# Quadruply Stochastic Gradient Method for Large Scale Nonlinear Semi-Supervised Ordinal Regression AUC Optimization

**Wanli Shi,**[1] **Bin Gu,**[1,2*] **Xiang Li,**[3] **Heng Huang**[4,2]
[1]School of Computer & Software, Nanjing University of Information Science & Technology, P.R.China
[2]JD Finance America Corporation
[3]Computer Science Department, University of Western Ontario, Canada
[4]Department of Electrical & Computer Engineering, University of Pittsburgh, USA
wanlishi@nuist.edu.cn, jsgubin@gmail.com, lxiang2@uwo.ca, heng.huang@pitt.edu

## Abstract

Semi-supervised ordinal regression ($S^2$OR) problems are ubiquitous in real-world applications, where only a few ordered instances are labeled and massive instances remain unlabeled. Recent researches have shown that directly optimizing concordance index or AUC can impose a better ranking on the data than optimizing the traditional error rate in ordinal regression (OR) problems. In this paper, we propose an unbiased objective function for $S^2$OR AUC optimization based on ordinal binary decomposition approach. Besides, to handle the large-scale kernelized learning problems, we propose a scalable algorithm called Q$S^3$ORAO using the doubly stochastic gradients (DSG) framework for functional optimization. Theoretically, we prove that our method can converge to the optimal solution at the rate of $O(1/t)$, where $t$ is the number of iterations for stochastic data sampling. Extensive experimental results on various benchmark and real-world datasets also demonstrate that our method is efficient and effective while retaining similar generalization performance.

## Introduction

Supervised ordinal regression (OR) problems have made great process in the past few decades, such as (Chu and Keerthi 2007; Fathony, Bashiri, and Ziebart 2017; Niu et al. 2016; Gu et al. 2015). However, in various practical fields, such as facial beauty assessment (Yan 2014), credit rating (Kim and Ahn 2012), social sciences (Fullerton and Xu 2012) or more, collecting a large amount of ordinal labeled instances is time-consuming, while unlabeled data are available in abundance. Often, the finite ordinal data are insufficient to learn a good ordinal regression model. To improve the performance of the classifiers, one needs to incorporate unlabeled instances into the training process. So far, semi-supervised ordinal regression ($S^2$OR) problems have attracted great attention in machine learning communities, such as (Srijith, Shevade, and Sundararajan 2013; Seah, Tsang, and Ong 2012).

To evaluate the performance of an OR model, many metrics could be used, *e.g.*, the mean absolute error, the mean squared error. However, Waegeman, De Baets and Boullart, (2008) have shown that OR models which minimize these errors do not necessarily impose a good ranking on the data. To handle this problem, many researchers start to use AUC or concordance index in solving OR problems since AUC is defined on an ordinal scale, such as (Waegeman, De Baets, and Boullart 2008; Waegeman and De Baets 2010; Fürnkranz, Hüllermeier, and Vanderlooy 2009; Uematsu and Lee 2014). We summarized several representative OR algorithms in Table 1.

However, existing AUC optimization methods focus on supervised OR problems, and none of them can be applied to semi-supervised learning problems. The main challenge is how to incorporate unlabeled instances into the AUC optimization process. For the semi-supervised learning research field in general, many existing methods, such as (Seah, Tsang, and Ong 2012; Fujino and Ueda 2016), have leveraged the cluster assumptions, which states that similar instances tend to share the same label, to solve this problem. However, the clustering assumption is rather restrictive and may mislead a model towards a biased solution. Nevertheless, recent works (Sakai, Niu, and Sugiyama 2018; Xie and Li 2018) have shown that the clustering assumption is actually unnecessary at least for binary classification problems. In the same vein, we propose an objective function of $S^2$OR AUC optimization based on ordinal binary decomposition without using the clustering assumption. Specifically, for a $k$ classes OR problem, we use $k-1$ hyperlanes to decompose the orginal problem into $k-1$ binary semi-supervised AUC optimization problems, where the AUC risk can be viewed as a linear combination of AUC risk between labeled instances and AUC risk between labeled and unlabeled instances. Then, the overall AUC risk in $S^2$OR is equivalent to the mean of AUC for $k-1$ subproblems.

Nonlinear data structures widely exist in many real-world problems, and kernel method is a typical way to solve such problems. However, kernel-based methods are hardly scalable. Specifically, the kernel matrix needs $O(n^2d)$ operations to be calculated and $O(n^2)$ to be stored, where $n$ denotes the number of training data and $d$ denotes the dimensionality of the data. Besides, the bottlenecks of the computational complexities become more severe in solving pair-

*Corresponding Author

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Table 1: Several representative OR algorithms. ($D$ denotes the number of random features, $k$ denotes the number of classes, $n$ denotes the number of training samples, and $t$ denotes number of iterations.)

| Learning setting | Algorithm | Reference | AUC | Computational complexity | Space complexity |
|---|---|---|---|---|---|
| Supervised | ALOR | Fathony et al, (2017) | No | $O(n^3)$ | $O(n^2)$ |
| | SVOREX | Chu et al, (2007) | No | $O(n^3)$ | $O(n^2)$ |
| | VUS | Waegeman et al, (2008) | Yes | $O(n^3)$ | $O(n^2)$ |
| | MultiRank | Uematsu and Lee, (2014) | Yes | $O(n^3)$ | $O(n^2)$ |
| Semi-supervised | TOR | Seah et al, (2012) | No | $O(n^3)$ | $O(n^2)$ |
| | SSORERM | (Tsuchiya et al. 2019) | No | $O(n^3)$ | $O(n^2)$ |
| | SSGPOR | Srijith et al, (2013) | No | $> O(n^3)$ | $O(n^2)$ |
| | ManifoldOR | Liu et al, (2011) | No | $O(n^3)$ | $O(n^2)$ |
| | QS$^3$ORAO | Ours | Yes | $O(Dt^2)$ | $O(Dn)$ |

wise learning problems such as AUC optimization. In addition, as required by AUC computation, the OR learning problem needs to be decomposed into several binary classification sub-problems, which further increases the problem size and computational complexity. Thus, the new challenge is how to scale up kernel-based S$^2$OR AUC optimization.

