entropy

MDPI

Article

Distributed Support Vector Ordinal Regression over Networks

Huan Liu, Jiankai Tu and Chunguang Li *

College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China
* Correspondence: cgli@zju.edu.cn

Abstract: Ordinal regression methods are widely used to predict the ordered labels of data, among which support vector ordinal regression (SVOR) methods are popular because of their good generalization. In many realistic circumstances, data are collected by a distributed network. In order to protect privacy or due to some practical constraints, data cannot be transmitted to a center for processing. However, as far as we know, existing SVOR methods are all centralized. In the above situations, centralized methods are inapplicable, and distributed methods are more suitable choices. In this paper, we propose a distributed SVOR (dSVOR) algorithm. First, we formulate a constrained optimization problem for SVOR in distributed circumstances. Since there are some difficulties in solving the problem with classical methods, we used the random approximation method and the hinge loss function to transform the problem into a convex optimization problem with constraints. Then, we propose subgradient-based algorithm dSVOR to solve it. To illustrate the effectiveness, we theoretically analyze the consensus and convergence of the proposed method, and conduct experiments on both synthetic data and a real-world example. The experimental results show that the proposed dSVOR could achieve close performance to that of the corresponding centralized method, which needs all the data to be collected together.

Keywords: ordinal regression; support vector machine; support vector ordinal regression; distributed algorithm; subgradient method

Citation: Liu, H.; Tu, J.; Li, C. Distributed Support Vector Ordinal Regression over Networks. *Entropy* **2022**, *24*, 1567. https://doi.org/10.3390/e24111567

Academic Editors: Minyu Feng, Liang-Jian Deng and Feng Chen

Received: 12 September 2022
Accepted: 28 October 2022
Published: 31 October 2022

1. Introduction

Many real-world data labels have natural orders that are usually called ordinal labels. For example, fault severity in industrial processes is usually divided into {*harmless*, *slight*, *medium*, *severe*}. Ordinal regression, which aims at predicting ordinal labels for given patterns, has attracted a great deal of research in many fields, such as disease severity assessment [1], satisfaction evaluation [2], wind-speed prediction [3], age estimation [4], credit-rating prediction [5], and fault severity diagnosis [6]. Although classical classification and regression methods can be applied to the ordinal regression problem [7,8], they require additional prior information about the distances between labels. Otherwise, they often perform unsatisfactorily since they cannot fully use ordering information [9,10].

To tackle the aforementioned problems of classical classification and regression methods, many ordinal regression methods were proposed [10]. Among them, the most popular type of approaches are threshold models, which assume that a continuous latent variable underlies the ordinal response [10]. In threshold models, the order of the labels is represented by a set of ordered thresholds. These ordered thresholds define a series of intervals, and the data label depends on the interval the corresponding latent variable falls into. Among the threshold models, support vector ordinal regression (SVOR) [11,12] is widely used because of good generalization performance. A representative work is the support vector ordinal regression with implicit constraints (SVORIM) proposed in [11,12]. This determines each threshold by taking all the samples into consideration, where the threshold inequality constraints can be satisfied without explicit constraints.

Most of the existing ordinal regression methods have been developed in a centralized framework. However, in practice, data used for ordinal regression may be distributed in

a network [13]. Each node of the network collects and stores part of the data, and it is not enough for a single node to train a model with good performance. For instance, in industrial processes, sensors are often used in factories to monitor the operating status of equipment and diagnose fault severity. Due to the rarity of faults, a single sensor can only collect very few data, and the faults encountered by each factory may also be different. To train a proper model, we need to use as many data as possible. However, in some realistic scenarios, it is difficult for data to be transmitted to a central node for various reasons [13]. For example, factories may not want to leak data regarding their equipment in order to protect privacy. Moreover, if the data are collected by image sensors or video sensors, it may be difficult for a single machine to store and process such a large amount of data. In such situations, centralized methods are inapplicable, and distributed methods are more suitable choices.

In this paper, we propose a distributed support vector ordinal regression algorithm based on the SVORIM method to deal with more complex nonlinear problems in distributed ordinal regression. First, we formulate a constrained optimization problem for SVORIM in the distributed scenarios. Classical methods usually solve the problem by transforming it into the dual problem. In distributed circumstances where the original data cannot be transmitted to others, it is difficult for classical methods to calculate the kernel function values and optimize the dual variables because they require data from different nodes. Thus, we adopted a random approximation method and the hinge loss function to transform the optimization problem to overcome the above difficulties. Increasing the number of random approximation dimensions can improve the approximation accuracy, but brings redundancy. In order to find an appropriate number of approximation dimensions, we further added a sparse regularization term of the approximation dimension number to the objective function. Through the above steps, we transformed the original problem into a convex optimization problem with consensus constraints. Then, to solve the problem, we propose a subgradient-based algorithm called distributed SVOR (dSVOR) where each node only uses its own data and the parameter estimates exchanged from its neighbors. To verify the effectiveness of dSVOR, we theoretically analyze its consensus and convergence, and conducted some experiments on synthetic data and a real-world example. The experimental results show that the proposed distributed algorithm under additional constraints could achieve close performance to that of the corresponding centralized method, which needs all the data to be collected to a central node.

The main contributions of this paper are summarized as follows.

1. Existing work on distributed ordinal regression [14] uses a linear model; therefore, it cannot deal with the problems of linearly inseparable data. We extended the SVOR method to distributed scenarios to solve distributed ordinal regression problems with linearly inseparable data.
2. We developed a decentralized implementation of SVOR, and propose a dSVOR algorithm. In the proposed algorithm, the kernel feature map is approximated by random feature maps to avoid transmitting the original data, and sparse regularization is added to avoid excessively high approximation dimensions.
3. The consensus and convergence of the proposed algorithm are theoretically analyzed.

The rest of this paper is organized as follows. In Section 2, we introduce related works. The ordinal regression problem and the SVORIM method are introduced in Section 3 as preliminary knowledge. In Section 4, we formulate the distributed support vector ordinal regression problem, propose the dSVOR algorithm, and perform theoretical analysis of the proposed algorithm. Experiments were conducted to evaluate the effectiveness of the proposed algorithm and they are presented in Section 5. Lastly, in Section 6, we draw some conclusions.

2. Related Works

Ordinal Regression Methods. Many ordinal regression methods have been proposed to solve ordinal regression problems. The ordered logit model [15,16] makes assumptions

about the distribution of the prediction error of the latent variable, and uses the cumulative distribution function to build the label cumulative probability function. The support vector ordinal regression (SVOR) [11,12] maximizes margins between two adjacent labels. Variants of SVOR with nonparallel hyperplanes were discussed in [17,18]. There are also ordinal regression methods that solve ordinal regression problems by solving a series of binary classification subproblems. In [4,19], extended labels were extracted from the original ordinal labels to learn a binary classifier (such as support vector machine [19] or logistic regression [4]); then, a ranking rule was constructed from the binary classifier to predict ordinal labels. In [20], the authors used the stick-breaking process to construct a series of binary classification subproblems to guarantee that the cumulative probabilities were monotonically decreasing. However, the above ordinal regression methods are all centralized and are infeasible in distributed scenarios.

Distributed methods. Distributed methods were extensively studied in many fields, such as distributed estimation [21,22], distributed optimization [23,24], distributed clustering [25], distributed Kalman filter [26], and distributed anomaly detection [27]. However, as far as we know, there are few works investigating distributed ordinal regression [14]. In [14], the authors proposed a distributed generalized ordered logit model, which is a linear model and therefore cannot handle complex problems.

3. Preliminaries

3.1. Ordinal Regression Problem

The classification problem aims at classifying the K-dimensional input vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^K$ into one of Q discrete categories $y \in \mathcal{Y} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_Q\}$. The ordinal regression problem is a type of classification problem in which the data labels have a natural order $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \cdots \prec \mathcal{C}_Q$, where $\prec$ is an order relation [10]. The purpose of ordinal regression is to find a mapping function $f : \mathcal{X} \to \mathcal{Y}$ to predict the ordinal labels for new patterns given a training set of N samples $D = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$.

3.2. Support Vector Ordinal Regression with Implicit Constraints

Let $\boldsymbol{\phi}(\boldsymbol{x})$ denote the feature vector in a high-dimensional reproducing kernel Hilbert space (RKHS) of input vector $\boldsymbol{x}$. The inner product in the RKHS is defined by the reproducing kernel function: $K(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x'})$.

