Scalable Kernel Ordinal Regression via Doubly Stochastic Gradients

Bin Gu, Member, IEEE, Xiang Geng, Xiang Li, Wanli Shi, Guansheng Zheng, Cheng Deng, Student Member, IEEE, and Heng Huang, Member, IEEE

Abstract—Ordinal regression (OR) is one of the most important machine learning tasks. The kernel method is a major technique to achieve nonlinear OR. However, traditional kernel OR solvers are inefficient due to increased complexity introduced by multiple ordinal thresholds as well as the cost of kernel computation. Doubly stochastic gradient (DSG) is a very efficient and scalable kernel learning algorithm that combines random feature approximation with stochastic functional optimization. However, the theory and algorithm of DSG can only support optimization tasks within the unique reproducing kernel Hilbert space (RKHS), which is not suitable for OR problems where the multiple ordinal thresholds usually lead to multiple RKHSs. To address this problem, we construct a kernel whose RKHS can contain the decision function with multiple thresholds. Based on this new kernel, we further propose a novel DSG-like algorithm, DSGOR. In each iteration of DSGOR, we update the decision functional as well as the function bias with appropriately set learning rates for each. Our theoretic analysis shows that DSGOR can achieve $O(1/t)$ convergence rate, which is as good as DSG, even though dealing with a much harder problem. Extensive experimental results demonstrate that our algorithm is much more efficient than traditional kernel OR solvers, especially on large-scale problems.

Index Terms—Doubly stochastic gradients (DSGs), kernel learning, ordinal regression (OR), random features.

Manuscript received June 2, 2019; revised November 24, 2019, March 28, 2020, and July 2, 2020; accepted August 8, 2020. Date of publication August 28, 2020; date of current version August 4, 2021. The work of Bin Gu was supported in part by the Natural Science Foundation under Grant BK20161534, in part by the Six Talent Peaks Project under Grant XDYDXX-042, and in part by the Project Funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions. (Bin Gu, Xiang Geng, and Xiang Li contributed equally to this work.) (Corresponding author: Heng Huang.)

Bin Gu is with the Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing 210044, China, also with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China, and also with JD Finance America Corporation, Mountain View, CA 94043 USA (e-mail: jsgubin@gmail.com).

Xiang Geng, Wanli Shi, and Guansheng Zheng are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: gengxiang@nuist.edu.cn; wanlishi@nuist.edu.cn; zgs@nuist.edu.cn).

Xiang Li is with the Computer Science Department, University of Western Ontario, London, ON N6A 3K7, Canada (e-mail: xliang2@uwvo.ca).

Cheng Deng is with the School of Electronic Engineering, Xidian University, Xi’an 710114, China (e-mail: chdeng@mail.xidian.edu.cn).

Heng Huang is with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15260 USA, and also with JD Finance America Corporation, Mountain View, CA 94043 USA (e-mail: heng.huang@pitt.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.3015937

I. INTRODUCTION

A S AN important machine learning model, ordinal regression (OR) has been playing a key role in many real-world applications. For example, the goal of information retrieval is to predict the relevance level of the references with respect to a given textual query, using a rank scale, such as definitely, possibly, or not relevant. In this scenario, the data labels are similar to multiclassification, which are marked by a set of categories. However, the ordering information among the labels makes OR different from multiclassification. On the other hand, OR is similar to metric regression [7] in maintaining the ordering information. However, the multiple thresholds between the ranks are not defined, which makes metric regression different from OR.

The kernel method [8] is one major technique to achieve nonlinear learning. It owes its name to the use of kernel functions that define the nonlinear mapping from input space to reproducing kernel Hilbert space (RKHS) through defining the inner product in RKHS. A lot of kernel OR algorithms have been proposed in the last two decades. Some of them are built on the simple idea of solving OR by kernel regression or classification, e.g., support vector regression (SVR) [9] or support vector cost-sensitive classification [10]. Another idea is to decompose the OR problem into several binary kernel classification tasks and then use the different ordinal decomposition matrices to predict class labels [11], [12]. Although they decompose the problem according to ranks, the ordinal relation between different subproblems is neglected and results in ambiguities when predicting. Threshold models, which are the most popular methods for OR problems, provide another perspective that learning global ordinal information from multiple thresholds. Especially, the threshold-based approach aims to learn a decision function mapping an input $x$ to 1-D real space and uses multiple thresholds to discretize the 1-D real space.

In the last two decades, many effective threshold-based approaches have been proposed for OR problems [13]. For example, Crammer and Singer [14] proposed an extended perceptron algorithm incorporating multiple thresholds to solve OR problems. Lin and Li [15] proposed an OR boosting algorithm by constructing thresholded ensembles. The proportional odds model (POM) [16] extended the logistic regression with different thresholds to estimate cumulative probabilities predicting the ordinal classes. To address nonlinear OR problems, many nonlinear threshold-based OR algorithms have been proposed by using kernels, from support vector machine (SVM) [17], [18] formulations [2], [19], [20] to
Gaussian processes [4] or kernel discriminant learning [21] (see Section II for a brief review).

In the big data era, machine learning algorithms often have to deal with large-scale data sets. However, existing kernel OR methods are inefficient on large-scale problems due to the existence of the large kernel matrix as well as the multiple thresholds that the model has to be aligned to. On the one hand, a kernel matrix often requires \( O(N^2) \) memory for storage and computation [22]. On the other hand, SVM-based methods often need to penalize each sample multiple times for different threshold violation, which results in a larger problem size \( N \), as we summarize in Table I. As a result, even though SVM-based methods can be solved by a properly modified state-of-the-art SVM solvers (e.g. LIBSVM), \( O(N^k) \) computation is still needed where \( 1 < k < 2.3 \) [23]. Besides, most of the probabilistic OR methods suffer \( O(N^3) \) computational cost because they need to compute the inverse of the kernel matrix. Even for the sparse modeling approach, the computation cost \( O(NS^2) \) is close to \( O(N^3) \) because a large \( S \) (the size of a representative data sample) is needed to approach a good generalization performance.

Recently, a scalable doubly stochastic gradient (DSG) [24]–[28] algorithm was proposed to achieve large-scale kernel learning. DSG is a stochastic gradient (SG) descent method that works on convex optimizations over functions in RKHS. Especially, in each iteration, DSG samples a random data instance to compute the SG in RKHS and then samples random features to approximate the kernel function in SG that becomes a doubly SGD program. In this way, a stochastic approximation of the functional gradient can be computed and used for updating the model. The theoretic analysis shows that such a doubly SG optimization algorithm can converge efficiently. In a word, it avoids computing and storing a kernel matrix while enjoys nice convergence property toward the optimal solution by using stochastic optimization over the functional space. As far as we know, DSG is the most scalable kernel approximation method. However, it is not trivial to extend existing algorithms and theoretical analysis of DSG to kernel OR problems due to the following two reasons.

1) Multiple Thresholds: DSG can only handle one decision function with a fixed threshold (bias) of zero. As we will demonstrate later, such a decision function can only act as a predecision function in the kernel OR setting. In fact, kernel OR concerns multiple parallel decision functions each with a different threshold; and they need to be learned jointly.

2) Functional Gradients: In [24], the functional gradient involved. As mentioned earlier, it is not trivial to extend the theoretic analysis of the DSG method [24] to DSGOR. Most importantly, whether or not the decision functions jointly learned across multiple subproblems would converge is unknown. We have established a new theoretic analysis for DSGOR that guarantees its fast convergence.