Recently, Dai et al, (2014) proposed doubly stochastic gradient (DSG) method to scale up kernel-based algorithms. Specifically, in each iteration, DSG randomly samples a data instance and its random features to compute the doubly stochastic functional gradient, and then the model function can be updated by using this gradient. However, the original DSG cannot be applied to solve the kernel-based S$^2$OR AUC optimization. On the one hand, optimizing AUC is a pairwise problem which is much more complicated than the pointwise problem considered in standard DSG framework. On the other hand, S$^2$OR optimization problems need to handle two different types of data, *i.e.,* unlabeled dataset and the datasets of class $i$, while standard DSG focuses on minimizing the empirical risk on a single dataset with all data instances labeled.

To address these challenging problems, we introduce multiple sources of randomness. Specifically, we randomly sample a positive instance, a negative instance, an unlabeled instance, and their random features in each subproblem to calculate the approximated stochastic gradients of our objective function in each iteration. Then the ranking function can be iteratively updated. Since we randomly sample instances from four data sources in each subproblem, we denote our method as quadruply stochastic gradient S$^2$OR AUC optimization method (QS$^3$ORAO). Theoretically, we prove that our proposed QS$^3$ORAO can converge to the optimal solution at the rate of $O(1/t)$. Extensive experimental results on benchmark datasets and real-world datasets also demonstrate that our method is efficient and effective while retaining similar generalization performance.

**Contributions.** The main contributions of this paper are summarized as follows.

1. We propose an objective function for solving S$^2$OR AUC optimization problems in an unbiased manner. To the best of our knowledge, this is the first objective formulation incorporating the unlabeled data into the AUC optimization process in OR problems.
2. To optimize the objective function under the kernel learning setting, we propose an efficient and scalable S$^2$OR AUC optimization algorithm, QS$^3$ORAO, based on DSG framework.
3. We provide the convergence analysis of QS$^3$ORAO, which indicates that an ideal $O(1/t)$ convergence rate is possible under certain mild assumptions.

## Related Works

### Semi-Supervised Ordinal Regression

In real-world applications, labeled instances are often costly to calibrate or difficult to obtain. This has led to a lot of efforts to study how to make full use of unlabeled data to improve the accuracy of classification, such as (Zhang et al. 2019; Han et al. 2018). Many existing methods incorporate unlabeled instances into learning propose by using various restrictive assumptions. For example, Seah, Tsang and Ong, (2012) proposed TOR based on cluster assumption, where the instances share the same label if there are close to each other. Liu et al, (2011) proposed a semi-supervised OR method, ManifoldOR, based on the assumption that the input data are distributed into a lower-dimensional manifold (Belkin, Niyogi, and Sindhwani 2006). Besides, Srijith et al, (2013) proposed SSGPOR based on the low density separation assumption (Chapelle, Scholkopf, and Zien 2009). We summarized these semi-supervised OR algorithms in Table 1. Note, in our semi-supervised OR AUC method, we do not need these restrictive assumptions.

### Kernel Approximation

Kernel approximation is a common method to scale up kernel-based algorithms, which can be decomposed into two categories. One is data-dependent methods, such as greedy basis selection techniques (Smola and Schölkopf 2000), incomplete Cholesky decomposition (Fine and Scheinberg 2001), Nyström method (Drineas and Mahoney 2005). In order to achieve a low generalization performance, these methods usually need a large amount of training instances to compute a low-rank approximation of the kernel matrix, which may have high memory requriement. Another one is data-independent methods, which directly approximates the kernel function unbiasedly with some basis functions, such as random Fourier feature (RFF) (Rahimi and Recht 2008). However, RFF method needs to save large amounts of random features. Instead of saving all the random features, Dai et al, (2014) proposed DSG algorithm to use *pseudo-random number generators* to generate the random features on-the-fly, which has been widely used (Shi et al. 2019; Geng et al. 2019; Li et al. 2017). Our method can be viewed as an extension of (Shi et al. 2019). However, OR is much more complicated than binary classification, since OR involves $k$ classes with ordering constraint, while (Shi et al. 2019) only studies binary classification. How the $k$ ordered classes could be learnt under the DSG framework is a novel and challenging problem. Theoretically, whether and to what extent the convergence property remains true is also a non-trivial problem.

## Preliminaries

In this section, we first give a brief review of the AUC optimization framework in supervised ordinal regression settings, and then we propose our objective function in S$^2$OR AUC optimization problems. Finally, we give a brief review of random Fourier features.

### Supervised Ordinal Regression AUC Optimization

Let $x \in \mathbb{R}^d$ be a $d$-dimensional data instance and $y = \{1, \cdots, k\}$ be the label of each instance. Let $p(x, y)$ be the underlying joint distribution density of $\{x, y\}$. In supervised OR problems, the labeled datasets of each class can be viewed as drawn from the conditional distributional density $p(x|y)$ as follows,

$$\mathcal{D}_j = \{x_i^j\}_{i=1}^{n_j} \sim p(x|y=j),\ j = 1, \cdots, k.$$

Generally speaking, the vast majority of existing ordinal regression models can be represented as ,

$$h(x) = \begin{cases} 1, & \text{if} \quad f(x) < b_1 \\ j, & \text{if} \quad b_{j-1} < f(x) < b_j, j = 2, \cdots, k-1 \\ k, & \text{if} \quad f(x) > b_{k-1} \end{cases},$$

where $b_1 < \cdots < b_{k-1}$ denote the thresholds and $f : \mathbb{R}^d \mapsto \mathbb{R}$ is commonly referred as a ranking function (Waegeman, De Baets, and Boullart 2008). The model $h$ means that we need to consider $k-1$ parallel hyperplanes, $f(x) - b_j$, which decompose the ordinal target variables into $k-1$ binary classification subproblems. Therefore, the problem of calculating the AUC in OR problems can be transformed to that of calculating AUC in $k-1$ binary subproblems.