Support vector machines construct a discriminant hyperplane in the RKHS by maximizing the distance between support vectors and the discriminant hyperplane. The discriminant hyperplane is defined by an optimal direction $\boldsymbol{w}$ and a single optimal threshold b. It divides the feature space into two regions for two classes.

The support vector ordinal regression constructs $Q-1$ parallel discriminant hyperplanes for Q ordinal labels where these hyperplanes are defined by optimal direction $\boldsymbol{w}$ and $Q-1$ thresholds $\{b_q\}_{q=1,\ldots,Q-1}$. The ordinal information in the labels is represented by threshold inequalities $b_1 \leq b_2 \leq \cdots \leq b_{Q-1}$. For convenience, vector $\boldsymbol{b} = [b_1\ b_1 \cdots b_{Q-1}]^T$ was used to denote these thresholds.

In [11,12], the SVORIM method determined a threshold b_q by utilizing the samples of all the labels. For threshold b_q, each sample belonging to $\mathcal{C}_p, \forall p \leq q$ should have a function value less than $b_q - 1$; otherwise, $\xi_{pi}^q = \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i^p) - (b_q - 1)$ is the empirical error of $\boldsymbol{x}_i^p$ for b_q. Similarly, each sample belonging to $\mathcal{C}_p, \forall p > q$ should have a function value greater than $b_q + 1$; otherwise, $\xi_{pi}^{*q} = (b_q + 1) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i^p)$ is the empirical error of $\boldsymbol{x}_i^p$ for b_q.

As proved in [11,12], this approach has the property that the threshold inequalities can be automatically satisfied after convergence without explicitly including the corresponding constraints. This method is called support vector ordinal regression with implicit constraints and is formulated as follows:

$$\begin{aligned}\min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\xi},\boldsymbol{\xi}^*} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{q=1}^{Q-1}\sum_{p=1}^{q}\sum_{i=1}^{N^p}\xi_{pi}^q + C\sum_{q=1}^{Q-1}\sum_{p=q+1}^{Q}\sum_{i=1}^{N^p}\xi_{pi}^{*q} \\ \text{s.t. } & \boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^p) - b_q \le -1 + \xi_{pi}^q, \quad \xi_{pi}^q \ge 0, \quad \forall i, q \text{ and } p = 1,\ldots,q \\ & \boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^p) - b_q \ge +1 - \xi_{pi}^{*q}, \quad \xi_{pi}^{*q} \ge 0, \quad \forall i, q \text{ and } p = q+1,\ldots,Q,\end{aligned} \tag{1}$$

where C is a predefined positive constant. The above problem can be solved by solving the dual problem, which can be derived with standard Lagrangian techniques. Let $\beta_{pi}^q \ge 0$, $\gamma_{pi}^q \ge 0, \beta_{pi}^{*q} \ge 0$, and $\gamma_{pi}^{*q} \ge 0$ be the Lagrangian multipliers for the constraints in the above equation. The dual problem is the following maximization problem [11,12].

$$\begin{aligned}\max_{\boldsymbol{\beta},\boldsymbol{\beta}^*} \ & -\frac{1}{2}\sum_{p,i}\sum_{p',i'}\Big(\sum_{q=1}^{p-1}\beta_{pi}^{*q} - \sum_{q=p}^{Q-1}\beta_{pi}^{q}\Big)\Big(\sum_{q=1}^{p'-1}\beta_{p'i'}^{*q} - \sum_{q=p'}^{Q-1}\beta_{p'i'}^{q}\Big)K(\boldsymbol{x}_i^p, \boldsymbol{x}_{i'}^{p'}) \\ & + \sum_{p,i}\Big(\sum_{q=1}^{p-1}\beta_{pi}^{*q} + \sum_{q=p}^{Q-1}\beta_{pi}^{q}\Big) \\ \text{s.t. } & \sum_{p=1}^{q}\sum_{i=1}^{N^p}\beta_{pi}^q = \sum_{p=q+1}^{Q}\sum_{i=1}^{N^p}\beta_{pi}^{*q}, \quad \forall q \\ & 0 \le \beta_{pi}^q \le C, \quad \forall i, q \text{ and } p \le q \\ & 0 \le \beta_{pi}^{*q} \le C, \quad \forall i, q \text{ and } p > q.\end{aligned} \tag{2}$$

For a new pattern $\boldsymbol{x}$, SVORIM calculates the function value $\boldsymbol{w}\cdot\phi(\boldsymbol{x})$ and then decides its category according to the interval the function value falls into, where the intervals are defined by thresholds $\{b_q\}_{q=1,\ldots,Q-1}$.

4. Distributed Support Vector Ordinal Regression Algorithm

4.1. Network and Data Model

In this paper, we consider a network consisting of M nodes. We could use a graph $\mathcal{G} = (\mathcal{M}, \mathcal{E})$ to represent this network. It consisted of a set of nodes $\mathcal{M} = \{1, 2, \ldots, M\}$ and a set of edges $\mathcal{E}$. Each edge $(m, n) \in \mathcal{E}$ connected a pair of distinct nodes. We used $\mathcal{N}_m = \{n|(m, n) \in \mathcal{E}\}$ to represent the set of neighbors of node $m \in \mathcal{M}$.

Data used for ordinal regression are distributedly collected and stored by the M nodes of this network. The i-th sample of node m is represented as $(\boldsymbol{x}_{m,i}, y_{m,i})$, where $\boldsymbol{x}_{m,i} \in \mathcal{X}$ and $y_{m,i} \in \mathcal{Y}$. More specifically, at node m, the total number of samples is N_m, the number of samples that belong to $\mathcal{C}_q$ is N_m^q, and the i-th sample of $\mathcal{C}_q$ is denoted as $(\boldsymbol{x}_{m,i}^q, y_{m,i}^q)$.

Figure 1 shows a schematic of a distributed network. In distributed networks, due to limited storage, computation and communication resources and the need for privacy protection, node m can only transmit some parameters θ_m instead of the original data to its neighbor nodes in $\mathcal{N}_m$, and perform local computation using only its own data $\{(\boldsymbol{x}_{m,i}, y_{m,i})\}_{1\le i\le N_m}$ and the parameters exchanged from its neighbors. Each node should eventually obtain a model consensus with that obtained by other nodes, and the performance of the model should be close to that of the model trained using all the data.

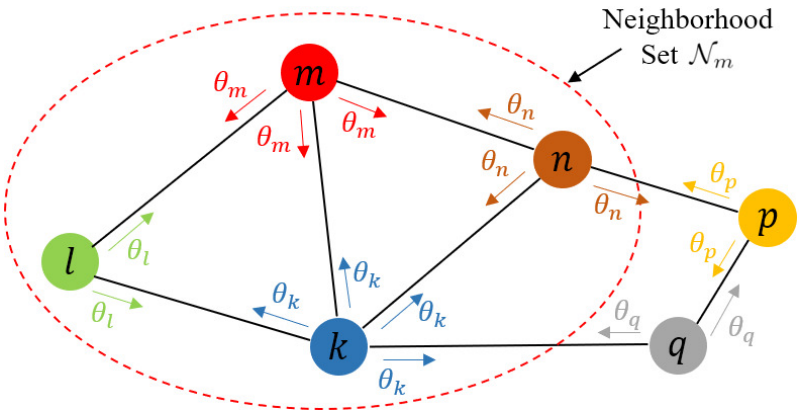


Figure 1. Schematic of a distributed network. Node m only transmits its parameters θ_m with nodes in $\mathcal{N}_m$.

4.2. Problem Formulation

In centralized SVOR, the objective is to find an optimal direction $\boldsymbol{w}$ and a vector $\boldsymbol{b}$. If the data from all the nodes of the distributed network can be collected together, then parameters $\theta = \{\boldsymbol{w}, \boldsymbol{b}\}$ can be obtained by solving Problem (1).