II. RELATED WORKS

In this section, we give a brief review of kernel approximation and kernel OR methods.
A. Kernel Approximation

There are many kernel approximation methods proposed to address the scalability issue of kernel regression and classification. For instance, low-rank factors are used to approximate the kernel matrix by $K \approx A^T A$, with $A \in \mathbb{R}^{m \times n}$ [29]. The computation and memory costs of this method are $O(nm^2 + nm)$ and $O(nn)$, respectively, which highly depends on rank $m$. Rahimi and Recht [30] provided another method that uses random features to approximate the map function explicitly. The approximation with $m$ random features only needs $O(nm)$ processing time and $O(nm)$ memory requirement. However, as analyzed in [31]–[33], the rank for low rank and the number of random features need to be $O(n)$ to obtain a good generalization ability. To further improve the random features approximation method, Dai et al. [24] proposed DSG descent algorithm. For the $t$th iteration, DSG only needs $O(td)$ computation and $O(t)$ memory and enjoys a $O(1/t)$ convergence rate. However, it is not trivial to extend original DSG to kernel OR as discussed previously.

B. Ordinal Regression

Gutiérrez et al. [13] proposed a taxonomy of OR methods, including the naive approaches, the ordinal binary decompositions, and the threshold models. Naive approaches directly cast OR into regression [34] or classification [35], [36] problem and usually have a bad performance. Ordinal binary decomposition methods decompose OR into multiple ordinal binary classification problem. According to the decompositions pattern, these methods can be divided into following groups: “OrderedPartitions” [11], “OneVsNext” [37], “OneVsFollowers” [38], and “OneVsPrevious” [39]. Threshold models map the input features to a real number and using multiple thresholds to represent intervals between different classes. Varieties of threshold-based methods have been proposed based of different technique, such as ensembles [40], perceptron learning [14], or discriminant learning [21]. Most of threshold-based kernel OR methods focus on SVM formulations due to the good performance. Especially, Shashua and Levin [1] have proposed the SLA algorithm to handle OR problem with large margin principle. However, they ignored that the thresholds need to be ordered at the solution. To address this issue, Chu and Keerthi [2] proposed two new SVOR formulations with explicit/implicit constraints (i.e., SVOREX and SVORIM). Later, Lin and Li [3] proposed a general SVM-based framework REDSVM. Some probabilistic methods also have been proposed [41], and Chu and Ghahramani [4] and Van Gestel et al. [5] assume that the data are mapped to latent variables by a Gaussian process or logistic regression. Furthermore, Gu et al. [42] proposed an effective incremental method for SVOR, which is very suitable for an online scenario. However, this method needs to compute an inverse matrix of the size of $(2|\gamma| + 2 + m')^2$ in every iteration, which results in a huge computational cost, where $m'$ is the size of margin support vectors strictly on the margins and $|\gamma|$ denotes the number of ranks. Besides, some methods have been proposed to handle specific situations. For example, an iterative oversampling approach has been proposed to handle imbalanced ordinal data in [43]. A multicriteria learning method has been proposed in [44] to take into consideration different criteria in order to obtain a trustworthy outcome.

III. Preliminaries

In this section, we first introduce the reduction framework for OR that derives our objective function. Then, we give a brief review of random feature approximation and the DSG algorithm [24].

A. Random Feature Approximation

The random feature is a powerful technique to make kernel methods scalable. It uses the intriguing duality between kernels and stochastic processes. Especially, according to the Bochner theorem [45], for any PD kernel $k_0(v, \cdot)$, there exists a set $\Omega$, a probability measure $P$, and a random feature map $\phi_0^0(v)$, such that

$$k_0(v, x') = \int_{\Omega} \phi_0^0(v) \phi_0^0(x') dP(\omega). \quad (1)$$

In this way, the value of the kernel function can be approximated by explicitly computing random features $\phi_0^0(x')$, i.e.,

$$k_0(v, x') \approx \frac{1}{m} \sum_{i=1}^{m} \phi_0^i(v) \phi_0^i(x') \quad (2)$$

where $m$ is the number of random features. Using the Gaussian RBF kernel as a concrete example, it yields a Gaussian distribution $P(\omega)$ over random feature maps of the Fourier basis functions $\phi_0^0(x') = \sqrt{2} \cos(\omega_0^0 x + b)$ to compute its feature mapping, where $\omega_0$ is drawn from $P(\omega)$ and $b$ is drawn uniformly form $[0, 2\pi]$ [30]. Moreover, many random feature construction methods have been proposed for various kernels, such as dot-product kernels [46] and Laplacian kernels [47].

B. Doubly Stochastic Gradient

The theory of RKHS [48] provides a rigorous mathematical framework for studying optimization problems in the functional space. Especially, we know that every PD kernel $k_0(v, \cdot)$ has a corresponding RKHS $\mathcal{H}_0$. An RKHS $\mathcal{H}_0$ is a Hilbert space of functions from $\mathcal{X}$ to $\mathbb{R}$, which has the reproducing property, i.e., $\forall x \in \mathcal{X}$ and $\forall f_0 \in \mathcal{H}_0$, we always have $\langle f_0(\cdot), k_0(x, \cdot) \rangle_{\mathcal{H}_0} = f_0(x)$.

Besides, functional gradient in RKHS $\mathcal{H}_0$ can be computed as $\nabla f_0(x) = k_0(x, \cdot)$ and $\|f_0\|_{\mathcal{H}_0}^2 = 2 f_0$, where $\|\cdot\|_{\mathcal{H}_0}$ denotes the norm [49] in RKHS $\mathcal{H}$. Naturally, when random features are used, functional gradients can be approximated as

$$\nabla f_0(x) \approx \frac{1}{m} \sum_{i=1}^{m} \phi_0^i(v) \phi_0^i(x) \quad (3)$$

Given the loss function $l(u, v)$, a randomly sampled data instance $x_i$, and a random feature $\omega_{i0}$, the DSG of $l(f_0(x_i), y_i)$ on RKHS is given as

$$\nabla l(f_0(x_i), y_i) \approx l'(f_0(x_i), y_i) \phi_0^{i0}(x_i) \phi_0^{00}(\cdot) \quad (4)$$

In comparison, the traditional SG on RKHS is

$$\nabla l(f_0(x_i), y_i) = l'(f_0(x_i), y_i) k_0(x_i, \cdot). \quad (5)$$
It is obvious that the advantage of DSGs is that it can be computed without explicitly knowing the kernel mapping function.

C. Reduction Framework

In this article, we focus on the reduction framework (also referred to as all thresholds (ATs) loss framework) that is mostly popular for OR. Consider a training data set \( X := \{ x_i \in \mathbb{R}^d \}_{i=1}^{n} \), where the label (rank) of each instance \( x_i \) is known. Let the label set be \( \mathcal{Y} = \{ 1, 2, \ldots, |\mathcal{Y}| \} \). AT loss reduces an OR problem to learning \( r = |\mathcal{Y}| - 1 \) binary classification subproblems jointly. For \( j = 1, 2, \ldots, r \), the \( j \)th subproblem answers the question: is the rank of \( x_i \) greater than \( j \)? We have that \( y_j = 1 \) if the answer is yes and \( y_j = -1 \) otherwise.