In binary classification, AUC means the probability that a randomly sampled positive instance receive a higher ranking than a randomly drawn negative instance. Thus, to calculate AUC in $j$-th subproblem, we need to define which part is positive. Fortunately, in OR problems, instances can naturally be ranked by their ordinal labels. Therefore, for the $j$-th binary classification hyperplane, the first consecutive $j$ categories, $1, \cdots, j$, can be regarded as negative, and the rest of the classes, $j+1, \cdots, k$, can be regarded as positive. Then we obtain two new datasets as follows,

$$\mathcal{D}_n^j = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_j \sim p_-^j = \frac{\sum_{i=1}^{j} \theta_i p(x|y=i)}{\sum_{i=1}^{j} \theta_i},$$

$$\mathcal{D}_p^j = \mathcal{D}_{j+1} \cup \cdots \cup \mathcal{D}_k \sim p_+^j = \frac{\sum_{i=j+1}^{k} \theta_i p(x|y=i)}{\sum_{i=j+1}^{k} \theta_i},$$

where $\theta_i$ denotes class prior of each class. Then AUC in each binary subproblem can be calculated by

$$\mathrm{AUC} = 1 - \mathbb{E}_{x_p^j \sim p_+^j}\left[\mathbb{E}_{x_n^j \sim p_-^j}\left[l_{01}\left(f(x_p^j), f(x_n^j)\right)\right]\right],$$

where $l_{01}(u, v) = \frac{1}{2}(1 - \mathrm{sign}(u - v))$ and $\mathbb{E}_{x \sim p(\cdot)}$ denotes the expectation over distribution $p(\cdot)$. The zero-one loss can be replaced by squared pairwise loss function $l_s(u, v) = (1 - u + v)^2$ (Gao and Zhou 2015; Gao et al. 2013). While in real-world problems, the distribution is unknown and one usually uses the empirical mean to replace the expectation. Thus, the second term can be rewritten as

$$R_{\mathrm{PN}}^j = \mathbb{E}_{x_p^j \in \mathcal{D}_p^j}\left[\mathbb{E}_{x_n^j \in \mathcal{D}_n^j}\left[l_{01}\left(f(x_p^j), f(x_n^j)\right)\right]\right], \tag{1}$$

where $\mathbb{E}_{x \in \mathcal{D}}$ denotes the empirical mean on the dataset $\mathcal{D}$. Equation (1) can be viewed as AUC risk between positive and negative instances. Obviously, maximizing AUC is equivalent to minimizing AUC risk $R_{\mathrm{PN}}$.

According to (Waegeman, De Baets, and Boullart 2008), the goal of AUC optimization in OR problems is to train a ranking function $f$ which can minimize the overall AUC risk of $k-1$ subproblems,

$$R_\alpha = \frac{1}{k-1}\sum_{j=1}^{k-1} R_{\mathrm{PN}}^j. \tag{2}$$

### Semi-Supervised Ordinal Regression AUC Optimization

In semi-supervised OR problems, the unlabeled data can be viewed as drawn from the marginal distribution $p(x)$ as follows,

$$\mathcal{D}_u = \{x_i^u\}_{i=1}^{n_u} \sim p(x), \tag{3}$$

where $p(x) = \sum_{j=1}^{k} \theta_j p(x|y=j)$. For the $j$-th subproblem, the unlabeled data can be viewed as drawn from distribution $p^j(x) = \pi^j p_+^j + (1-\pi^j) p_-^j$, where $\pi^j = \sum_{i=j+1}^{k} \theta_i$.

The key idea to incorporate the unlabeled instances into the binary AUC optimization process is to treat the unlabeled instances as negative and then compare them with positive instances; treat them as positive and compare them with negative data (Wang et al. 2015). Thus, the AUC risk $R_{\mathrm{PU}}^j$ between positive and unlabeled instances and the AUC risk $R_{\mathrm{NU}}^j$ between unlabeled and negative instances can be defined as follow,

$$R_{\mathrm{PU}}^j = \mathbb{E}_{x_p^j \in \mathcal{D}_p^j}\left[\mathbb{E}_{x_u^j \in \mathcal{D}_u}\left[l_{01}\left(f(x_p^j), f(x_u^j)\right)\right]\right], \tag{4}$$

$$R_{\mathrm{NU}}^j = \mathbb{E}_{x_u^j \in \mathcal{D}_u}\left[\mathbb{E}_{x_n^j \in \mathcal{D}_n^j}\left[l_{01}\left(f(x_u^j), f(x_n^j)\right)\right]\right], \tag{5}$$

Xie and Li, (2018) have shown that $R_{\mathrm{PU}}^j$ and $R_{\mathrm{NU}}^j$ are equivalent to $R_{\mathrm{PN}}^j$ with a linear transformation as follows,

$$R_{\mathrm{PN}}^j = R_{\mathrm{PU}}^j + R_{\mathrm{NU}}^j - \frac{1}{2}. \tag{6}$$

Thus, the AUC risk $R^j_{\mathrm{PNU}}$ for the $j$-th binary semi-supervised problem is

$$R^j_{\mathrm{PNU}} = \gamma^j R^j_{\mathrm{PN}} + (1-\gamma^j)\left(R^j_{\mathrm{PU}} + R^j_{\mathrm{NU}} - \frac{1}{2}\right), \quad (7)$$

where the first term is the AUC risk computed from the labeled instances only, the second term is an estimation AUC risk using both labeled and unlabeled instances and $\gamma^j$ is trade-off parameter. Similar to Equation (2), the overall AUC risk for the $k-1$ hyperplanes in the S$^2$OR problem can be formulated as follows,

$$R_\mu = \frac{1}{k-1}\sum_{j=1}^{k-1} R^j_{\mathrm{PNU}}. \quad (8)$$

To avoid overfitting caused by directly minimizing Equation (8), a regularization term is usually added as follows,

$$\mathcal{L}(f) = \frac{\lambda}{2}\parallel f \parallel^2_{\mathcal{H}} + \frac{1}{k-1}\sum_{j=1}^{k-1} R^j_{\mathrm{PNU}}, \quad (9)$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm in RKHS $\mathcal{H}$, $\lambda > 0$ is regularization parameter .

