theta_S)}{\partial \bar{S_+}}, \tag{33}$$

$$\frac{\partial \phi(\theta_S)}{\partial \bar{V_+}} = \frac{\partial \mathcal{R}(r(\beta,\theta_S);\theta_S)}{\partial \sigma_p} \frac{\partial \sqrt{\bar{V_+} - (\bar{S_+})^2}}{\partial \bar{V_+}} - \frac{\partial \mathcal{R}(r(\alpha,\theta_S);\theta_S)}{\partial \sigma_p} \frac{\partial \sqrt{\bar{V_+} - (\bar{S_+})^2}}{\partial \bar{V_+}}, \tag{34}$$

$$\frac{\partial \phi(\theta_S)}{\partial \bar{S_-}} = \frac{\partial \mathcal{R}(r(\beta,\theta_S);\theta_S)}{\partial \bar{S_-}} - \frac{\partial \mathcal{R}(r(\alpha,\theta_S);\theta_S)}{\partial \bar{S_-}}, \tag{35}$$

$$\frac{\partial \phi(\theta_S)}{\partial \bar{V_-}} = \frac{\partial \mathcal{R}(r(\beta,\theta_S);\theta_S)}{\partial \sigma_p} \frac{\partial \sqrt{\bar{V_-} - (\bar{S_-})^2}}{\partial \bar{V_-}} - \frac{\partial \mathcal{R}(r(\alpha,\theta_S);\theta_S)}{\partial \sigma_p} \frac{\partial \sqrt{\bar{V_-} - (\bar{S_-})^2}}{\partial \bar{V_-}}. \tag{36}$$

Gradient of Expectation function

In mini-batch optimization algorithm, we define a function g, which use to calculate the distribution parameters and has details like this

$$g_{\Omega_k}(\boldsymbol{w_k}) = (\hat{\mu_p}, \hat{\mu_n}, \hat{\sigma_p}, \hat{\sigma_n}) \tag{37}$$

We denote $N^+_{\Omega_k}/N^-_{\Omega_k}$ be number of positive/negative instances in Ω_k.

$$g_{\Omega_k}(\boldsymbol{w}_k) = \left(\frac{1}{N^+_{\Omega_k}} \sum_{i=1}^{N^+_{\Omega_k}} \boldsymbol{w}_k^T \boldsymbol{x}_i^+, \frac{1}{N^-_{\Omega_k}} \sum_{i=1}^{N^-_{\Omega_k}} \boldsymbol{w}_k^T \boldsymbol{x}_i^-, \sqrt{\frac{1}{N^+_{\Omega_k}} \sum_{i}^{N^+_{\Omega_k}} (\boldsymbol{w}_k^T \boldsymbol{x}_i^+ - \hat{\mu_p})^2}, \sqrt{\frac{1}{N^-_{\Omega_k}} \sum_{i}^{N^-_{\Omega_k}} (\boldsymbol{w}_k^T \boldsymbol{x}_i^- - \hat{\mu_n})^2} \right) \tag{38}$$

$\nabla g\left(\boldsymbol{w}_k\right)$ is an essential part for SCGD optimization. So we give the calculation of it.

$$\nabla g_{\Omega_k}\left(\boldsymbol{w}_k\right) = \left(\frac{1}{N^+_{\Omega_k}} \sum_{i=1}^{N^+_{\Omega_k}} \boldsymbol{x}_i^+, \frac{1}{N^-_{\Omega_k}} \sum_{i=1}^{N^-_{\Omega_k}} \boldsymbol{x}_i^-, \frac{\frac{1}{N^+_{\Omega_k}} \sum_{i=1}^{N^+_{\Omega_k}} \boldsymbol{w}_k (\boldsymbol{x}_i^+)^2 - \boldsymbol{x}_i^+ \mu_p}{\sqrt{\frac{1}{N^+_{\Omega_k}} \sum_{i}^{N^+_{\Omega_k}} (\boldsymbol{w}_k^T \boldsymbol{x}_i^+ - \mu_p)^2}}, \frac{\frac{1}{N^-_{\Omega_k}} \sum_{i=1}^{N^-_{\Omega_k}} \boldsymbol{w}_k (\boldsymbol{x}_i^-)^2 - \boldsymbol{x}_i^- \mu_n}{\sqrt{\frac{1}{N^-_{\Omega_k}} \sum_{i}^{N^-_{\Omega_k}} (\boldsymbol{w}_k^T \boldsymbol{x}_i^- - \mu_n)^2}} \right) \tag{39}$$

And we the gradient of function $\mathcal{R}_{\alpha,\beta}$, denote it as $\nabla \mathcal{R}_{\alpha,\beta}(g_{\Omega_k}(\boldsymbol{w}_k))$

$$\nabla \mathcal{R}_{\alpha,\beta}(g_{\Omega_k}(\boldsymbol{w}_k)) = \left(\frac{\partial \mathcal{R}_{\alpha,\beta}(g_{\Omega_k}(\boldsymbol{w}_k))}{\partial \mu_p}, \frac{\partial \mathcal{R}_{\alpha,\beta}(g_{\Omega_k}(\boldsymbol{w}_k))}{\partial \mu_n}, \frac{\partial \mathcal{R}_{\alpha,\beta}(g_{\Omega_k}(\boldsymbol{w}_k))}{\partial \sigma_p}, \frac{\partial \mathcal{R}_{\alpha,\beta}(g_{\Omega_k}(\boldsymbol{w}_k))}{\partial \sigma_n} \right). \tag{40}$$