The decision function of the \( j \)th binary subproblem is denoted as \( F_j(x) = f_0(x) + b_j \), where \( f_0(x) = \langle f_0, \phi_0(x) \rangle \) if given a positive definite (PD) kernel \( k_0(x,x') = \langle \phi_0(x), \phi_0(x') \rangle \), and we denote \( f_0 \) as the predecision function. The AT loss is simply the summation of all the losses for each subproblem, i.e.,

\[
L(F, x, y) = \frac{1}{r} \sum_{j=1}^{r} l(F_j(x), y_j)
\]

where \( l(u, y) \) can be common loss function for binary classification, as summarized in Table II.

Thus, the objective function of OR can be written as follows:

\[
\min_{f_0, b} R(f_0, b) := \mathbb{E}_{(x,y)}[L(F, x, y)] + \frac{\nu}{2} \| f_0 \|_{L_0}^2
\]

where \( b = \{ b_1, \ldots, b_r \} \), \( \mathcal{H}_0 \) is the RKHS induced by kernel \( k_0 \), and \( \nu > 0 \) is the regularization parameter. With this framework, the prediction function can be obtained by \( h(x) := 1 + \sum_{j=1}^{r} \mathbb{I}[F_j(x) > 0] \), where the Boolean function \( \mathbb{I}[\cdot] \) is 1 if the inner condition is true, and 0 otherwise.

The AT loss has been applied in many successful OR algorithms, such as ADAOR [15], SVORIM [2], and REDSVM [3]. There are two main theoretical advantages of AT loss.

1) Ordered Thresholds: In AT methods, the multiple thresholds are ordered automatically at the optimal solution [2], [3], which is important to obtain good generalization [13].

2) Fisher Consistency: AT can be seen as a surrogate loss of the absolute error, which is the canonical OR loss function. The Fisher consistency is a desirable property for surrogate loss functions, which implies that optimizing the surrogate loss would yield the best possible model for the true loss in the limit [50]. As given in [51], all the loss functions listed in Table II make the AT loss Fisher consistent.

IV. DOUBLY STOCHASTIC GRADIENT DESCENT FOR KERNEL ORDINAL REGRESSION

In standard DSG, after we randomly sample data instance \( x_i \) and random feature \( \omega_i \), the stochastic functional gradients could be computed easily using (4). However, for kernel OR problem (7), we need to jointly learn multiple decision functions \( F_j \)'s with different biases. Especially, \( F_j(\cdot) = f_0(\cdot) + b_j \) may not be in the same functional space as the predecision function \( f_0 \) (i.e., \( \mathcal{H}_0 \)) due to the existence of the bias term \( b_j \). In fact, each \( F_j \) may have a different RKHS. Without knowing \( F_j \)'s functional space and the associated kernel, we are not able to compute the stochastic functional gradient of \( l(F_j(x), y_j) \).

We will solve this challenging problem.

A. Reparameterization

Similar to [3], we construct a new kernel and derive its corresponding RKHS \( \mathcal{H} \). We will show that \( \mathcal{H} \) incorporates

<table>
<thead>
<tr>
<th>Name</th>
<th>( l(u, y) )</th>
<th>( l'(u, y) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hinge</td>
<td>max{0, 1 - yu}</td>
<td>{0 if ( yu \geq 1 ), (-y) if ( yu &lt; 1 ) }</td>
</tr>
<tr>
<td>Squared Hinge</td>
<td>( \frac{1}{2} \max{0, 1 - yu} )</td>
<td>{0 if ( yu \geq 1 ), ( uy ) if ( yu &lt; 1 ) }</td>
</tr>
<tr>
<td>Logistic</td>
<td>( \log(1 + \exp(-yu)) )</td>
<td>( -y \exp(-yu) )</td>
</tr>
<tr>
<td>Exponential</td>
<td>( \exp(-yu) )</td>
<td>( -uy )</td>
</tr>
<tr>
<td>Squared</td>
<td>( \frac{1}{2}(1 - yu)^2 )</td>
<td>( -uy )</td>
</tr>
</tbody>
</table>

Algorithm 1 DSGOR Train

Input: \( \mathbb{P}(\omega), \phi_0(\cdot), \theta, f(\cdot, \cdot), \nu \)

1: Set \( \alpha = 0, \beta = 0 \)
2: for \( i = 1, \ldots, t \) do
3: \( \gamma_i = \frac{\beta_i}{\tau}, \gamma_{i-1} = \frac{\gamma_i}{\tau} \) \# The constant \( \tau \) is a hyperparameter.
4: Sample \( (x^i, y^i) \sim \mathbb{P}(x, y) \)
5: Sample \( a'^i \sim \mathbb{P}(a) \) with seed \( i \)
6: for \( j = 1, \ldots, r \) do
7: \( b_j = b_j - \gamma_i \frac{1}{2} \mathbb{E}_{(x^i, y^i)}[l'(f_0([x^i, e^i], y^i)] \) \# The constant \( \tau \) is a hyperparameter.
8: end for
9: for \( j = 1, \ldots, r \) do
10: \( b_j = b_j - \gamma_i \frac{1}{2} \mathbb{E}_{(x^i, y^i)}[l'(f_0([x^i, e^i], y^i)] \phi_{a^i}([x^i, e^i]) \) \# The constant \( \tau \) is a hyperparameter.
11: end for

Output: \( \{ a'^{i1}, \ldots, a'^{ir} \}, \{ b'^{i1}, \ldots, b'^{iri} \} \)

Algorithm 2 DSGOR Predict

Input: \( \mathbb{P}(\omega), \phi_0(x), [x, e^i], \{ a'^{i1}, \ldots, a'^{iri} \}, \{ b'^{i1}, \ldots, b'^{iri} \} \)

1: Set \( f_0(x) = 0, b_j = 0 \)
2: for \( i = 1, \ldots, t \) do
3: Sample \( a'^i \sim \mathbb{P}(a) \) with seed \( i \)
4: \( f_0(x) = f_0(x) + a'^i \phi_{a^i}(x) \)
5: \( b_j = b_j + b_j \phi_{a^i}(x) \)
6: end for
7: \( f_b([x^i, e^i]) = f_0(x) + b_j \)

Output: \( f_b([x^i, e^i]) \)
both \( f_0 \) and all \( F_j \)'s. The new kernel is defined as
\[
  k(x, x') := k_0(x, x') + \langle x_e, x'_e \rangle = \langle \phi(x), \phi(x') \rangle
\]
where \( x := [x, x_e] \), and \( x_e \) is an undetermined one-hot vector indicating the \( j \)th subproblem with \( r \) dimensions. If we are explicitly dealing with the \( j \)th subproblem, we have \( x := x_j = [0, \ldots, 0, 1, 0, \ldots, 0]. \) To represent the relationship between \( f_0 \) and \( F_j \), for any given \( b \in \mathbb{R}^r \) and \( f_0 \in \mathcal{H}_0 : \mathcal{X} \to \mathbb{R} \), we define a new function \( f_b : \mathcal{X} \times \mathbb{R}^r \to \mathbb{R} \) as
\[
  f_b(x, x_e) = f_0(x) + \langle b, x_e \rangle.
\]

**Remark 1:** If \( k_0(\cdot, \cdot) \) is a PD kernel, then \( k(\cdot, \cdot) \) is also a PD kernel with one of its feature map being \( \phi(x) = [\phi_0(x), x_e]. \) The corresponding random feature map \( \phi_{\omega_0} \) can be constructed as
\[
  \phi_{\omega_0}([x, x_e]) = [\phi_0^0(x), x_e].
\]