## Random Fourier Feature

For any *continuous*, *real-valued*, *symmetric* and *shift-invariant* kernel function $k(x,x')$, according to Bochner Theorem (Rudin 2017), there exists a nonnegative Fourier transform function as $k(x,x') = \int_{\mathbb{R}^d} p(\omega)e^{j\omega^T(x-x')}d\omega$, where $p(w)$ is a density function associated with $k(x,x')$. The integrand $e^{j\omega^T(x-x')}$ can be replaced with $\cos\omega^T(x-x')$ (Rahimi and Recht 2008). Thus, the feature map for $m$ random features of $k(x,x')$ can be formulated as follows.

$$\phi_\omega(x) = \sqrt{1/D}[\cos(\omega_1^T x), \cdots, \cos(\omega_m^T x), \sin(\omega_1^T x), \cdots, \sin(\omega_m^T x)]^T,$$

where $\omega_i$ is randomly sampled according to the density function $p(\omega)$. Obviously, $\phi^T_\omega(x)\phi_\omega(x')$ is an unbiased estimate of $k(x,x')$.

## Quadruply Stochastic Gradient Method

Based on the definition of the ranking function $f \in \mathcal{H}$, we can obtain $\nabla f(x) = k(x,\cdot)$, and $\nabla \parallel f \parallel^2_{\mathcal{H}} = 2f$. To calculate the gradient of objective function, we use the squared pairwise loss $l_s(u,v)$ function to replace zero-one loss $l_{01}(u,v)$. Then we can obtain the full gradient of our objective function w.r.t. $f$ as follows,

$$\begin{aligned}\nabla\mathcal{L} = \frac{1}{k-1}\sum_{j=1}^{k-1}(&\gamma^j \mathbb{E}_{x^j_p\in\mathcal{D}^j_p}[\mathbb{E}_{x^j_n\in\mathcal{D}^j_n}[l_1'(f(x^j_p), f(x^j_n))k(x^j_p,\cdot)\\ &+ l_2'(f(x^j_p), f(x^j_n))k(x^j_n,\cdot)]]\\ &+ (1-\gamma^j)(\mathbb{E}_{x^j_p\in\mathcal{D}^j_p}[\mathbb{E}_{x^j_u\in\mathcal{D}_u}[l_1'(f(x^j_p), f(x^j_u))k(x^j_p,\cdot)\\ &+ l_2'(f(x^j_p), f(x^j_u))k(x^j_u,\cdot)]]\\ &+ \mathbb{E}_{x^j_u\in\mathcal{D}_u}[\mathbb{E}_{x^j_n\in\mathcal{D}^j_n}[l_1'(f(x^j_u), f(x^j_n))k(x^j_u,\cdot)\\ &+ l_2'(f(x^j_u), f(x^j_n))k(x^j_n,\cdot)]])) + \lambda f\end{aligned}$$

where $l_1'(u,v)$ denotes the derivative of $l_s(u,v)$ w.r.t. the first argument in the functional space, $l_2'(u,v)$ denotes the derivative of $l_s(u,v)$ w.r.t. the second argument in the functional space.

## Stochastic Functional Gradients

Directly calculating the full gradient is time-consuming. In order to reduce the computational complexity, we update the ranking function using a quadruply stochastic framework. For each subproblem, we randomly sample a positive instance $x^j_p$ from $\mathcal{D}^j_p$, a negative instance $x^j_n$ from $\mathcal{D}^j_n$ and an unlabeled instance $x_u$ from $\mathcal{D}_u$ in each iteration.

For convenience, we use $l^j_i$, $i = 1,\cdots,6$, to denote the abbreviation of $l_1'(f(x^j_p), f(x^j_n))$, $l_2'(f(x^j_p), f(x^j_n))$, $l_1'(f(x^j_p), f(x^j_u))$, $l_2'(f(x^j_p), f(x^j_u))$, $l_1'(f(x^j_u), f(x^j_n))$, $l_2'(f(x^j_u), f(x^j_n))$ in $j$-th subproblem, respectively. Then the stochastic gradient of Equation (8) w.r.t $f$ can be calculated by using these random instances,

$$\begin{aligned}\xi(\cdot) = \frac{1}{k-1}\sum_{j=1}^{k-1}(&\gamma^j(l^j_1 k(x^j_p,\cdot) + l^j_2 k(x^j_n,\cdot))\\ &+ (1-\gamma^j)(l^j_3 k(x^j_p,\cdot) + l^j_4 k(x^j_u,\cdot)\\ &+ l^j_5 k(x^j_u,\cdot) + l^j_6 k(x^j_n,\cdot)))\end{aligned} \quad (10)$$

## Kernel Approximation

When calculating the gradient $\xi(\cdot)$, we still need to calculate the kernel matrix. In order to further reduce the complexity, we introduce random Fourier features into gradient $\xi(\cdot)$. Then we can obtain the following approximated gradient,

$$\begin{aligned}\zeta(\cdot) = \frac{1}{k-1}\sum_{j=1}^{k-1}(&\gamma^j(l^j_1\phi_\omega(x^j_p)\phi_\omega(\cdot) + l^j_2\phi_\omega(x^j_n)\phi_\omega(\cdot))\\ &+ (1-\gamma^j)(l^j_3\phi_\omega(x^j_p)\phi_\omega(\cdot) + l^j_4\phi_\omega(x^j_u)\phi_\omega(x)\\ &+ l^j_5\phi_\omega(x^j_u)\phi_\omega(\cdot) + l^j_6\phi_\omega(x^j_n)\phi_\omega(\cdot)))\end{aligned} \quad (11)$$

Obviously, we have $\xi(\cdot) = \mathbb{E}_\omega[\zeta(\cdot)]$. Besides, since four sources of randomness of each subproblem, $x^j_p$, $x^j_n$, $x^j_u$, $\omega$, are involved in calculating gradient $\zeta(\cdot)$, we can denote the approximated gradient $\zeta(\cdot)$ as quadruply stochastic functional gradient.