In distributed situations, data are not allowed to be transmitted to a central node. Each node can only use its own data and some parameters from its neighbors. In this case, each node m has a local estimate θ_m of θ. With a connected network, we imposed constraints $\theta_m = \theta_n, \forall (m,n) \in \mathcal{E}$ to ensure the consensus of $\{\theta_m\}_{m=1,\ldots,M}$. Then, the corresponding optimization problem in distributed scenarios can be written as follows:

$$\begin{aligned}
\min \quad & \frac{1}{2}\sum_{m=1}^{M}\|\boldsymbol{w}_m\|^2 + C\sum_{m=1}^{M}\sum_{q=1}^{Q-1}\sum_{p=1}^{q}\sum_{i=1}^{N_m^p}\xi_{m,pi}^{q} + C\sum_{m=1}^{M}\sum_{q=1}^{Q-1}\sum_{p=q+1}^{Q}\sum_{i=1}^{N_m^p}\xi_{m,pi}^{*q} \\
\text{s.t.} \quad & \boldsymbol{w}_m \cdot \phi(\boldsymbol{x}_{m,i}^{p}) - b_{m,q} \le -1 + \xi_{m,pi}^{q}, \quad \xi_{m,pi}^{q} \ge 0, \\
& \forall m, i, q \text{ and } p = 1, \ldots, q \\
& \boldsymbol{w}_m \cdot \phi(\boldsymbol{x}_{m,i}^{p}) - b_{m,q} \ge +1 - \xi_{m,pi}^{*q}, \quad \xi_{m,pi}^{*q} \ge 0, \\
& \forall m, i, q \text{ and } p = q+1, \ldots, Q \\
& \boldsymbol{w}_m = \boldsymbol{w}_n, \boldsymbol{b}_m = \boldsymbol{b}_n, \forall (m,n) \in \mathcal{E},
\end{aligned} \tag{3}$$

where $\xi_{m,pi}^{q}$ is the empirical error of $\boldsymbol{x}_{m,i}^{p}$ for $b_{m,q}$ when $p = 1, \ldots, q$ and $\xi_{m,pi}^{*q}$ is the empirical error of $\boldsymbol{x}_{m,i}^{p}$ for $b_{m,q}$ when $p = q+1, \ldots, Q$. With the help of the consensus constraints, this problem is equivalent to Problem (1).

4.3. Problem Transformation

In classical solutions, a primal problem is solved by solving the corresponding dual problem. Applying such methods to Distributed Problem (3) is confronted with two major difficulties:

1. For nonlinear kernel functions, the dimension of the RKHS is unknown, and we can only calculate the inner product of $\phi(\boldsymbol{x}_{m,i})$ and $\phi(\boldsymbol{x}_{n,j})$ rather than them. Because the data are distributed in various nodes of the network, the kernel function $K(\boldsymbol{x}_{m,i}, \boldsymbol{x}_{n,j})$ requiring data from different nodes is difficult to calculate without transmitting the original data.
2. The dual variables of samples should satisfy constraints in (2). In the distributed scenarios, the dual variables of the first constraint in (2) are usually from different nodes. Since each node is only allowed to exchange information with its neighbors, it is difficult to optimize these dual variables.

To overcome the first difficulty, we use a random approximate function [28] $z : \mathbb{R}^K \to \mathbb{R}^D$, where $D > K$, to map the data to a D-dimensional space instead of RKHS. In this study, for Gaussian kernel function

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\sigma^2}\right), \tag{4}$$

we adopted $z(\boldsymbol{x}) = [z_{\boldsymbol{\omega}_1}(\boldsymbol{x}), \ldots, z_{\boldsymbol{\omega}_D}(\boldsymbol{x})]^T$, where each dimension $z_{\boldsymbol{\omega}_i}(\boldsymbol{x})$ was

$$z_{\boldsymbol{\omega}_i}(\boldsymbol{x}) = \sqrt{\frac{2}{D}} \cos(\boldsymbol{\omega}_i^T \boldsymbol{x} + \psi_i), \tag{5}$$

where ψ_i is drawn uniformly from $[0, 2\pi]$, and $\boldsymbol{\omega}_i$ is drawn from the Fourier transform of Gaussian kernel function

$$p(\boldsymbol{\omega}) = (2\pi)^{-\frac{K}{2}} \exp\left(-\frac{\sigma^2 \|\boldsymbol{\omega}\|^2}{2}\right). \tag{6}$$

As proved in [28], if dimensional number D is large enough, $z(\boldsymbol{x})^T z(\boldsymbol{x}')$ can approximate $K(\boldsymbol{x}, \boldsymbol{x}')$ well, and $z(\boldsymbol{x})$ can approximate $\phi(\boldsymbol{x})$ well. According to Cover's theorem [29], a complex pattern-classification problem nonlinearly cast in a high-dimensional space is more likely to be linearly separable than it is in a low-dimensional space. Therefore, to ensure good performance, we should set a relatively large D. For other shift-invariant kernels such as Laplacian and Cauchy, the authors in [28] provided corresponding finite-dimensional random approximate functions. For additive homogeneous kernels, such as Hellinger's, χ^2, intersection and Jensen-Shannon, the authors in [30] also provided efficient finite-dimensional approximate mapping functions. For a linear kernel function, random approximation is not necessary, so we defined $z(\boldsymbol{x}) = \phi(\boldsymbol{x}) = \boldsymbol{x}$.

With the random approximation, mapping function $\phi(\boldsymbol{x})$ in (3) is replaced by $z(\boldsymbol{x})$. The calculation of $z(\boldsymbol{x})$ only requires one data point from a single node instead of a pair of data from different nodes like the kernel function, so the first difficulty is solved.

After the random approximation is performed, the data are mapped into a D-dimensional feature space instead of the RKHS with unknown dimension. Thus, we could directly solve the primal problem instead of the dual problem, which automatically tackles the second difficulty.

With the use of hinge loss function $L(x) = \max(1 - x, 0)$ [31], the problem can be rewritten as follows:

$$\begin{aligned} \min \frac{1}{2} \sum_{m=1}^{M} \|\boldsymbol{w}_m\|^2 + C \sum_{m=1}^{M} \sum_{q=1}^{Q-1} \sum_{p=1}^{q} \sum_{i=1}^{N_m^p} L(b_{m,q} - \boldsymbol{w}_m \cdot z(\boldsymbol{x}_{m,i}^p)) \\ + C \sum_{m=1}^{M} \sum_{q=1}^{Q-1} \sum_{p=q+1}^{Q} \sum_{i=1}^{N_m^p} L(\boldsymbol{w}_m \cdot z(\boldsymbol{x}_{m,i}^p) - b_{m,q}) \\ \text{s.t. } \boldsymbol{w}_m = \boldsymbol{w}_n, \boldsymbol{b}_m = \boldsymbol{b}_n, \forall (m, n) \in \mathcal{E}. \end{aligned} \tag{7}$$

4.4. Sparse Regularization

In the above steps, a D-dimensional random approximate function $z(\boldsymbol{x})$ is used to approximate the unknown mapping function $\phi(\boldsymbol{x})$. In general, a large D can lead to small approximation error and good classification performance. However, an overlarge D may cause redundancy, which wastes storage space, and brings high computational complexity and high communication costs. There is a trade-off between the above two aspects, so we added a sparse regularization term. The regularization term pushes some dimensions of $\boldsymbol{w}_m$ to 0, which means that these dimensions are redundant and can be discarded. When some dimensions of $\boldsymbol{w}_m$ converge to 0, these dimensions do not need to be calculated, stored and transmitted.

The l_0-norm is typically used to measure sparsity. However, it is nonconvex, and l_0-norm-based problems are NP-hard. In practice, we can use the l_1-norm as a convex approximation of the l_0-norm. Introducing the l_1-norm into the objective function in (7), we obtain

$$\begin{aligned} \min \sum_{m=1}^{M} \left[(1-\alpha)\frac{1}{2}\|\boldsymbol{w}_m\|^2 + \alpha\|\boldsymbol{w}_m\|_1 \right] \\ + C\sum_{m=1}^{M}\sum_{q=1}^{Q-1}\sum_{p=1}^{q}\sum_{i=1}^{N_m^p} L(b_{m,q} - \boldsymbol{w}_m \cdot z(\boldsymbol{x}_{m,i}^p)) \\ + C\sum_{m=1}^{M}\sum_{q=1}^{Q-1}\sum_{p=q+1}^{Q}\sum_{i=1}^{N_m^p} L(\boldsymbol{w}_m \cdot z(\boldsymbol{x}_{m,i}^p) - b_{m,q}) \\ \text{s.t. } \boldsymbol{w}_m = \boldsymbol{w}_n, \boldsymbol{b}_m = \boldsymbol{b}_n, \forall (m,n) \in \mathcal{E}, \end{aligned} \tag{8}$$

where $\alpha \in [0,1]$ controls the proportion of the l_1-norm sparsity regularization term in the entire regularization term. A larger α can lead to a sparser solution of $\boldsymbol{w}_m$. Therefore, since we set a relatively large D to ensure good performance, we could set a relatively large α to reduce redundancy.