APPENDIX D
PROOF OF ASYMPTOTIC NORMALITY

Give positive input space, The multivariate normal distribution of a d-dimensional random vector $\boldsymbol{X}^+ = (X_1^+, \cdots, X_d^+)^T$ can be written in the following notation:

$$\boldsymbol{X}^+ \sim \mathcal{N}_d(\boldsymbol{\mu}^+, \boldsymbol{\Sigma}^+) \tag{41}$$

We can calculate mean vector of random variable $\boldsymbol{X}$ with d-dimensional follow as:

$$\boldsymbol{\mu}^+ = \mathbb{E}[\boldsymbol{X}^+] = (\mathbb{E}[\boldsymbol{X}_1^+], \mathbb{E}[\boldsymbol{X}_2^+], \cdots, \mathbb{E}[\boldsymbol{X}_d^+])^T \tag{42}$$

Meantime, give the definition of covariance matrix of random variable $\boldsymbol{X}$

$$\boldsymbol{\Sigma}_{i,j}^+ = \mathbb{E}[(\boldsymbol{X}_i^+ - \boldsymbol{\mu}_i^+)(\boldsymbol{X}_j^+ - \boldsymbol{\mu}_j^+)] = Cov[\boldsymbol{X}_i^+, \boldsymbol{X}_j^+] \tag{43}$$

And then assumed that the classifier's form is $f = \boldsymbol{w}^T\boldsymbol{x}$ which $\hat{\boldsymbol{w}} \in \mathbb{R}^d$. According to the addition theory of normal distribution, we have

$$\hat{\boldsymbol{w}}^T\boldsymbol{X}^+ \sim \mathcal{N}(\hat{\boldsymbol{w}}^T\boldsymbol{\mu}^+, \hat{\boldsymbol{w}}^T\boldsymbol{\Sigma}^+\hat{\boldsymbol{w}}) \tag{44}$$

The same with negative input space $\boldsymbol{X}^- = (X_1^-, \cdots, X_d^-)^T$.

$$\hat{\boldsymbol{w}}^T\boldsymbol{X}^- \sim \mathcal{N}(\hat{\boldsymbol{w}}^T\boldsymbol{\mu}^-, \hat{\boldsymbol{w}}^T\boldsymbol{\Sigma}^-\hat{\boldsymbol{w}}) \tag{45}$$

Import new variable S, V, and denote N_+, N_- be positive and negative instances number. We have the definition of them as

$$\bar{S_+} = \frac{1}{N_+}\sum_{i=1}^{N_+}\boldsymbol{w}^T\boldsymbol{x}_i^+, \bar{S_-} = \frac{1}{N_-}\sum_{j=1}^{N_-}\boldsymbol{w}^T\boldsymbol{x}_j^- \tag{46}$$

$$\bar{V_+} = \frac{1}{N_+}\sum_{i=1}^{N_+}(\boldsymbol{w}^T\boldsymbol{x}_i^+)^2, \bar{V_-} = \frac{1}{N_-}\sum_{j=1}^{N_-}(\boldsymbol{w}^T\boldsymbol{x}_j^-)^2 \tag{47}$$

Assumed that $\mathbb{E}[\bar{S_+}] = \alpha_1^+, \mathbb{E}[\bar{V_+}] = \alpha_2^+, \mathbb{E}[\bar{S_-}] = \alpha_1^-, \mathbb{E}[\bar{V_-}] = \alpha_2^-$. Let $\mathbb{E}[(\bar{V_+})^2] = \alpha_4^+$, $\mathbb{E}[S_+V_+] = \alpha_3^+$, we can give another way to express variance $V = n^{-1}\sum_{i=1}^n(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^2 = \bar{\boldsymbol{X}}^2 - (\bar{\boldsymbol{X}})^2$ and Covariance $COV(S, V) = \mathbb{E}[SV] - \mu_S\mu_V$.

$$COV(\bar{S_+}, \bar{S_+}) = \mathbb{V}[\bar{S_+}] = \alpha_2^+ - (\alpha_1^+)^2 \tag{48}$$

$$COV(\bar{V_+}, \bar{V_+}) = \mathbb{V}[\bar{V_+}] = \alpha_4^+ - (\alpha_2^+)^2 \tag{49}$$

$$COV(\bar{S_+}, \bar{V_+}) = COV(\bar{V_+}, \bar{S_+}) = \alpha_3^+ - \alpha_1^+\alpha_2^+ \tag{50}$$