## Update Rules

For convenience, we denote the function value as $h(x)$ while using the real gradient $\xi(\cdot)$, and $f(x)$ while using the approximated gradient $\zeta(\cdot)$. Obviously, $h(x)$ is always in the RKHS $\mathcal{H}$ while $f(x)$ may be outside $\mathcal{H}$. We give the update rules of $h(\cdot)$ as follows,

$$h_{t+1}(\cdot) = h_t(\cdot) - \eta_t\nabla\mathcal{L}(h) = \sum_{i=1}^{t} a^i_t \xi^i(\cdot), \quad \forall\, t > 1$$

where $\eta_t$ denotes the step size in $t$-th iteration, $a^i_t = -\eta_t\prod_{j=i+1}^{t}(1-\eta_j\lambda)$ and $h^1_0(\cdot) = 0$.

Since $\zeta(\cdot)$ is an unbiased estimate of $\xi(\cdot)$, they have the similar update rules. Thus, the update rule by using $\zeta_j(\cdot)$ is

$$f_{t+1}(\cdot) = f_t(\cdot) - \eta_t \nabla \mathcal{L}(f) = \sum_{i=1}^{t} a_t^i \zeta^i(\cdot), \quad \forall\, t > 1$$

where $f_0^1(\cdot) = 0$.

In order to implement the algorithm in a computer program, we introduce sequences of constantly-changing coefficients $\{\alpha_i\}_{i=1}^t$. Then the update rules can be rewritten as

$$f(x) = \sum_{i=1}^{t} \alpha_i \phi_\omega(x), \tag{12}$$

$$\begin{aligned}\alpha_i = &-\frac{\eta_i}{k-1} \sum_{j=1}^{k-1} (\gamma^j (l_1^j \phi_w(x_p^j) + l_2^j \phi_w(x_n^j)) \\ &+ (1-\gamma^j)(l_3^j \phi_w(x_p^j) + l_4^j \phi_w(x_u^j) \\ &+ l_5^j \phi_w(x_n^j) + l_6^j \phi_w(x_u^j))),\end{aligned} \tag{13}$$

$$\alpha_j = (1-\eta_i \lambda)\alpha_j, \ for\ j = 1, \cdots, i-1, \tag{14}$$

## Calculate the Thresholds

Since the thresholds are ignored in AUC optimization, an additional strategy is required to calculate them. As all the function values $f(x)$ of labeled instances are already known, the thresholds can be calculated by minimizing the following equation, which penalizes every erroneous threshold of all the binary subproblems (Fathony, Bashiri, and Ziebart 2017).

$$\min_{\mathbf{b}} \mathcal{L}_{AT} = \sum_{i=1}^{n_l} \left( \sum_{j=1}^{y_i - 1} (f(x_i) - b_j)^2 + \sum_{j=y_i}^{k} (b_j - f(x_i))^2 \right),$$

where $n_l$ denotes the number of labeled instances and $b_k = \infty$. Obviously, it is a Linear Programming problem and can be easily solved. Besides, the solution has following property (Proof in Appendix).

**Lemma 1** *Let* $\boldsymbol{b}^* = [b_1^*, \cdots, b_{k-1}^*]$ *be the optimal solution, we have that* $\boldsymbol{b}^*$ *is unique and* $b_1^* < \cdots < b_{k-1}^*$.

## Algorithms

The overall algorithms for training and prediction are summarized in Algorithm 1 and 2. Instead of saving all the random features, we following the *pseudo random number generator* setting of (Dai et al. 2014) with seed $i$ to generate random features in each iteration. We only need to save the seed $i$ and keep it aligned between training and prediction, then we can regenerate the same random features. We also use the coefficients to speed up calculating the function value. Specifically, each iteration of the training algorithm executes the following steps.

1. *Randomly Sample Data Instances:* We can randomly sample a batch instances from class $i, \cdots, k$ and unlabeled dataset respectively, and then conduct the data of $k-1$ subproblems instead of sampling instances for each subproblem.
2. *Approximate the Kernel Function:* Sample $\omega_i \sim p(\omega)$ with random seed $i$ to calculate the random features on-the-fly. We keep this seed aligned between prediction and training to regenerate the same random features.
3. *Update Coefficients:* We compute the current coefficient $\alpha_i$ in $i$-th loop, and then update the former coefficients $\alpha_j$ for $j = 1, \cdots, i-1$.

**Algorithm 1** QS$^3$ORAO

```
Input: $p(\omega), \phi_\omega(x), l(u,v), \lambda, \gamma_i, \sigma, k, t$.
Output: $\{\alpha_i\}_{i=1}^t, b_1, \cdots, b_{k-1}$
1:  for $i = 1, ..., t$ do
2:      for $j = 1, \cdots, k-1$ do
3:          Sample $x_p^j$ from $\mathcal{D}_{j+1} \cup \cdots \cup \mathcal{D}_k$.
4:          Sample $x_n^j$ from $\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_j$.
5:          Sample $x_u^j \sim \mathcal{D}_u$.
6:      end for
7:      Sample $\omega_i \sim p(\omega)$ with seed $i$.
8:      for $j = 1, \cdots, k-1$ do
9:          $f(x_i) =$ Predict$(x_i, \{\alpha_j\}_{j=1}^{i-1}, \{\beta_j\}_{j=1}^{i-1})$.
10:     end for
11:     $\alpha_i = -\frac{\eta_i}{k-1} \sum_{j=1}^{k-1} (\gamma(l_1^j \phi_w(x_p^j) + l_2^j \phi_w(x_n^j)) + (1-\gamma)(l_3^j \phi_w(x_p^j) + l_4^j \phi_w(x_u^j) + l_5^j \phi_w(x_n^j) + l_6^j \phi_w(x_u^j)))$
12:     $\alpha_j = (1-\eta_j \lambda)\alpha_j \ for\ j = 1, ..., i-1$.
13: end for
14: Minimize $\mathcal{L}_{AT}$ to find threshold $b_1, \cdots, b_{k-1}$.
```

**Algorithm 2** $f(x) =$ **Predict**$(x, \{\alpha_i\}_{i=1}^t)$

```
Input: $p(\omega), \phi_\omega(x)$
Output: $f(x)$
1: Set $f(x) = 0$.
2: for $i = 1, ..., t$ do
3:     Sample $\omega_i \sim p(\omega)$ with seed $i$.
4:     $f(x) = f(x) + \alpha_i \phi_\omega(x)$
5: end for
```

## Convergence Analysis

In this section, we prove that QS$^3$ORAO converges to the optimal solution at the rate of $O(1/t)$. We first give several assumptions which are common in theoretical analysis.