We could view this problem from another perspective. If the last two terms in (8) are regarded to be the objective function, the first two terms combined together can be seen as a similar penalty to the elastic net penalty in [32], where α measures the weight of the l_1-norm penalty term.

After the above steps, we transformed Problem (3) into a convex optimization problem with consensus constraints (8).

4.5. Distributed SVOR Algorithm

In this subsection, we propose the dSVOR algorithm to solve Problem (8). First, we used the following notation for convenience

$$\begin{aligned} J_m(\theta_m) = (1-\alpha)\frac{1}{2}\|\boldsymbol{w}_m\|^2 + \alpha\|\boldsymbol{w}_m\|_1 + C\sum_{q=1}^{Q-1}\sum_{p=1}^{q}\sum_{i=1}^{N_m^p} L(b_{m,q} - \boldsymbol{w}_m \cdot z(\boldsymbol{x}_{m,i}^p)) \\ + C\sum_{q=1}^{Q-1}\sum_{p=q+1}^{Q}\sum_{i=1}^{N_m^p} L(\boldsymbol{w}_m \cdot z(\boldsymbol{x}_{m,i}^p) - b_{m,q}), \end{aligned} \tag{9}$$

which is a convex function. The calculation of $J_m(\theta_m)$ does not need the data and estimated parameters from other nodes. Then, Problem (8) can be rewritten as follows:

$$\begin{aligned} \min J = \sum_{m=1}^{M} J_m(\theta_m) \\ \text{s.t. } \theta_m = \theta_n, \forall (m,n) \in \mathcal{E}. \end{aligned} \tag{10}$$

To deal with consensus constraints $\theta_m = \theta_n, \forall (m,n) \in \mathcal{E}$, we adopted the penalty function method. The penalty function used in this paper is $\|\theta_m - \theta_n\|^2$, and the corresponding positive penalty coefficient is λ_{mn}. Then, the optimization problem becomes

$$\min \sum_{m=1}^{M} J_m(\theta_m) + \sum_{(m,n)\in\mathcal{E}} \lambda_{mn}\|\theta_m - \theta_n\|^2 \tag{11}$$

The larger the λ_{mn} is, the closer the solutions of Problems (11) and (10) are.

We then applied the subgradient method to optimize Problem (11). For the hinge loss function $L(x) = \max(1 - x, 0)$, we adopted the following subgradient:

$$L'(x) = \begin{cases} -1, & x < 1 \\ 0, & x \geq 1 \end{cases}, \tag{12}$$

and for the l_1-norm, we adopted

$$\operatorname{sgn}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \\ 0, & x = 0 \end{cases}. \tag{13}$$

At step $k+1$, the iterative equation is

$$\theta_m^{k+1} = \theta_m^k - \eta^k \nabla_{\theta_m} J_m(\theta_m^k) - 2\eta^k \sum_{n \in \mathcal{N}_m} \lambda_{mn}(\theta_m^k - \theta_n^k), \tag{14}$$

where η^k is the step size in step $k+1$, which is positive. The specific subgradients are

$$\begin{aligned} \nabla_{\boldsymbol{w}_m} J_m(\theta_m^k) = {} & (1-\alpha)\boldsymbol{w}_m^k + \alpha \operatorname{sgn}(\boldsymbol{w}_m^k) \\ & - C \sum_{q=1}^{Q-1} \sum_{p=1}^{q} \sum_{i=1}^{N_m^p} L'(b_{m,q}^k - \boldsymbol{w}_m^k \cdot z(\boldsymbol{x}_{m,i}^p)) z(\boldsymbol{x}_{m,i}^p) \\ & + C \sum_{q=1}^{Q-1} \sum_{p=q+1}^{Q} \sum_{i=1}^{N_m^p} L'(\boldsymbol{w}_m^k \cdot z(\boldsymbol{x}_{m,i}^p) - b_{m,q}^k) z(\boldsymbol{x}_{m,i}^p), \end{aligned} \tag{15}$$

$$\begin{aligned} \nabla_{b_{m,q}} J_m(\theta_m^k) = {} & C \sum_{p=1}^{q} \sum_{i=1}^{N_m^p} L'(b_{m,q}^k - \boldsymbol{w}_m^k \cdot z(\boldsymbol{x}_{m,i}^p)) \\ & - C \sum_{p=q+1}^{Q} \sum_{i=1}^{N_m^p} L'(\boldsymbol{w}_m^k \cdot z(\boldsymbol{x}_{m,i}^p) - b_{m,q}^k). \end{aligned} \tag{16}$$

In the subgradient method, in order to converge to the optimal solution, step size η^k should satisfy [33]

$$\sum_{k=0}^{+\infty} \eta^k = +\infty, \text{ and } \sum_{k=0}^{+\infty} (\eta^k)^2 < +\infty. \tag{17}$$

We can rearrange Iterative Equation (14) as follows.

$$\theta_m^{k+1} = (1 - 2\eta^k \sum_{n \in \mathcal{N}_m} \lambda_{mn}) \theta_m^k + \sum_{n \in \mathcal{N}_m} 2\eta^k \lambda_{mn} \theta_n^k - \eta^k \nabla_{\theta_m} J_m(\theta_m^k). \tag{18}$$

If we use the following notations for convenience

$$c_{mn} = 2\eta^k \lambda_{mn}, \quad c_{mm} = 1 - \sum_{n \in \mathcal{N}_m} 2\eta^k \lambda_{mn}, \tag{19}$$

the iterative equation can be rewritten as

$$\theta_m^{k+1} = \sum_{n \in \mathcal{N}_m \cup \{m\}} c_{mn} \theta_n^k - \eta^k \nabla_{\theta_m} J_m(\theta_m^k). \tag{20}$$

It can be divided into two steps, i.e., a combination step and an adaption step:

$$\phi_m^k = \sum_{n \in \mathcal{N}_m \cup \{m\}} c_{mn} \theta_n^k, \tag{21}$$

$$\theta_m^{k+1} = \phi_m^k - \eta^k \nabla_{\theta_m} J_m(\theta_m^k). \tag{22}$$

In Combination Step (21), node m combines the parameters estimated by its neighbors and itself to obtain an intermediate estimate ϕ_m^k, where the combination coefficient of node m and its neighbor n is denoted as c_{mn}. In Adaption Step (22), node m uses the subgradient calculated by using only its own data to update θ_m.

Combination coefficients $\{c_{mn}\}_{\forall(m,n)\in\mathcal{E}}$ represent a cooperation rule among nodes. Equation (19) was not used to define $\{c_{mn}\}$ because λ_{mn} was not defined in advance. In distributed algorithms, combination coefficients are generally determined by a certain cooperative protocol. In this study, we used the Metropolis rule [34]:

$$c_{mn} = \begin{cases} \dfrac{1}{\max(|\mathcal{N}_m|, |\mathcal{N}_n|)}, & n \in \mathcal{N}_m \\ 1 - \displaystyle\sum_{n\in\mathcal{N}_m} c_{mn}, & m = n, \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

where $|\mathcal{N}_m|$ denotes the degree of node m, and

$$\boldsymbol{C}\mathbf{1} = \mathbf{1}, \mathbf{1}^T\boldsymbol{C} = \mathbf{1}^T, \tag{24}$$

where $\boldsymbol{C}$ is an $M \times M$ matrix whose entries are defined by (23).

Equation (19) shows that $\lambda_{mn} = \frac{c_{mn}}{2\eta^k}$. Step size η^k satisfies (17), where the latter implies that $\lim_{k\to\infty} \eta^k = 0$. As $k \to \infty$, step size $\eta^k \to 0$ and penalty coefficient $\lambda_{mn} \to \infty$, which renders the solutions of Problems (11) and (10) nearly equal.