The RKHS of kernel \( k(\cdot, \cdot) \) can be derived using the following inner product space \( \mathcal{H} \):
\[
  \langle f_1([x, x_e]), g_2([x, x_e]) \rangle_\mathcal{H} = \langle f_0(x), g_0(x') \rangle_{\mathcal{H}_0} + \langle b_1 \cdot x_e, b_2 \cdot x'_e \rangle
\]
where \( f_0 \) and \( g_0 \) can be any function in \( \mathcal{H}_0. \)

**Theorem 1:** If \( k_0(\cdot, \cdot) \) is the theoretical foundation for deriving the functional gradient of any \( f_b \in \mathcal{H}, \) and we have \( \nabla f_b([x, x_e]) = k([x, x_e], \cdot) \) and \( \nabla \| f_b \|_{\mathcal{H}}^2 = \nabla \langle f, f_b \rangle_{\mathcal{H}} = 2f_b. \) Furthermore, for any \( b \) and \( f_0 \), we always have \( f_b \in \mathcal{H}. \) It is interesting to note that when \( b = 0 \), we have that \( f_{b=0}([x, x_e]) = f_0(x). \) With possible abuse of notation, we will use \( f_0 \) to denote both the function \( f_0 \in \mathcal{H}_0 \) and the function \( f_{b=0} \in \mathcal{H}. \)

Again that the function value of \( f_b([x, x_e]) \) is the same as \( f_j(x), \) hence we are able to reparameterize the AT loss as
\[
  L(F, x, y) = L(f_b, x, y) = \frac{1}{r} \sum_{j=1}^{r} l(f_b([x, e]), y_j)
\]
and the objective function as
\[
  \min_{f_b \in \mathcal{H}} R(f_b) = \mathbb{E}_{(x, y) \sim L} L(f_b, x, y) + \frac{\nu}{2} \| f_0 \|_{\mathcal{H}}^2.
\]

Using this reparameterization, the functional gradient of kernel OR can be computed directly.

**B. DSGOR Algorithm**

According to the definition of function \( f_b, \) i.e., \( f_b([x, x_e]) = f_0(x) + \langle b, x_e \rangle, \) we can divide SGs for \( R(f_b) \) into two parts
\[
  \nabla R(f_b) = \nabla R(f_0, b) = [\nabla_1 R(f_0, b), \nabla_2 R(f_0, b)]
\]
where the first term is the SGs for updating \( f_0 \) and the other for \( b. \) Especially, we have that
\[
  \nabla_1 R(f_0, b) = \frac{1}{r} \sum_{j=1}^{r} \xi_j(\cdot) + \nu f_0(\cdot)
\]
\[
  \nabla_2 R(f_0, b) = \frac{1}{r} [\tilde{\xi}_1(\cdot), \tilde{\xi}_2(\cdot), \ldots, \tilde{\xi}_r(\cdot)]
\]
where \( \xi_j(\cdot) \) is the gradient contributed by the loss in the \( j \)th subproblem, i.e., \((\partial l(f_b([x, e]), y_j))/\partial f_0(\cdot)). \) It can be computed using the chain rule
\[
  \xi_j(\cdot) = \sum_{l=1}^{r} \left( l(f_b([x, e]), y_j) \right) \frac{\partial f_l(\cdot)}{\partial f_0(\cdot)}
\]
\[
  = l(f_b([x, e]), y_j) \nabla f_b(x) \frac{\partial f_l(\cdot)}{\partial f_0(\cdot)}
\]
(16)

When we use stochastically generated random feature \( \omega \), we can further approximate \( \xi_j(\cdot) \) using DSG as
\[
  \xi_j(\cdot) \approx l(f_b([x, e]), y_j) \phi_{\omega_0}([x, e]) \phi_{\omega}(\cdot).
\]
(17)
Note that \( \xi_j(\cdot) = \mathbb{E}_{\omega} [\xi_j(\cdot)]. \) This leads to an unbiased estimator of the original functional gradient.

In the \( i \)th iteration, the doubly stochastic (functional) gradient update rule for \( f_0 \) is
\[
  f_0^{i+1}(\cdot) = f_0^{(i)}(\cdot) - \gamma_t \nabla_1 R(f_0^{(i)}, b^{(i)})
\]
\[
  = f_0^{(i)}(\cdot) - \gamma_t \left( \sum_{j=1}^{r} \xi_j(\cdot) + \nu f_0^{(i)}(\cdot) \right)
\]
\[
  = \frac{1}{r} \sum_{i=1}^{r} \sum_{j=1}^{r} \xi_j(\cdot)
\]
(18)
where \( a_j^t = -\gamma_t \prod_{k=1}^{t-1} (1 - \gamma_k \nu), \) and the initial value \( f_0^{(i)}(\cdot) = 0. \) The doubly stochastic update rule for parameter \( b \) is
\[
  b^{i+1} = b^{(i)} - \frac{\gamma_j}{\gamma} [\tilde{z}_1(\cdot), \tilde{z}_2(\cdot), \ldots, \tilde{z}_r(\cdot)]
\]
(19)
Note that \( b^{i+1} \) is updated based on the latest \( f_0^{i+1}. \) These two update rules (18)–(26) make it possible to jointly learn multiple functions with different thresholds.

Meanwhile, if we could manage to compute the stochastic (functional) gradients \( \xi_j(\cdot), \) the update rule for \( f_0 \) becomes
\[
  h^{i+1}(\cdot) = h_0^{(i)}(\cdot) - \gamma_t \nabla_1 R(f_0^{(i)}, b^{(i)'}
\]
\[
  = h_0^{(i)}(\cdot) - \gamma_t \left( \sum_{j=1}^{r} \xi_j(\cdot) + \nu h_0^{(i)}(\cdot) \right)
\]
\[
  = \frac{1}{r} \sum_{i=1}^{r} \sum_{j=1}^{r} \xi_j(\cdot)
\]
(20)
where we have used \( h_0^{i+1} \) instead of \( f_0^{i+1} \) to distinguish from the doubly stochastic (functional) gradient update rule. However, to avoid the expense of kernel computation, our algorithm will use the doubly stochastic update rule (18) instead of (20).

Following the abovementioned DSG update rules (18)–(26), we provide the DSGOR training and prediction procedures in Algorithms 1 and 2 receptively. Especially, following the pseudorandom number generators setting of [24], our random feature generator is initialized by a predefined seed according to iteration. Thus, DSGOR does not need to save the random feature matrix, which makes it more memory friendly. In the \( i \)th iteration, our method will execute the following steps.

1) **Sample Data Instance** (Line 4 in Algorithm 1): Stochastically sample a training sample \((x^t, y^t)\).
2) Sample Random Features (Line 5 in Algorithm 1): Stochastically sample \( o^i \sim \mathcal{P}(o) \) with seed \( i \) and generate random features according to (9).

3) Update Coefficients (Lines 6–13 in Algorithm 1): Evaluate the function value for each subproblem, and update \( f_0 \) and multiple thresholds \( b \), respectively, according to (18) and (26).

For now, whether \( f_0 \) and \( b \) should be updated with the same learning rate is still in question, we will discuss this issue in Section V as we prove the convergence of DSGOR.