**Assumption 1** *There exists an optimal solution* $f^*$ *to the problem (9).*

**Assumption 2** *(Lipschitz continuous). The first order derivative of* $l_s(u,v)$ *is* $L_1$-***Lipschitz continous*** *in terms of* $u$ *and* $L_2$-***Lipschitz continous*** *in terms of* $v$.

**Assumption 3** *(Bound of derivative). Assume that, we have* $|l_1'(u,v)| \leq M_1$ *and* $|l_2'(u,v)| \leq M_2$*, where* $M_1 > 0$ *and* $M_2 > 0$.

**Assumption 4** *(Bound of kernel function). The kernel function is bounded, i.e.,* $k(x,x') \leq \kappa$*, where* $\kappa > 0$.

**Assumption 5** *(Bound of random features norm). The random features norm is bounded, i.e., $|\phi_\omega(x)\phi_\omega(x')| \leq \phi$.*

Then we prove that $f_t$ can converge to the optimal solution $f^*$ based on the framework in (Dai et al. 2014). Since $f_t$ may outside the RKHS, we use $h_t$ as an intermediate value to decompose the error between $f_t$ and $f^*$:

$$|f_t(x) - f^*(x)|^2 \leq 2|f_t(x) - h_t(x)|^2 + 2\kappa \parallel h_t - f^* \parallel_{\mathcal{H}}^2,$$

where the first term can be regarded as the error caused by random features and the second term can be regarded as the error caused by randomly sampling data instances. We first give the bound of these two errors in Lemma 2 and Lemma 4 respectively. All the detailed proofs are in Appendix.

**Lemma 2 (Error due to random features)** *Assume $\chi$ denote the whole training set. For any $x \in \chi$, we have*

$$\mathbb{E}_{x_p^t, x_n^t, x_u^t, w_t}\left[|f_t(x) - h_t(x)|\right] \leq B_{1,t+1}^2, \qquad (15)$$

*where $B_{1,t+1}^2 := M^2(\kappa + \phi)^2 \sum_{i=1}^t |a_t^i|$, $M = \frac{1}{k-1}\sum_{j=1}^{k-1}(2-\gamma^j)(M_1 + M_2)$, and $B_{1,1} = 0$.*

**Lemma 3** *Suppose $\eta_i = \frac{\theta}{i}$ $(1 \leq i \leq t)$ and $\theta\lambda \in (1,2) \cup \mathbb{Z}_+$. We have $|a_t^i| \leq \frac{\theta}{t}$ and $\sum_{i=1}^t |a_t^i|^2 \leq \frac{\theta^2}{t}$.*

**Remark 1** *According to Lemmas 2 and 3, the error caused by random features has the convergence rate of $O(1/t)$ with proper learning rate and $\theta\lambda \in (1,2)$.*

**Lemma 4 (Error due to random data)** *Set $\eta_t = \frac{\theta}{t}$, $\theta > 0$, such that $\theta\lambda \in (1,2) \cup \mathbb{Z}_+$, we have*

$$\mathbb{E}_{x_p^t, x_n^t, x_u^t, \omega_t}\left[\|h_{t+1} - f^*\|_{\mathcal{H}}^2\right] \leq \frac{Q_1^2}{t}, \qquad (16)$$

*where $Q_1 = \max\left\{\|f^*\|_{\mathcal{H}}, \frac{Q_0 + \sqrt{Q_0^2 + (2\theta\lambda - 1)(1+\theta\lambda)^2\theta^2\kappa M^2}}{2\theta\lambda - 1}\right\}$, $Q_0 = \sqrt{2}\kappa^{1/2}(\kappa + \phi)LM\theta^2$ and $L = \frac{1}{k-1}\sum_{j=1}^{k-1}(2 - \gamma^j)(L_1 + L_2)$.*

According to Lemma 2 and Lemma 4, we can bound the error between $f_t$ and $f^*$.

**Theorem 1 (Convergence in expectation)** *Let $\chi$ denote the whole training set in semi-supervised learning problem. Set $\eta_t = \frac{\theta}{t}$, $\theta > 0$, such that $\theta\lambda \in (1,2) \cup \mathbb{Z}_+$. $\forall x \in \chi$, we have*

$$\mathbb{E}_{x_t^p, x_t^n, x_t^u, \omega_t}\left[|f_{t+1}(x) - f^*(x)|^2\right] \leq \frac{2C^2 + 2\kappa Q_1^2}{t},$$

*where $C^2 = (\kappa + \phi)^2 M^2 \theta^2$.*

**Remark 2** *Theorem 1 means that for any given $x$, the evaluated value of $f_{t+1}$ at $x$ will converge to that of $f^*$ at the rate of $O(1/t)$. This rate is the same as that of standard DSG even though our problem is much more complicated and has multiple sources of randomness.*

Table 2: Datasets used in the experiments.

| | **Name** | **Features** | **Samples** | **classes** |
|---|---|---|---|---|
| **Discretized** | 3D | 3 | 434,874 | 5 |
| | Sgemm | 14 | 241,600 | 5 |
| | Year | 90 | 463,715 | 5 |
| | Yolanda | 100 | 400,000 | 5 |
| **Real-world** | Baby | 1000 | 160,792 | 5 |
| | Beauty | 1000 | 198,502 | 5 |
| | Clothes | 1000 | 278,677 | 5 |
| | Pet | 1000 | 157,836 | 5 |

## Experiments

In this section, we present the experimental results on various benchmark and real-world datasets to demonstrate the effectiveness and efficiency of our proposed Q$^3$ORAO.

### Experimental Setup

We compare the AUC results and running time of Q$^3$ORAO with other methods summarized as follows,

1. ***SVOREX***: Supervised OR algorithm proposed in (Chu and Keerthi 2007).
2. ***M-PNU-AUC***: Multi class version of PNU-AUC (Sakai, Niu, and Sugiyama 2018), which focuses on binary semi-supervised AUC optimization.
3. ***M-SAMULT***: Multi class version of SAMULT (Xie and Li 2018), which focuses on binary semi-supervised AUC optimization.