The whole processes of dSVOR are summarized in Algorithm 1.

Algorithm 1 Distributed SVOR algorithm

Initialization: initialize hinge loss function weight C, sparsity regularization weight α, random approximate dimension D, and total iteration number T. Each node m initializes $\theta_m = \{\boldsymbol{w}_m, \boldsymbol{b}_m\}$.
for $k = 1 : T$
  for $m = 1 : M$
    Communication Step: communicate parameters θ_m with neighbors $n \in \mathcal{N}_m$.
  end for
  for $m = 1 : M$
    Combination Step: compute intermediate estimate ϕ_m^k via (21).
    Computation Step: Compute the subgradients $\nabla_{\boldsymbol{w}_m} J_m(\theta_m^k)$, $\nabla_{b_{m,q}} J_m(\theta_m^k)$ via (15) and (16);
    Adaption Step: update θ_m^{k+1} via (22).
  end for
end for

Remark 1. *In the above problems, $\phi(\cdot)$ is a nonlinear mapping function that maps input $\boldsymbol{x}$ into a RKHS for classification, and input $\boldsymbol{x}$ is the original data or extracted features. In general, function $\phi(\cdot)$ can also be regarded to be a generalized feature mapping function that extracts features of $\boldsymbol{x}$, and maps $\boldsymbol{x}$ into a feature space for classification. Thus, it can also use an artificial neural network with learnable parameters. However, that may destroy the convexity of the problem, so that it is no longer guaranteed to converge to the global optimum.*

4.6. Theoretical Analysis

In this subsection, we theoretically analyze the consensus and convergence of dSVOR.

We first introduce a reasonable assumption that is needed in analysis. According to [34], when the graph is not bipartite, this assumption can be guaranteed.

Assumption 1. *Spectral radius* $\rho(\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T) < 1$, *where* $\boldsymbol{C}$ *is the combination coefficient matrix set as in Equation (23).*

Then, we give two theorems about consensus and convergence each.

Theorem 1 (Consensus)**.** *If Assumption 1 holds, and step size* η^k *satisfies Condition (17), then* $\lim_{k\to\infty} \|\theta_m^k - \bar{\theta}^k\| = 0, \forall m$, *where* $\bar{\theta}^k = \frac{1}{M}\sum_{m=1}^M \theta_m^k$.

Theorem 2 (Convergence)**.** *If Assumption 1 holds, and step size* η^k *satisfies Condition (17), then* $\lim_{k\to\infty} \sum_{m=1}^M J_m(\theta_m^k) = J^*$, *where* $J^* = \min J$.

For the proof, see Appendices A and B for details.

5. Experiments

In this section, we carry out experiments on synthetic data and a real-world example to demonstrate the performance of the proposed dSVOR algorithm.

We implemented the following algorithms for comparison:

1. proposed dSVOR algorithm (dSVOR);
2. centralized SVOR (cSVOR), which relies on all the data available in a central node;
3. distributed SVOR with a noncooperative strategy (ncSVOR). In ncSVOR, each node uses only its own data to train a model without any information exchanged with other nodes.

All the algorithms were implemented using the PyTorch framework [35].

There are three points to emphasize:

1. The centralized method needs data in a central node. For comparison, we artificially collected all the data distributed in the nodes of the network together to render it applicable, which is impractical in reality.
2. In cSVOR [11,12], problems were solved by the SMO algorithm instead of subgradient-based algorithms, so we only display its final results.
3. The distributed algorithms were subject to additional constraints, so a distributed algorithm is generally satisfactory if it can achieve comparable performance to the corresponding centralized algorithm.

In this study, we used the prediction accuracy (ACC) and mean absolute error (MAE) on the testing set as the performance evaluation metrics. ACC is a commonly used metric in classification problems, but it does not consider the ordered information of the labels. MAE is the mean absolute deviation of the predicted rank from the true one, which is commonly used in ordinal regression. Using a function $\mathcal{O}(\cdot)$ to denote the position of a certain label in the ordinal scale, i.e., $\mathcal{O}(\mathcal{C}_q) = q, q = 1, \ldots, Q$, we have

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\mathcal{O}(y_i) - \mathcal{O}(\hat{y}_i)| \in [0, Q-1]. \tag{25}$$

The performance of distributed algorithms (dSVOR and ncSVOR) is defined as the mean performance of models obtained by each node. The distributed algorithms ran on a randomly generated connected network that consisted of 20 nodes. For fair comparison, on a certain dataset, all implemented algorithms used the same parameters. All the results were obtained by averaging the results of 10 independent experiments.

5.1. Synthetic Data

In this subsection, we evaluate the performance of all algorithms on two synthetic datasets. On the first dataset, samples could be separated by a set of parallel straight lines if ignoring noises, and samples of the second dataset could be separated by a set of concentric circles. Figure 2a,b show some samples of these two datasets from one of the 10 independent experiments. Both datasets had 1200 samples: 1000 were used as the

training set, and the others were the testing set. The training samples were randomly assigned to 20 nodes to simulate the situation where the data were collected and stored by these nodes in a distributed manner.

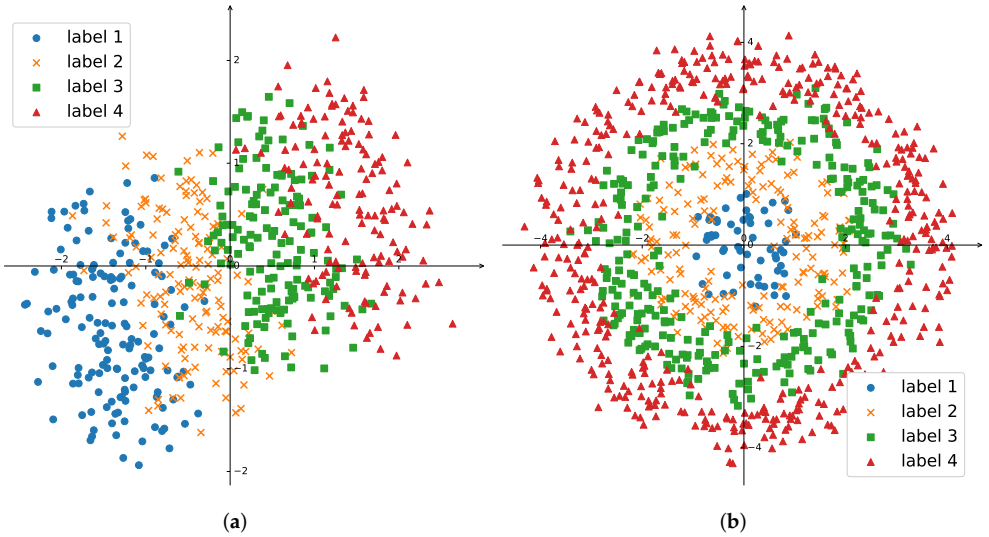


Figure 2. Data visualization of the (**a**) first and (**b**) second synthetic datasets.

These two synthetic datasets were generated with the following methods. For the first dataset, we generated 1200 samples with uniform distribution from a rectangular area $x_{1min} \leq x_1 \leq x_{1max}, x_{2min} \leq x_2 \leq x_{2max}$. We then used three straight lines $\{x_1 = b_i\}_{i=1,2,3}$ to divide this area into 4 parts for 4 classes. The data labels were determined by their locations. Then, Gaussian noise with 0 mean and σ_1 standard deviation was added to each dimension of input vector $\boldsymbol{x} = [x_1\ x_2]^T$. After that, these samples were rotated around the origin with β. Without loss of generality, in the experiments, these parameters were set as follows:

$$x_{1min} = -2, x_{1max} = 2, x_{2min} = -1, x_{2max} = 1,$$
$$b_1 = -1, b_2 = 0, b_3 = 1, \sigma_1 = 0.5, \beta = \frac{\pi}{8}.$$

For the second dataset, we generated 1200 samples with uniform distribution from a circle $x_1^2 + x_2^2 < R^2$, which could be divided into four parts by three concentric circles $\{x_1^2 + x_2^2 = R_i^2\}_{i=1,2,3}$. The data labels were determined by their locations. Then, Gaussian noise with 0 mean and σ_2 standard deviation was added to each dimension of input vector $\boldsymbol{x} = [x_1\ x_2]^T$. Without loss of generality, the parameters were set to be $R = 4$, $R_1 = 1$, $R_2 = 2$, $R_3 = 3$, $\sigma_2 = 0.2$.