**Remark 2:** Another common approach to handle decision function, such as \( f_0(x) + b_j \), is to treat the thresholds \( b \) as a separating term irrelevant of the RKHS of \( f(\cdot) \). Instead, we have reparameterized the RKHS to include the bias. This is because such a reparametrization maintains the strong convexity of the problem, while separating \( b \) from the RKHS does not [52]. Note that although we reparametrize the kernel to help theoretical analysis, we do not cast this OR problem to binary classification, such as what REDSVM [3] does, because it would yield a relatively different optimization problem with an extra term \(|b|^2\). This term limits the range of important thresholds. This is why SVORIM and SOVREX perform better than REDSVM in [13].

### V. Theoretical Guarantees

In this section, we prove that DSGOR converges to the optimal solution with rate \( O(1/t) \) under the proper learning rates for \( f_0 \) and \( b \). Our analysis uses the following assumptions that are standard in doubly stochastic optimization [24], [53].

**Assumption 1:** The objective function is differentiable and strongly convex.

**Assumption 2:** \( l(u, v) \) is \( L \)-Lipschitz continuous in terms of its first argument. We have an upper bound for the derivative of \( l(u, v) \) with respect to its first argument, i.e., \(|l'| < M\).

**Assumption 3:** We have an upper bound for the kernel value, \( k(x, x') \leq \kappa \).

**Assumption 4:** There is an upper bound of random feature norm, i.e., \( |\phi_o(x)\phi_o(x')| \leq \phi \).

Assumption 2 is reasonable so long as \( k_0(x, x') \) is bounded [see (8)]. Assumption 3 is reasonable as long as \(|\phi_{o_0}(x)\phi_{o_0}(x')|\) has an upper bound [see (9)].

Similar to the framework of [24], we prove the convergence of the sequence \(|f_{\beta'}^t((x, x_i)) - f_{\beta'}^*(x, x_i)|\) by decomposing the error

\[
|f_{\beta'}^t((x, x_i)) - f_{\beta'}^*((x, x_i))|^2 \\
\leq 2|f_{\beta'}^t((x, x_i)) - h_{\beta'}^t((x, x_i))|^2 + 2\kappa\|h_{\beta'}^t - f_{\beta'}^*\|_H^2
\]

(21)

where \( f_{\beta} \) and \( h_{\beta} \) can be computed using (18) and (20), respectively.

Different from the original DSG setting, each part of the error of DSGOR contains error induced by the multiple thresholds. Now, we show our analysis for the error of random feature in Lemma 1, the error of random data in Lemma 3, and how multiple thresholds affect the bound. All proofs are provided in the Appendix.

**Lemma 1:** For any \( x \in \mathcal{X} \), any integer \( j \in [1, r] \), we have

\[
\mathbb{E}_{D_{1, o}}[|f_{\beta'}^t((x, e_j)) - h_{\beta'}^t((x, e_j))|^2] \leq B_{1, t+1}^2
\]

where \( B_{1, t+1}^2 := M^2(\sqrt{\kappa} + \sqrt{\theta})^2 \sum_{i=1}^{t} |\alpha_i|^2 \), and \( B_{1, 1} = 0 \).

This lemma depends on the convergence of \(|\alpha_i|^2\) that will be given in Lemma 2.

**Lemma 2:** Suppose that \( \gamma_i = (\theta/i)(1 \leq i \leq t) \) and \( \theta v \in (1, (3/2)) \). We have \(|\alpha_i|^2 \leq \theta/t \) and \( \sum_{i=1}^{t} |\alpha_i|^2 \leq (\theta^2/t) \).

**Lemma 3:** Set \( \gamma_i = (\theta/i)\gamma_i' \), where \( \gamma_i' \) is a random variable that satisfies \( \gamma_i' \leq (\theta/1.5)(1 \leq i \leq t), \theta > 0, \) and \( \theta v \in (1, (3/2)) \). We have

\[
\mathbb{E}_{D_{1, o}}\left[\|f_{\beta'}^t - f_{\beta'}^*\|_{H}^2\right] \leq \frac{Q^2}{t}
\]

(23)

where \( Z_1 = \frac{1}{\sqrt{v}M}L \left( \sqrt{\kappa} + \sqrt{\theta} \right) \), \( Z_2 = \frac{\kappa}{vM}(1 + v\theta)^2 M^2 \), \( Z_3 = \frac{\left((M\theta\phi)/\sqrt{v}\right)}{vM}(1 - 1/2) \), and \( Q_i = \max(|f_{\beta'}^{|i}||_H, t_{(Z_1/2 + Z_3)^2/2} + (Z_1/2 + Z_3)^2/(v\theta - 1)) \).

**Remark 3:** Notice that \( Z_3 \) is the main error term caused by the inclusion of the multiple thresholds. This is another major difference from our analysis to that of Dai et al. [24]. Also, notice that the validity of this lemma relies on setting \( \gamma_i \) and \( \gamma_i' \) differently. This indicates that we need different learning rates for updating \( b \) and \( f_0 \) in order to achieve convergence (see the detailed proof in the Appendix).

According to Lemmas 1 and 3, we give the main result for the convergence of \(|f_{\beta'}^t - f_{\beta'}^*|^2\) in Theorem 2.

**Theorem 2 (Convergence in Expectation):** Set \( \gamma_i = (\theta/i)\gamma_i' \), where \( \gamma_i' \) is a random variable that satisfies \( \gamma_i' \leq (\theta/1.5)(1 \leq i \leq t), \theta > 0, \) and \( \theta v \in (1, (3/2)) \) for Algorithms 1 and 2, and we always have

\[
\mathbb{E}_{D_{1, o}}\left[\|f_{\beta'}^t((x, e_j)) - f_{\beta'}^*(x, e_j))\|^2\right] \leq \frac{2C^2 + 2\kappa Q^2}{t}
\]

(24)

for any integer \( j \in [1, r] \), where \( Q_1 \) is as defined in Lemma 3 and \( C = (\sqrt{\kappa} + \sqrt{\theta} M^2 \).\)

**Remark 4:** Here, \( f_{\beta'}^* \) is the optimal function for solving the kernel OR problem (12) in the new RKHS \( H \). Theorem 2 implies that the evaluated value of \( f_{\beta'}^t \) at any given data \( x \) will converge to that of \( f_{\beta'}^* \). The rate of this convergence is \( O(1/t) \). This is the same as the original DSG even though the OR problem is much more complex.

### VI. Experiments and Analysis

In this section, we compare the generalization performance, training time, and memory cost between the proposed DSGOR algorithm with other state-of-the-art OR solvers.

**A. Experimental Setup**

1) **Design of Experiments:** We compare DSGOR with several existing state-of-the-art kernel OR solvers. Gutiérrez et al. [13] reported that SOVREX and SVORIM are two of the best methods among 16 state-of-the-art OR models based on the results on 41 benchmark data sets, showing competitive generalization performance and time efficiency. However, SOVREX and SVORIM are not scalable even on medium-scale data sets. We also include SVR [54] to further compare the performance on large-scale data sets. SVR treats
the OR problem as a 1-D regression problem and assigns the predicted values to the closest ordinal label [9].

For comparing the generalization performance, we have utilized two evaluation metrics: 1) mean absolute error (MAE), i.e., \( \frac{1}{n} \sum_{i=1}^{n} |h(x_i) - y_i| \) and 2) mean zero-one error (MZE), i.e., \( \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[h(x_i) \neq y_i] \). These two evaluation metrics are widely applied in OR problems [2]. For verifying the generalization performance of our DSGOR, we used fivefold cross validation to compare the MAE and MZE metrics of all competing methods. To evaluate the acceleration and scalability of the competing methods, we record experimental statistics of MAE/MZE versus running time and also running time versus data size. We also compare the maximum memory cost of each method to show that DSGOR is memory friendly.