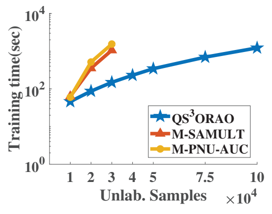
We implemented QSG-ORS$^2$AO, SVOREX and SAMULT in MATLAB. We used the MATLAB code from https://github.com/t-sakai-kure/PNU as the implementation of PNU-AUC. Originally, both PNU-AUC and SAMULT focus on binary semi-supervised AUC optimization problems. We extend them to multi-class version by using a multiclass training paradigm. Specifically, similar to our binary decomposition in our method, we use PNU-AUC and SAMUlT to training $k-1$ classifiers, $f_j(x)$, $j = 1, \cdots, k-1$, Then we calculate the average AUC of unlabeled by Equation (2). We denote these multiclass versions as M-PNU-AUC and M-SAMULT. For all algorithms, we use the squared pairwise loss function $l(u,v) = (1 - u + v)^2$ and Gaussian kernel $k(x,x') = \exp(\sigma \parallel x - x' \parallel^2)$. The hyper-parameters ($\lambda$ and $\sigma$) were chosen via 5-fold cross-validation from the region $\{(\lambda,\gamma)|2^{-3} \leq \lambda \leq 2^3, 2^{-3}\sigma \leq 2^3\}$. The trade-off parameters $\{\gamma^j\}$ for $k-1$ subproblems were searched from 0 to 1 at intervals of 0.1.

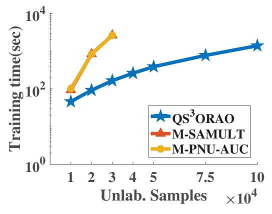
Note all the experiments were run on a PC with 56 2.2GHz cores and 80GB RAM and all the results are the average of 10 trials.

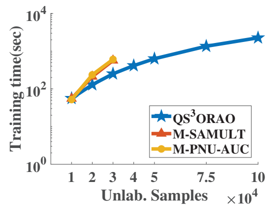
### Datasets

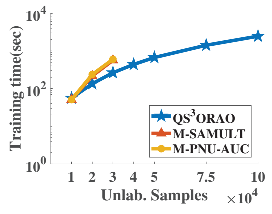
Table 2 summarizes 4 regression datasets collected from UCI, LIBSVM repositories and 4 real-world datasets from **Amazon product datasets**[1]. We discretize the regression datasets into equal-frequency bins. For real-world datasets,

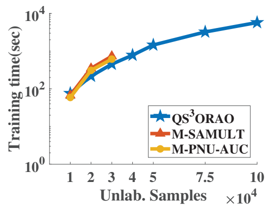
[1]http://jmcauley.ucsd.edu/data/amazon/

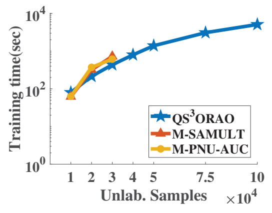

(a) 3D

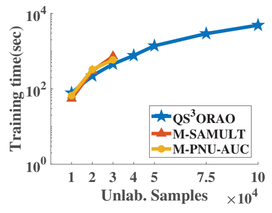

(b) Sgemm


(c) Year


(d) Yolanda


(e) Baby


(f) Beauty


(g) Clothes

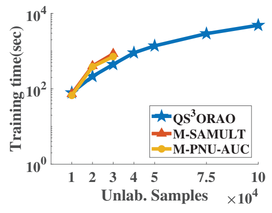

(h) Pet

Figure 1: The training time of QS$^3$ORAO, M-SAMULT and M-PNU-AUC against different sizes of unlabeled samples, where the sizes of labeled samples are fixed at 500. (The lines of M-SAMULT and M-PNU-AUC are incomplete because their implementations crash on larger training sets.)

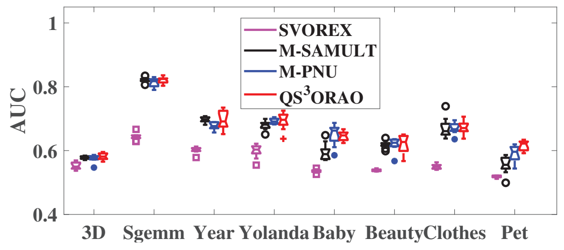


Figure 2: The boxplot of AUC results on unlabeled datasets for SVOREX, M-PNU-AUC, M-SAMULT and our QS$^3$ORAO.

we first use TF-IDF to process text data, and then reduce the data dimensions to 1000 by using SVD. To conduct the experiments for semi-supervised problems, we randomly sample 500 labeled instances and drop labels of the rest instances. All the data features are normalized to $[0, 1]$ in advance.

## Results and Disscussion

Figure 2 presents the AUC results on the unlabeled dataset of these algorithms. The results show that in most cases, our proposed QS$^3$ORAO has the highest AUC results. In addition, we also compare the AUC results with supervised method SVOREX which uses 500 labeled instances to train a model. Obviously, our semi-supervised learning method has higher AUC than SVOREX, which demonstrate that incorporating unlabeled instances can improve the performance.

Figure 1 presents the training time against different size of unlabeled samples. The two lines of M-SAMULT and M-PNU-AUC are incomplete. This is because these two methods need to save the whole kernel matrix, and as unlabeled data continues to increase, they are all out of memory. In contrast, QS$^3$ORAO only need to keep $m$ random features in each iteration. This low memory requirement allows it to do an efficient training for large scale datasets. Besides, we can easily find that our method is faster than M-SAMULT and M-PNU-AUC when the number of unlabeled instances is larger than 10000. This is because the M-SAMULT and M-PNU-AUC need $O(n^3)$ operations to compute the inverse of kernel matrix. Differently, QS$^3$ORAO uses RFF to approximate the kernel function, and each time it only needs $O(Dn)$ operations to calculate the random features with seed $i$.

Based on these results, we conclude that QSG-S2AUC is superior to other state-of-the-art algorithms in terms of efficiency and scalability, while retaining similar generalization performance.