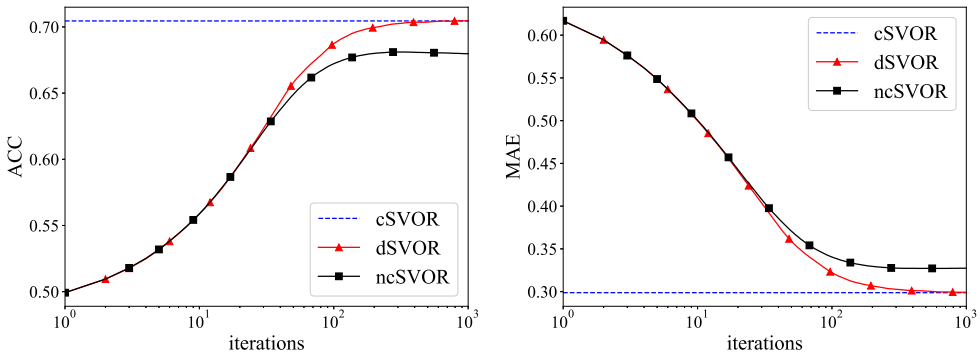
On the first dataset, we used a linear kernel function. In all methods, positive constant C was set to be $1000/N$, where N is the number of samples of all nodes. Because the feature space was only 2-dimensional, the sparse regularization term in our method was not necessary. Thus, we set the coefficient of sparse regularization term $\alpha = 0$. In the distributed algorithm, we used the following diminishing step size:

$$\eta^k = \frac{\eta^0}{1 + \tau k}, \tag{26}$$

which satisfied Condition (17). In (26), parameter η^0 determines the initial step size, and τ determines the decreasing rate of the diminishing step size. We empirically set $\eta^0 = 0.1$ and $\tau = 0.01$ in the following experiments.

Figure 3a,b show the ACC and MAE curves of different algorithms on the first synthetic dataset. As time increased, the MAE of our dSVOR algorithm decreased, and the ACC increased significantly. After about 500 iterations, the dSVOR algorithm converged to a

value that was almost the same as that of cSVOR, while the result of ncSVOR was still some distance away from them. This means that it was not enough for a single node to train a model with good performance using its own data. The proposed dSVOR algorithm, which uses the local data of each node and the parameter estimates from neighbor nodes, could achieve a similar performance to that of the corresponding centralized method.



(**a**) (**b**)

Figure 3. (**a**) ACC and (**b**) MAE curves of different algorithms on the first synthetic dataset.

Figure 4 gives the parameters of each node estimated by different algorithms. In the ncSVOR algorithm, the estimated parameters obtained by different nodes were quite different. Thus, the model obtained by each node with its own data was quite different from the model trained using all the data. In contrast, the estimated parameters of different nodes in dSVOR were almost the same as the parameters in cSVOR. This illustrates the consensus of the proposed dSVOR algorithm. Because we used a linear kernel function here, optimal direction $\boldsymbol{w}$ in the centralized method had an explicit expression that allowed for us to compare it with the estimates of the distributed algorithms. In the following experiments using nonlinear kernel functions, we do not give the results about consensus.

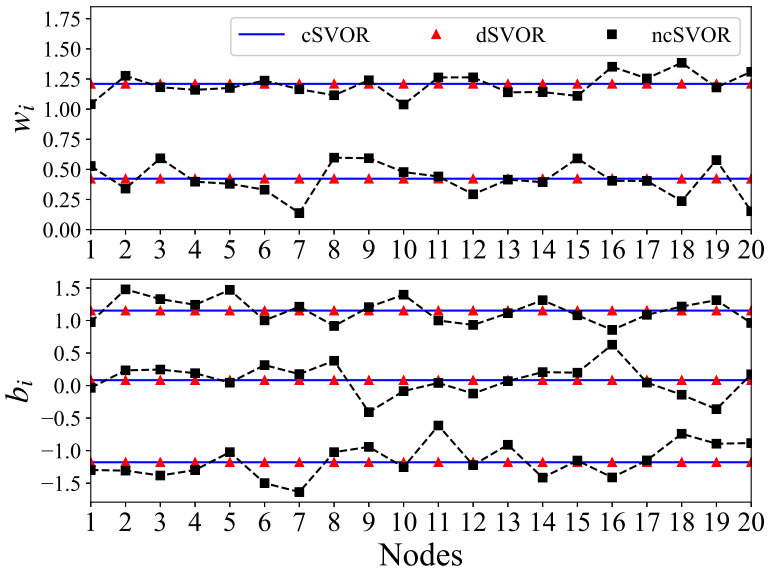
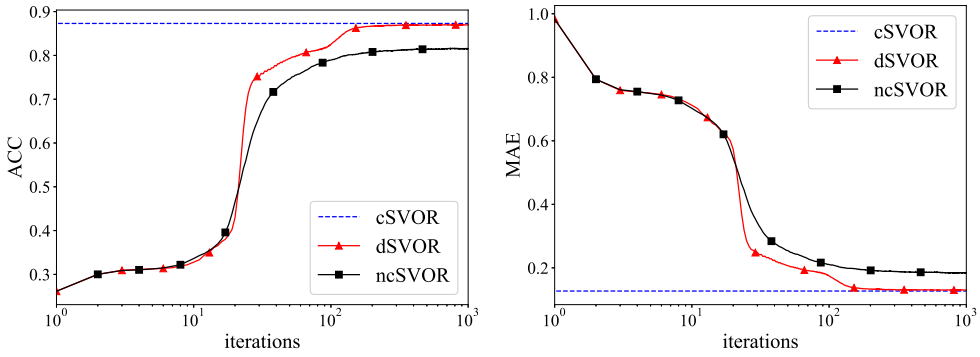


Figure 4. Final estimated parameters of different methods.

On the second dataset, we used a Gaussian kernel function. The kernel size was set to be $\sigma = \frac{1}{K}$ after Z-score normalization, where K is the dimension of input space. In all methods, positive constant C was set to be $1000/N$. As analyzed before, in our method, we set a relatively large D and a relatively large a, $D = 200$, $\alpha = 0.9$. α was not set to 1 because

we wanted to use the strong convexity of the l_2-norm regularization term to increase the convexity of the objective function, which is theoretically beneficial to the optimization of the problem. The learning rate parameters were still set to be $\eta^0 = 0.1$ and $\tau = 0.01$.

Figure 5a,b show the ACC and MAE curves of different algorithms on the second synthetic dataset. The proposed dSVOR algorithm was able to obtain almost the same result as that of the centralized method, while ncSVOR could not.



(**a**) (**b**)

Figure 5. (**a**) ACC and (**b**) MAE curves of different algorithms on the second synthetic dataset.

We also conducted experiments under different hyperparameters D and α to show the parameter sensitivity of dSVOR. Figure 6 gives the MAEs of dSVOR for different D when α was fixed as 0.9. As D increased, the performance of dSVOR gradually improved and was eventually almost the same as that of the centralized method. With a relatively large approximation dimension $D \geq 100$, dSVOR could always obtain a similar MAE to that of cSVOR. However, as mentioned before, an overlarge D may cause redundancy. So, when using a large D to ensure good performance, it is better to use the sparse regularization term to reduce the redundancy. Figure 7a,b gives the MAEs of dSVOR and the proportions of dimensions of $\boldsymbol{w}_m$ that were equal to 0 for different α when D is fixed as 200. The MAE was stable under different α, but the sparsity of $\boldsymbol{w}_m$ was greatly affected by α. A small α led to a dense $\boldsymbol{w}_m$, which caused a lot of redundancy. A large α could bring a sparse $\boldsymbol{w}_m$, where the dimensions that converged to 0 could no longer be stored, calculated, and transmitted after converging to 0, thus saving storage, computation, and communication resources.