Under the assumption of input sample is infinite. The central limit theorem states that

$$\sqrt{N_+}\left(\begin{pmatrix}\bar{S_+}\\ \bar{V_+}\end{pmatrix} - \begin{pmatrix}\alpha_1^+\\ \alpha_2^+\end{pmatrix}\right) \xrightarrow{D} \mathcal{N}_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \tilde{\boldsymbol{\Sigma}}\right) \tag{51}$$

where $\tilde{\Sigma}$ as

$$\tilde{\boldsymbol{\Sigma}} = \begin{pmatrix}\alpha_2^+ - (\alpha_1^+)^2 & \alpha_3^+ - \alpha_1^+\alpha_2^+\\ \alpha_3^+ - \alpha_1^+\alpha_2^+ & \alpha_4^+ - (\alpha_2^+)^2\end{pmatrix} \tag{52}$$

Similarly, we can give negative cases. Since the positive and negative instances are independent of each other, combine them and both sides are multiplied by an $\sqrt{N_+N_-}$ at the same time, and transform covariance matrix to get

$$\left(\begin{pmatrix}\bar{S_+}\\ \bar{V_+}\\ \bar{S_-}\\ \bar{V_-}\end{pmatrix} - \begin{pmatrix}\alpha_1^+\\ \alpha_2^+\\ \alpha_1^-\\ \alpha_2^-\end{pmatrix}\right) \xrightarrow{D} \mathcal{N}_4\left(\begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \tilde{\boldsymbol{\Sigma}}\right) \tag{53}$$

which redefined $\tilde{\boldsymbol{\Sigma}}$ as

$$\tilde{\boldsymbol{\Sigma}} = \begin{pmatrix}\frac{\alpha_2^+ - (\alpha_1^+)^2}{N_+} & \frac{\alpha_3^+ - \alpha_1^+\alpha_2^+}{N_+} & 0 & 0\\ \frac{\alpha_3^+ - \alpha_1^+\alpha_2^+}{N_+} & \frac{\alpha_4^+ - (\alpha_2^+)^2}{N_+} & 0 & 0\\ 0 & 0 & \frac{\alpha_2^- - (\alpha_1^-)^2}{N_-} & \frac{\alpha_3^- - \alpha_1^-\alpha_2^-}{N_-}\\ 0 & 0 & \frac{\alpha_3^- - \alpha_1^-\alpha_2^-}{N_-} & \frac{\alpha_4^- - (\alpha_2^-)^2}{N_-}\end{pmatrix} \tag{54}$$

Let $\boldsymbol{\theta}_S = \{\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-\}$ and $\boldsymbol{\theta}_S^* = \mathbb{E}[\boldsymbol{\theta}_S]$, $\tilde{\boldsymbol{\Sigma}}$ be the covariance matrix of $\boldsymbol{\theta}_S$. We construct a function $\phi : \mathbb{R}^4 \to \mathbb{R}$ such that has the same output as $\mathcal{R}(u;\boldsymbol{\theta})|_{r(\beta,\boldsymbol{\theta})}^{r(\alpha,\boldsymbol{\theta})}$ with $\boldsymbol{\theta}_S$ as input when the given data is the same. So we have estimated pAUC like: $-\hat{\text{pAUC}} = \phi(\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-)$. When the input is $\boldsymbol{\theta}_S^*$, we can obtain the optimal approximated pAUC: $-\text{pAUC}^* = \phi(\boldsymbol{\theta}_S^*)$. If there exists a linear map $\nabla\phi_{\boldsymbol{\theta}_S^*} : \mathbb{R}^4 \to \mathbb{R}$ such that

$$\nabla\phi_{\boldsymbol{\theta}_S^*} = \left[\frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\bar{S}_+}, \frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\sigma_p}\frac{\partial\sigma_p}{\partial\bar{V}_+}, \frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\bar{S}_-}, \frac{\partial\phi(\boldsymbol{\theta}_S^*)}{\partial\sigma_n}\frac{\partial\sigma_n}{\partial\bar{V}_-}\right],$$

according to the delta method, we have:

$$[\phi(\bar{S}_+, \bar{V}_+, \bar{S}_-, \bar{V}_-) - \phi(\boldsymbol{\theta}_S^*)] \xrightarrow{D} \mathcal{N}\left(0, \nabla\phi_{\boldsymbol{\theta}_S^*}^T \tilde{\boldsymbol{\Sigma}} \nabla\phi_{\boldsymbol{\theta}_S^*}\right).$$

APPENDIX E
DATASET DETAILS

TABLE III
STATISTICS OF ALL DATASETS FOR EXPERIMENT.

Dataset	#Training	#Testing	#feature	#positive/#negative
breast_cancer	546	137	10	0.538
sonar	166	42	60	0.873
ionosphere	281	70	34	1.785
liver-disorders	116	29	5	0.611
diabetes	614	154	8	1.869
fourclass	689	174	2	0.551
australian	552	138	14	0.803
german	800	200	24	0.428
splice	800	200	60	1.072
svmguide3	994	249	22	0.313
heart	216	54	13	0.800
a1a	1,284	321	119	0.326
svmguide1	2,471	618	4	0.544
mushrooms	6,499	1,625	112	0.930
a9a	26,048	6,513	123	0.317
cod-rna	47,628	11,907	8	0.500
covtype	464,809	116,203	54	1.050
poker	820,008	205,002	10	1.004
HIGGS	7,333,333	1,833,334	28	1.125