## Conclusion

In this paper, we propose an unbiased objective function of semi-supervised OR AUC optimization and propose a novel scalable algorithm, QS$^3$ORAO to solve it. We decompose the original problem by $k - 1$ parallel hyperplanes to $k - 1$ binary semi-supervised AUC optimization problems. Then we use a DSG-based method to achieve the optimal solution. Even though this optimization process contains four sources of randomness, theoretically, we prove that QS$^3$ORAO has a convergence rate of $O(1/t)$. The experimental results on various benchmark datasets also demonstrate the superiority of the proposed QS$^3$ORAO.

## Acknowledgments

This work was supported by Six talent peaks project (No. XYDXX-042) and the 333 Project (No. BRA2017455) in Jiangsu Province and the National Natural Science Foundation of China (No: 61573191).

## References

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7(Nov):2399–2434.

Chapelle, O.; Scholkopf, B.; and Zien, A. 2009. Semi-supervised learning. *IEEE Transactions on Neural Networks* 20(3):542–542.

Chu, W., and Keerthi, S. S. 2007. Support vector ordinal regression. *Neural computation* 19(3):792–815.

Dai, B.; Xie, B.; He, N.; Liang, Y.; Raj, A.; Balcan, M.-F. F.; and Song, L. 2014. Scalable kernel methods via doubly stochastic gradients. In *Advances in NIPS*, 3041–3049.

Drineas, P., and Mahoney, M. W. 2005. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research* 6(Dec):2153–2175.

Fathony, R.; Bashiri, M. A.; and Ziebart, B. 2017. Adversarial surrogate losses for ordinal regression. In *Advances in NIPS*, 563–573.

Fine, S., and Scheinberg, K. 2001. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research* 2(Dec):243–264.

Fujino, A., and Ueda, N. 2016. A semi-supervised auc optimization method with generative models. In *ICDM*, 883–888.

Fullerton, A. S., and Xu, J. 2012. The proportional odds with partial proportionality constraints model for ordinal response variables. *Social science research* 41(1):182–198.

Fürnkranz, J.; Hüllermeier, E.; and Vanderlooy, S. 2009. Binary decomposition methods for multipartite ranking. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 359–374. Springer.

Gao, W., and Zhou, Z.-H. 2015. On the consistency of auc pairwise optimization. In *IJCAI*, 939–945.

Gao, W.; Jin, R.; Zhu, S.; and Zhou, Z.-H. 2013. One-pass auc optimization. In *International Conference on Machine Learning*, 906–914.

Geng, X.; Gu, B.; Li, X.; Shi, W.; Zheng, G.; and Huang, H. 2019. Scalable semi-supervised svm via triply stochastic gradients. In *IJCAI*.

Gu, B.; Sheng, V. S.; Tay, K.; Romano, W.; and Li, S. 2015. Incremental support vector learning for ordinal regression. *IEEE Transactions on Neural Networks and Learning Systems* 26:1403–1416.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in NIPS*, 8527–8537.

Kim, K.-j., and Ahn, H. 2012. A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach. *Computers & Operations Research* 39(8):1800–1811.

Li, X.; Gu, B.; Ao, S.; Wang, H.; and Ling, C. X. 2017. Triply stochastic gradients on multiple kernel learning. In *UAI*.

Liu, Y.; Liu, Y.; Zhong, S.; and Chan, K. C. 2011. Semi-supervised manifold ordinal regression for image ranking. In *Proceedings of the 19th ACM international conference on Multimedia*, 1393–1396. ACM.

Niu, Z.; Zhou, M.; Wang, L.; Gao, X.; and Hua, G. 2016. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the CVPR*, 4920–4928.

Rahimi, A., and Recht, B. 2008. Random features for large-scale kernel machines. In *Advances in NIPS*, 1177–1184.

Rudin, W. 2017. *Fourier analysis on groups*. Courier Dover Publications.

Sakai, T.; Niu, G.; and Sugiyama, M. 2018. Semi-supervised auc optimization based on positive-unlabeled learning. *Machine Learning* 107(4):767–794.

Seah, C.-W.; Tsang, I. W.; and Ong, Y.-S. 2012. Transductive ordinal regression. *IEEE transactions on neural networks and learning systems* 23(7):1074–1086.

Shi, W.; Gu, B.; Li, X.; Geng, X.; and Huang, H. 2019. Quadruply stochastic gradients for large scale nonlinear semi-supervised auc optimization. In *IJCAI*.

Smola, A. J., and Schölkopf, B. 2000. Sparse greedy matrix approximation for machine learning.

Srijith, P.; Shevade, S.; and Sundararajan, S. 2013. Semi-supervised gaussian process ordinal regression. In *Joint European conference on machine learning and knowledge discovery in databases*, 144–159. Springer.

Tsuchiya, T.; Charoenphakdee, N.; Sato, I.; and Sugiyama, M. 2019. Semi-supervised ordinal regression based on empirical risk minimization. *CoRR* abs/1901.11351.

Uematsu, K., and Lee, Y. 2014. Statistical optimality in multipartite ranking and ordinal regression. *IEEE transactions on pattern analysis and machine intelligence* 37(5):1080–1094.

Waegeman, W., and De Baets, B. 2010. A survey on roc-based ordinal regression. In *Preference learning*. Springer. 127–154.

Waegeman, W.; De Baets, B.; and Boullart, L. 2008. Roc analysis in ordinal regression learning. *Pattern Recognition Letters* 29(1):1–9.

Wang, S.; Li, D.; Petrick, N.; Sahiner, B.; Linguraru, M. G.; and Summers, R. M. 2015. Optimizing area under the roc curve using semi-supervised learning. *Pattern recognition* 48(1):276–287.

Xie, Z., and Li, M. 2018. Semi-supervised auc optimization without guessing labels of unlabeled data.

Yan, H. 2014. Cost-sensitive ordinal regression for fully automatic facial beauty assessment. *Neurocomputing* 129:334–342.

Zhang, C.; Ren, D.; Liu, T.; Yang, J.; and Gong, C. 2019. Positive and unlabeled learning with label disambiguation. In *Proceedings of IJCAI-19*, 4250–4256.