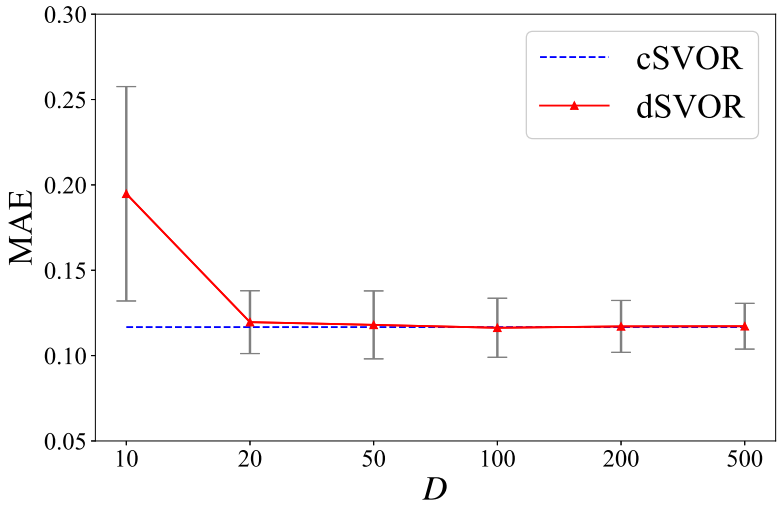
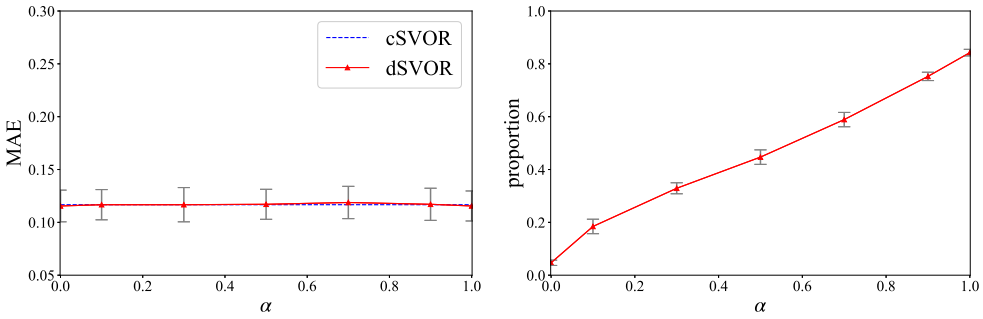


Figure 6. MAEs of dSVOR on the second synthetic dataset for different D when α is fixed as 0.9.



(**a**) (**b**)

Figure 7. Results of dSVOR on the second synthetic dataset for different α when D is fixed as 200. (**a**) MAEs; (**b**) proportions of dimensions of $\boldsymbol{w}_m$ that are equal to 0.

5.2. A Real-World Example

We now take the distributed fault severity diagnosis of rolling element bearings as a real-world example to illustrate the effectiveness of dSVOR.

Rolling element bearings are widely used in factory equipments. The fault severity diagnosis of bearings is a crucial task to ensure reliability in industrial processes. In recent years, data-driven methods have been widely used to identify faults and their severity [36]. To achieve good performance, these data-driven methods usually require a lot of data. However, due to the rarity of faults, a single sensor can only collect very few fault data, and the faults encountered by each factory may also be different. Thus, data from many sensors in many factories are needed to train a proper model. Sometimes, factories may not want to leak the data about their equipments, so it is not allowed to transmit the data to others. The centralized methods which need all the data available in a central node become inapplicable. The distributed methods become a better choice. Taking into account the ordinal information in the fault severity, it is suitable to apply the proposed dSVOR algorithm.

In this study, we used the rolling element bearings data provided by the Case Western Reserve University (CWRU) [37] for experiments. CWRU data were the vibration signals of drive end and fan end bearings collected by sensors at 12,000 and 48,000 samples/s under four different loads of 0–3 hp. There are three types of faults: outer race (OR), inner race (IR), and ball (B) faults, and each type has at most four severity levels (fault width: 0.18, 0.36, 0.53, 0.71 mm). In the experiments, we used drive end bearing data collected at 12,000 samples/s, and performed 4-level fault severity diagnosis in a total of 12 situations (3 different fault types and 4 different loads).

We adopted the feature based on permutation entropy (PE) proposed in [38] as the input $\boldsymbol{x}$. For one datum, we intercepted a sequence of length 2400 from vibration signal data. This sequence was decomposed into a series of intrinsic mode functions (IMFs) by ensemble empirical mode decomposition (EEMD) with 100 ensembles and 0.2 noise amplitude to catch information on multiple time scales. Then, the PE values of the first 5 IMFs are calculated as the input feature of this piece of data.

For each fault severity level, we randomly took 300 training samples and 200 testing samples, and the samples in the testing set were different from those in the training set. For 4-level fault level diagnosis, there were a total of 1200 training samples and 800 testing samples. These training samples were randomly assigned to 20 nodes to simulate the situation where the data were collected and stored by these nodes in a distributed manner.

In the experiments, we used a Gaussian kernel function with kernel size $\sigma = \frac{1}{K}$ after Z-score normalization. In all methods, positive constant C was set to be 10,000/N. In our method, we still set a relatively large hyperparameter D and α, $D = 200$, $\alpha = 0.9$. The other parameters used the same settings as before, i.e., $\eta^0 = 0.1$ and $\tau = 0.01$.

Table 1 shows the experimental results where the value was the mean ± standard deviation of 10 independent experiments. The performance of ncSVOR was worse than that of cSVOR because each node only had part of the training samples that were not enough to represent the entire training set to train a proper model. Compared to ncSVOR, the proposed dSVOR algorithm could achieve similar results to those of cSVOR. In dSVOR, each node can only use the data of its own and exchange some estimated parameters with neighbor nodes. It was satisfactory to be able to achieve performance close to that of the centralized method that uses all the data from all nodes.

Table 1. ACCs and MAEs of different algorithms in a real-world example (mean ± std).

Fault Type	Load	cSVOR		ncSVOR		dSVOR	
		ACC	MAE	ACC	MAE	ACC	MAE
OR	0	0.9585 ± 0.0057	0.0415 ± 0.0057	0.7977 ± 0.0211	0.2069 ± 0.0230	0.9553 ± 0.0064	0.0447 ± 0.0064
	1	0.9317 ± 0.0147	0.0683 ± 0.0147	0.7376 ± 0.0228	0.2726 ± 0.0264	0.9278 ± 0.0136	0.0727 ± 0.0138
	2	0.9547 ± 0.0091	0.0457 ± 0.0096	0.7901 ± 0.0136	0.2172 ± 0.0153	0.9517 ± 0.0094	0.0492 ± 0.0099
	3	0.9253 ± 0.0099	0.0747 ± 0.0099	0.7599 ± 0.0158	0.2489 ± 0.0173	0.9243 ± 0.0095	0.0758 ± 0.0096
IR	0	0.8853 ± 0.0133	0.1149 ± 0.0133	0.7472 ± 0.0087	0.2589 ± 0.0091	0.8844 ± 0.0120	0.1157 ± 0.0120
	1	0.8624 ± 0.0112	0.1376 ± 0.0112	0.7288 ± 0.0103	0.2781 ± 0.0110	0.8556 ± 0.0137	0.1444 ± 0.0137
	2	0.8435 ± 0.0109	0.1565 ± 0.0109	0.7071 ± 0.0116	0.3000 ± 0.0133	0.8391 ± 0.0113	0.1611 ± 0.0113
	3	0.8726 ± 0.0095	0.1291 ± 0.0091	0.7238 ± 0.0110	0.2918 ± 0.0122	0.8632 ± 0.0094	0.1392 ± 0.0091
B	0	0.7768 ± 0.0110	0.2586 ± 0.0129	0.5440 ± 0.0184	0.5975 ± 0.0311	0.7594 ± 0.0221	0.2771 ± 0.0245
	1	0.7836 ± 0.0105	0.2419 ± 0.0099	0.5770 ± 0.0124	0.5284 ± 0.0195	0.7710 ± 0.0067	0.2540 ± 0.0106
	2	0.8256 ± 0.0088	0.1886 ± 0.0088	0.5820 ± 0.0156	0.5341 ± 0.0264	0.8177 ± 0.0147	0.1980 ± 0.0150
	3	0.8627 ± 0.0167	0.1541 ± 0.0193	0.6345 ± 0.0138	0.4648 ± 0.0253	0.8485 ± 0.0169	0.1710 ± 0.0204

Taking the dataset of the IR fault type and 0 hp load as examples, we also show the results of dSVOR under different hyperparameters D and α in Figures 8 and 9. Figure 8 shows that, with a relatively large random approximation dimension $D \geq 100$, dSVOR could obtain a similar MAE to that of cSVOR, which illustrates the effectiveness of the random approximation. Figure 9 shows that a relatively large α can lead to a sparse $\boldsymbol{w}_m$ without affecting the MAE performance, thus effectively reducing redundancy.