2) Data Sets: Table III summarizes the eight large-scaled data sets for testing the scalability of DSGOR in our experiments. They are divided into two parts according to the source of data sets, i.e., the discretized regression data sets\(^1\) (the labels of these data sets are discretized into five equal-frequency bins) and the real OR data set [55], [56]. Since SVORIM and SVOREX cannot handle large-scale problems, the max training size of large-scale data sets for verifying accuracy is 75,000. Meanwhile, when we evaluate the scalability of DSGOR, we enlarge the max training size to be \( 10^6 \). To further compare the accuracy, we also included eight widely used small-scaled data sets. The details and results of small-scaled data sets can be found in Appendix E.

3) Implementation Details: We implement DSGOR algorithm by MATLAB. For the sake of efficiency, our DSGOR implementation also uses a minibatch setting, which is similar to DSG. We perform experiments on 36-core two-socket Intel Xeon E5-2696 machine with 48-GB RAM.

For all methods, we use the Gaussian RBF kernel, \( k_0(x, x') = \exp(-\sigma ||x - x'||^2) \). The fivefold cross validation was used to determine the optimal settings (MAEs and MZEs) of the model parameters (the regularization factor \( \nu \) and the Gaussian kernel parameter \( \sigma \)) involved in the problem.

\(^1\)http://archive.ics.uci.edu/ml/data sets.html
formulation. The parameter search process was done on a $7 \times 7$ coarse grid linearly spaced in the region $\{(\log_{10} \nu, \log_{10} \sigma) | -3 \leq \log_{10} \nu \leq 3, -3 \leq \log_{10} \sigma \leq 3\}$ for SVORIM, SOVREX, SVR, and DSGOR. For DSGOR, the step size $\gamma_i$ equals $(1/(1+i*\eta))$, and $\gamma_i'$ equals $(1/(1+(i*1.5*\eta)))$, where $-3 \leq \log_{10} \eta \leq 3$. Besides, the feature block is set to be 2048, and the batch set size is set to 256. Given that DSGOR is essentially an online gradient descent algorithm, we stop the iteration after one pass over the entire data set. All results are the average of ten trials.

B. Experimental Results

Fig. 1 shows the MAE values versus the running time of different algorithms. The results show that DSGOR runs much faster than SOVREX, SVORIM, and SVR. The figure of MZE values versus the running time shows a similar conclusion shown in Fig. 2. In Figs. 1 and 2, we could only plot the performance of the earlier training stage of SOVREX, SVORIM, and SVR since our method converges significantly faster. Fig. 3 confirms the high efficiency of DSGOR even on data sets with one million samples, as shown in Fig. 3(a).

TABLE III

| LARGE-SCALED DATA SETS USED IN THE EXPERIMENTS. (TRAINING DENOTES THE TRAINING SIZE USED FOR ACCURACY AND MEMORY COST COMPARISON, I.E., FIGS. 1, 2, AND 4 AND TABLE IV, AND TEST SIZE IS 20% OF TRAINING SIZE. D DENOTES THE INPUT DIMENSION, AND R DENOTES THE NUMBER OF LABEL CATEGORIES) |
|------------------|-----------------|------------|-------------|-------------|
| DISCRETIZED REGRESSION DATASETS |
| DATASET | TRAINING SIZE | SIZE | D | R | CLASS DISTRIBUTION |
| ETHYLENE (CO) | 75,000 | 4,208,262 | 16 | 5 | 15,000 PER CLASS |
| (3D) NETWORK | 75,000 | 434,874 | 3 | 5 | 15,000 PER CLASS |
| FEATURE (FT) | 75,000 | 107888 | 482 | 5 | 15,000 PER CLASS |
| REAL ORDINAL REGRESSION DATASETS |
| DATASET | TRAINING SIZE | SIZE | D | R | CLASS DISTRIBUTION |
| MQ2007 (M7) | 50,000 | 69,623 | 46 | 3 | (37,079 10,146 2,775) |
| MQ2008 (M8) | 10,000 | 25,211 | 46 | 3 | (8,072 1,315 613) |
| MSLR (MR) | 75,000 | 723,412 | 136 | 5 | (39,185 2,411 9,857 1,312 534) |
| HP2003 (H3) | 75,000 | 14,760 | 64 | 5 | (15,000 15,000 15,000 15,000 15,000) |
| NP2003 (N3) | 75,000 | 14,760 | 64 | 5 | (15,000 15,000 15,000 15,000 15,000) |

Fig. 4 shows that DSGOR has the lowest memory cost. Notice that this is true given that the other three competing methods have applied their own cache strategies.

Fig. 4 shows that DSGOR has the lowest memory cost. Notice that this is true given that the other three competing methods have applied their own cache strategies.

Table IV shows the accuracies (MAEs and MZEs) of SVORIM, SOVREX, SVR, and DSGOR. The results were tested at the optimal solution for SVORIM, SOVREX, and SVR. Since DSGOR only obtains an approximate solution,
it does not always perform best. From the figure, we could tell that DSGOR achieves similar generalization performance with SVORIM and SVOREX methods both manually discretized regression data sets and real OR data sets. The figure of MZE values versus the running time can be found in the Appendix, which supports a similar conclusion as that of Table IV. These experimental results confirm that DSGOR has a good convergence rate.

VII. CONCLUSION

In this article, we provide a novel extension to DSG for kernel OR problems where decision functions with various thresholds must be optimized simultaneously. First, we construct a new kernel and its RKHS $\mathcal{H}$ that incorporates both the predecision function $f_0$ and the final decision functions $F_j$'s with differed thresholds. Based on this new theoretic setup, we make it possible to use functional gradient analysis, and in turn, DSG methods, on kernel OR problem. The proposed DSGOR method achieves significant speedup over traditional kernel OR methods, leveraging the advantages of stochastic optimization and random feature approximation. With careful theoretic study, we have successfully discovered two different sets of learning rates for the predecision function $f_0$ and the bias to guarantee that DSGOR converges with the rate of $O(1/\ell)$. Experimental results on different large-scale data sets demonstrate that our DSGOR has a comparative performance and smaller memory requirement and runs much faster than the state-of-the-art OR solvers.

APPENDIX A

PROOF OF LEMMA 1

We denote $W_i((x, x_i)) = a_i^l(\zeta^i((x, x_i)) - \zeta^i((x, x_i)))$. Based on the definition and assumptions in the body text, $W_i((x, x_i))$ is bounded as follows $W_i((x, x_i)) \leq |a_i^l|(|\zeta^i((x, x_i))| + |\zeta^i((x, x_i))|) = M(\sqrt{x_i} + \sqrt{-x_i})|a_i^l|$, and we obtain Lemma 1.

APPENDIX B

PROOF OF LEMMA 2

Proof: It is obvious that $|a_i^l| \leq (\theta/\ell)$. Then, we have

$$|a_i^l| = \frac{d_0}{d_i^{l+1}} \frac{d_i^{l+1}}{d_0^{l+1}} (1 - \lambda \gamma_i^{l+1})$$

$$= i + 1 - \frac{\lambda \theta}{i + 1} \cdot |a_i^{l+1}| = \left| \frac{i + 1 - \lambda \theta}{i} \cdot |a_i^{l+1}| \right|.$$

When $\lambda \theta \in (1, (3/2)) \forall i \geq 1$, we have $i + 1 - \lambda \theta$, so $|a_i^l| < |a_i^{l+1}| \leq (\theta/\ell)$ and $\sum_{i=1}^r |a_i^{l+1}| \leq (\theta^2/\ell)$. Thus, we get $\sum_{i=1}^r |a_i^{l+1}|^2 \leq (\theta^2/\ell)$. Therefore, we obtain Lemma 2.

APPENDIX C

PROOF OF LEMMA 3

Proof: Denote

$$g_t = \frac{1}{r} \sum_{j=1}^r \zeta_j + v h_{b_t}$$

$$= \frac{1}{r} \sum_{j=1}^r [l'(f_b([x^l, e_j]), y')] k([x, x_e], \cdot) + v h_{b_t}$$

$$\tilde{g}_t = \frac{1}{r} \sum_{j=1}^r \zeta_j + v h_{b_t}$$

$$= \frac{1}{r} \sum_{j=1}^r [l'(f_b([x^l, e_j]), y')] k([x, x_e], \cdot) + v h_{b_t}$$

so $h_{b_t}^{l+1} = h_{b_t}^l - \gamma_t \tilde{g}_t$.

Denote $A_t = ||h_{b_t}^l - f_{\phi^*}||_{\mathcal{H}}^2$. We have

$$A_{t+1} = \|h_{b_t}^{l+1} - f_{\phi^*}||^2_{\mathcal{H}} = \|h_{b_t}^{l+1} - h_{b_t}^l + h_{b_t}^l - f_{\phi^*}||^2_{\mathcal{H}}$$

$$\leq \|h_{b_t}^{l+1} - h_{b_t}^l||^2_{\mathcal{H}} + \|h_{b_t}^l - f_{\phi^*}||^2_{\mathcal{H}}$$

$$+ 2\sqrt{\|h_{b_t}^{l+1} - h_{b_t}^l||^2_{\mathcal{H}} \cdot \|h_{b_t}^l - f_{\phi^*}||^2_{\mathcal{H}}}$$

in which

$$\|h_{b_t}^{l+1} - h_{b_t}^l||^2_{\mathcal{H}} = ||b_{l+1} - b_l||^2 = \sum_{j=1}^r |\zeta_j^{l+1} - \zeta_j^l||^2_{\mathcal{H}}$$

$$\leq \frac{\gamma_t^2}{r} M^2 \phi^2$$

where we have used the update rule of $b$ as follows:

$$b_{t+1} = b_t - \frac{\gamma_t^l}{r} \zeta_j^l \zeta_j^l$$.

---

**TABLE IV**

ACCURACIES (MAEs AND MZEs) OF SVR, SVORIM, SVOREX, AND DSGOR ON LARGE-SCALED DATA SETS.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>SVR</th>
<th>SVORIM</th>
<th>SVOREX</th>
<th>DSGOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MZE</td>
<td>MAE</td>
<td>MZE</td>
</tr>
<tr>
<td></td>
<td>MAE</td>
<td>MZE</td>
<td>MAE</td>
<td>MZE</td>
</tr>
<tr>
<td></td>
<td>MAE</td>
<td>MZE</td>
<td>MAE</td>
<td>MZE</td>
</tr>
<tr>
<td></td>
<td>MAE</td>
<td>MZE</td>
<td>MAE</td>
<td>MZE</td>
</tr>
<tr>
<td></td>
<td>MAE</td>
<td>MZE</td>
<td>MAE</td>
<td>MZE</td>
</tr>
</tbody>
</table>

---

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 05, 2021 at 02:48:39 UTC from IEEE Xplore. Restrictions apply.
Meanwhile, we have
\[
\begin{align*}
\| h^{t+1}_\nu - f^*_{\nu} \|_{\| \cdot \|_\nu} & = \| h^{t+1}_\nu - f^*_{\nu} - (\gamma^t g_t - f^*_{\nu}) \|_{\| \cdot \|_\nu} \\
& = A_t + \gamma^t \| g_t \|_{\| \cdot \|_\nu}^2 - 2 \gamma^t (h^{t+1}_\nu - f^*_{\nu}, g_t)_{\| \cdot \|_\nu} \\
& = A_t + \gamma^t \| g_t \|_{\| \cdot \|_\nu}^2 - 2 \gamma^t (h^{t+1}_\nu - f^*_{\nu}, \tilde{g}_t)_{\| \cdot \|_\nu} \\
& \quad + 2 \gamma^t (h^{t+1}_\nu - f^*_{\nu}, \tilde{g}_t - \hat{g}_t)_{\| \cdot \|_\nu} + 2 \gamma^t (h^{t+1}_\nu - f^*_{\nu}, \tilde{g}_i - g_t)_{\| \cdot \|_\nu}.
\end{align*}
\]

With the strong convexity of the objective function, we also have
\[
\begin{align*}
\| h^{t+1}_\nu - f^*_{\nu} \|_{\| \cdot \|_\nu} & \geq \frac{V}{2} \| h^{t+1}_\nu - f^*_{\nu} \|_{\| \cdot \|_\nu} \\
& = \frac{V}{2} A_t, 
\end{align*}
\]
and
\[
\begin{align*}
\| h^{t+1}_\nu - f^*_{\nu} \|_{\| \cdot \|_\nu} & \leq A_t + \gamma^t \| g_t \|_{\| \cdot \|_\nu}^2 - 2 \gamma^t (h^{t+1}_\nu - f^*_{\nu}, \tilde{g}_t)_{\| \cdot \|_\nu} \\
& \quad + 2 \gamma^t (h^{t+1}_\nu - f^*_{\nu}, \tilde{g}_i - \hat{g}_t)_{\| \cdot \|_\nu} \\
& = (1 - \nu \gamma^t) A_t + \gamma^t \mathcal{M}_t + 2 \gamma^t [\mathcal{N}_t + \mathcal{R}_t],
\end{align*}
\]
where
\[
\begin{align*}
\mathcal{M}_t & = \| g_t \|_{\| \cdot \|_\nu}^2, \\
\mathcal{N}_t & = \| h^{t+1}_\nu - f^*_{\nu} - \tilde{g}_t \|_{\| \cdot \|_\nu} \\
\mathcal{R}_t & = \| h^{t+1}_\nu - f^*_{\nu} - \tilde{g}_i - g_t \|_{\| \cdot \|_\nu}.
\end{align*}
\]
Similar to [24], we could prove the following:
\[
\begin{align*}
\mathcal{M}_t & \leq \kappa M^2 (1 + \nu c_t)^2, \\
\mathcal{E}_{D_t, o_{\nu}}[\mathcal{N}_t] & = 0 \\
\mathcal{E}_{D_t, o_{\nu}}[\mathcal{R}_t] & \leq \kappa^{1/2} LB_{1, t} / \sqrt{\mathcal{E}_{D_t, o_{\nu}, i_{\nu}}[A_t]},
\end{align*}
\]
Putting everything we have so far, i.e., (27)–(29) back into (25), denote \( e_{t+1} = \mathcal{E}_{D_t, o_{\nu}}[A_t] \), and we get
\[
\begin{align*}
e_{t+1} & \leq \frac{1}{r} \gamma^t \frac{M^2 \phi^2}{2} + (1 - \nu \gamma^t) e_t + \gamma^t M^2 \kappa (1 + \nu c_t)^2 \\
& \quad + \gamma \kappa^{1/2} LB_{1, t} / \sqrt{r} e_t + 2 \sqrt{P} \tag{30}
\end{align*}
\]
where \( P = (\gamma^t / r) M^2 \phi^2 (1 - \nu \gamma^t) e_t + \gamma^t M^2 \kappa (1 + \nu c_t)^2 + \gamma \kappa^{1/2} LB_{1, t} / \sqrt{r} e_t \). Applying \( \gamma_t = (\theta / t), \gamma'_\nu = (\theta / t^{1.5}), c_t \leq 0, \) and \( B^2_{t+1} \leq M^2 (\sqrt{\kappa} + \sqrt{\phi})^2 (\theta^2 / t) \), we get
\[
\begin{align*}
e_{t+1} & \leq \left[ (1 - \frac{\eta}{t}) e_t + \frac{Z_1}{t} \sqrt{e_t} + \frac{Z_2}{t^2} \right] \frac{Z_3}{t^3} \\
& \quad + 2 \frac{Z_3}{t^{1.5}} \left( \frac{1 - \frac{\eta}{t}}{t} \right) e_t + \frac{Z_1}{t} \sqrt{e_t} + \frac{Z_2}{t^2} \tag{31}
\end{align*}
\]
where \( \eta = \theta v \in (1, (3/2)), Z_1 = \kappa^{1/2} LM (\sqrt{\kappa} + \sqrt{\phi}) \phi^2, \)
\[
\begin{align*}
Z_2 = \kappa M^2 (1 + v \phi^2) \phi^2, \text{ and } Z_3 = ((M^2 \phi^2 \phi^3) / r).
\end{align*}
\]
Finally, we complete our proof by invoking the following lemma.
\[\square\]
or equivalently

\[ \frac{W_t}{t^2} \geq 2 \frac{Z_3}{t^{3/2}} \sqrt{\frac{Q}{t+1}} - \frac{Z_3^2}{t^3}, \quad \text{and} \quad \frac{Q}{t+1} - \frac{W_t}{t^2} \geq 0 \]

whose sufficient condition is

\[ \frac{W_t}{t^2} \geq 2 \frac{Z_3^2}{t^2} \sqrt{\frac{Q}{t+1}} - \frac{Z_3^2}{t^3}, \quad \text{and} \quad \frac{Q}{t+1} - \frac{W_t}{t^2} \geq 0 \]

whose sufficient condition is

\[ W_t \geq 2 \sqrt{Z_3^2 Q}, \quad \text{and} \quad \frac{Q}{t+1} - \frac{W_t}{t^2} \geq 0 \]

or

\[
\begin{cases}
(Q-1)Q - (Z_1 + 2Z_3)\sqrt{Q} - Z_2 \geq 0 \\
\frac{Q}{t+1} - \frac{W_t}{t^2} \geq 0
\end{cases}
\]

whose sufficient conditions are

\[
Q \geq \left[ \frac{\sqrt{(\eta - 1)Z_2 + (Z_1^2 + 2Z_3)^2} + Z_1 - \eta}{\eta - 1} \right]^2
\]

\[
\left( \frac{t^2}{t+1} + 1 - \eta \right)Q + Z_1\sqrt{Q} + Z_2 \geq 0.
\]

Since \( \eta \in (1, (3/2)) \), \( t \geq 1 \), the lower inequality will always hold.

Thus, the induction holds true when \( Q \leq \left( (((\eta - 1)Z_2 + (Z_1^2 + 2Z_3)^2)^{1/2} + (Z_1/2 + Z_3)/\eta - 1)^2 \right) \).

### APPENDIX D

#### PROOF OF THEOREM 1

When \( k_0 \) is a PD kernel, we denote \( K_0 \) as the kernel matrix of \( k_0(x, x') \), and then, \( K_0 \) is a symmetric positive definite matrix according to the definition of PD kernel with \( k_0(x, x') = k_0(x', x) \). Next, we prove that the new kernel \( k \) has the symmetric property and positive definite property as follows.

1) **Symmetric Property:** \( k(\tilde{x}, \tilde{x}') = k_0(x, x') + \langle x, x' \rangle = k_0(x', x) = k(x', \tilde{x}) \).

2) **Positive Definite Property:** We denote a matrix \( J \), the element is \( \langle x, x' \rangle \) that can only be 0 or 1 and all diagonal elements are all 1, and it easy to see that all order principal minor determinants of \( J \) are greater than 0, so \( J \) is positive definite. Because both \( K_0 \) and \( J \) are positive definite, the \( k \)-s kernel matrix \( K = K_0 + J \) is positive definite.

Thus, \( k(\tilde{x}, \tilde{x}') \) is a PD kernel.

### APPENDIX E

#### EXPERIMENTS RESULTS ON SMALL SCALED DATA SETS

To further compare the accuracy, we show the experimental results of eight small-scaled data sets that are widely used in OR studies in Table V. Table VI shows that our DSGOR can achieve similar performance on standard OR data sets although DSGOR is an approximation method. When the data size is

\[ \text{http://www.gatsby.ucl.ac.uk/ chuwei/ordinalregression.html} \]
small, Fig. 5 shows that DSGOR runs faster than other kernel OR methods in most of the cases. Note that, when the data size is small, the computation of the kernel matrix is not so time-consuming. Thus, the acceleration of DSGOR is not obvious on small-scaled data sets.

REFERENCES


Xiang Geng received the B.S. and M.S. degrees in computer science and technology from the Nanjing University of Information Science and Technology (NUIST), Nanjing, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with Nanjing University (NJU), Nanjing.

His research interests include machine learning and natural language processing.

Xiang Li received the B.S. and M.S. degrees in computer science from the National University of Defense Technology (NUDT), Changsha, China, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from the University of Western Ontario, London, ON, Canada, in 2017.

He currently holds a post-doctoral position at the University of Western Ontario. His research interests include machine learning and data mining.

Wanli Shi received the B.E. degree in electronic science and technology from the Binjiang College, Nanjing University of Information Science and Technology, Nanjing, China, in 2017, where he is currently pursuing the master's degree.

His main research interests include machine learning and data mining.

Guansheng Zheng received the B.E. and M.E. degrees from the College of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China, in 1996 and 2003, respectively.

He is currently an Associate Professor with the School of Computer and Software, Nanjing University of Information Science and Technology. His research interests include machine learning and data mining.

Cheng Deng (Student Member, IEEE) received the B.E., M.S., and Ph.D. degrees in signal and information processing from Xidian University, Xi'an, China, in 2001, 2005, and 2009, respectively.

He is currently a Full Professor with the School of Electronic Engineering, Xidian University. He is the author or a coauthor of more than 50 scientific articles at top venues, including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), the IEEE TRANSACTIONS ON MULTIMEDIA (TMM), the IEEE TRANSACTIONS ON CYBERNETICS (TCYB), the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS (TSMC), the IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), the International Conference on Computer Vision (ICCV), the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), the International Joint Conferences on Artificial Intelligence (IJCAI), and the Association for the Advancement of Artificial Intelligence (AAAI). His research interests include computer vision, multimedia processing and analysis, and information hiding.

Heng Huang (Member, IEEE) received the B.S. and M.S. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1997 and 2001, respectively, and the Ph.D. degree in computer science from Dartmouth College, Hanover, NH, USA, in 2006.

He is currently a John A. Jarenko Endowed Professor of computer engineering with the Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA, USA. He is also a Consulting Researcher with JD Finance American Corporation, Mountain View, CA, USA. His research interests include machine learning, data mining, computer vision, and biomedical data science.