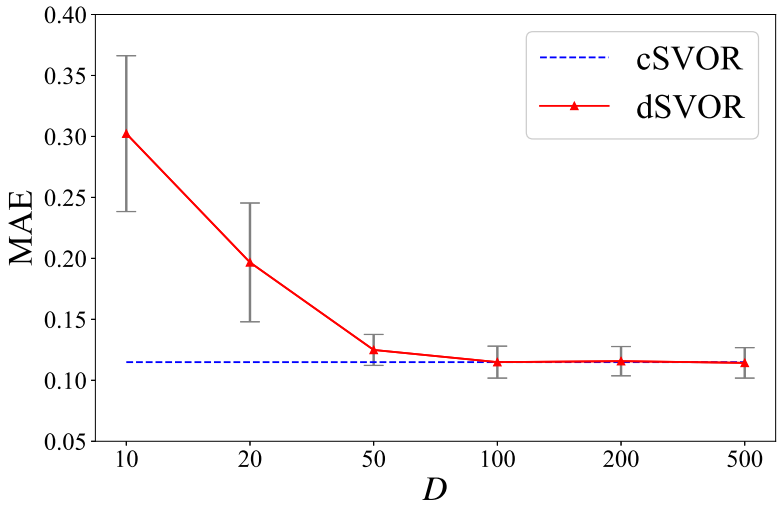
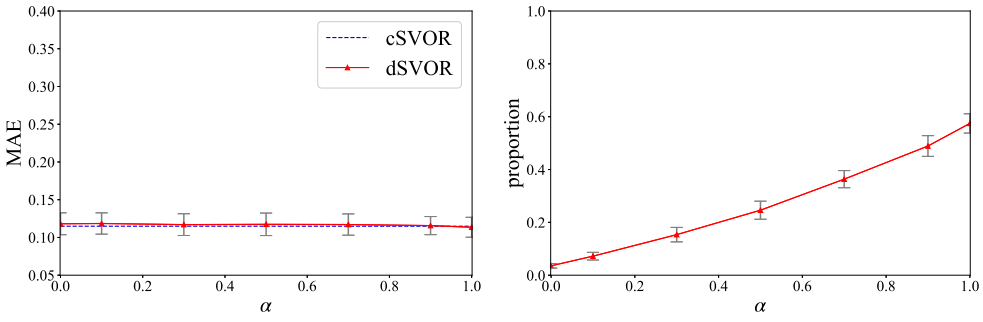


Figure 8. MAEs of dSVOR on the CWRU dataset of IR fault type and 0 hp load for different D when α is fixed as 0.9.



(**a**) (**b**)

Figure 9. The results of dSVOR on the CWRU dataset of IR fault type and 0 hp load for different α when D is fixed as 200 (**a**) the MAEs (**b**) the proportions of dimensions of w_m that are equal to 0.

6. Conclusions

When data are distributedly collected and stored by multiple nodes, and are difficult to transmit to a central node, existing centralized ordinal regression methods become inapplicable. To this end, in order to handle the ordinal regression problem in distribution scenarios, we extended the SVORIM to a distributed version, and derived a distributed SVOR (dSVOR) algorithm. In dSVOR, each node combines the parameters estimated by its neighbors and performs local calculations using only its own data. After convergence, each node can obtain a model whose performance is close to that obtained by the centralized method relying on all the data available in a central node. Theoretically, we analyzed the consensus and the convergence of dSVOR. Practically, we carried out experiments on synthetic data and a real-world example to illustrate its effectiveness.

In our future work, we intend to consider how to automatically determine the proper parameters in dSVOR, e.g., introducing multi-kernel learning to automatically find suitable parameters of random approximate. We also aim to design adaptive strategies for adjusting combination coefficients.

Author Contributions: Conceptualization, H.L. and C.L.; methodology, H.L. and C.L.; formal analysis, H.L., J.T. and C.L.; writing—original draft preparation, H.L.; writing—review and editing, H.L., J.T. and C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (grant No. U20A20158), the Key-Area Research and Development Program of Guangdong Province (grant No. 2021B0101410004), and the National Program for Special Support of Eminent Professionals.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: the authors declare no conflict of interest.

Appendix A. Proof of Theorem 1

Proof. For convenience, we use the following notation:

$$\boldsymbol{\theta}^k = [\theta_1^k, \ldots, \theta_M^k]^T, \quad \boldsymbol{G}_{\boldsymbol{\theta}}^k = [\nabla_{\theta_1} J_1(\theta_1^k), \ldots, \nabla_{\theta_M} J_M(\theta_M^k)]^T. \tag{A1}$$

Then, Iterative Equation (20) can be written as follows.

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{C}\boldsymbol{\theta}^k - \eta^k \boldsymbol{G}_{\boldsymbol{\theta}}^k. \tag{A2}$$

Considering $\bar{\theta}^k = \frac{1}{M}\mathbf{1}^T\boldsymbol{\theta}^k$, the proof of $\lim_{k\to\infty}\|\theta_m^k - \bar{\theta}^k\| = 0, \forall m$ can be done by proving $\lim_{k\to\infty}\|\boldsymbol{\theta}^k - \frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^k\| = 0$. We first construct

$$\begin{aligned}\boldsymbol{\theta}^{k+1} - \frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^{k+1} &= (\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\boldsymbol{\theta}^{k+1} = (\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)(\boldsymbol{C}\boldsymbol{\theta}^k - \eta^k\boldsymbol{G}_{\boldsymbol{\theta}}^k)\\ &= (\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\boldsymbol{\theta}^k - (\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\eta^k\boldsymbol{G}_{\boldsymbol{\theta}}^k.\end{aligned} \tag{A3}$$

Notice that

$$\boldsymbol{C}\frac{1}{M}\mathbf{1}\mathbf{1}^T = \frac{1}{M}\mathbf{1}\mathbf{1}^T = \frac{1}{M}\mathbf{1}\mathbf{1}^T\frac{1}{M}\mathbf{1}\mathbf{1}^T. \tag{A4}$$

We have

$$\begin{aligned}\boldsymbol{\theta}^{k+1} - \frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^{k+1} &= (\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\boldsymbol{\theta}^k - (\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\eta^k\boldsymbol{G}_{\boldsymbol{\theta}}^k\\ &\quad - \boldsymbol{C}\frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^k + \frac{1}{M}\mathbf{1}\mathbf{1}^T\frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^k\\ &= (\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)(\boldsymbol{\theta}^k - \frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^k) - (\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\eta^k\boldsymbol{G}_{\boldsymbol{\theta}}^k.\end{aligned} \tag{A5}$$

For convenience, we use notation $\Delta\boldsymbol{\theta}^k = \boldsymbol{\theta}^k - \frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^k$, then

$$\Delta\boldsymbol{\theta}^{k+1} = (\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\Delta\boldsymbol{\theta}^k - (\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T)\eta^k\boldsymbol{G}_{\boldsymbol{\theta}}^k. \tag{A6}$$

We then prove $\lim_{k\to\infty}\|\Delta\boldsymbol{\theta}^k\| = 0$.

Taking the l_2-norm on both sides of the above equation, we have

$$\|\Delta\boldsymbol{\theta}^{k+1}\| \le \|\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T\|\|\Delta\boldsymbol{\theta}^k\| + \eta^k\|\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T\|\|\boldsymbol{G}_{\boldsymbol{\theta}}^k\| = \rho\|\Delta\boldsymbol{\theta}^k\| + \eta^k c\|\boldsymbol{G}_{\boldsymbol{\theta}}^k\|, \tag{A7}$$

where $c = \|\boldsymbol{I} - \frac{1}{M}\mathbf{1}\mathbf{1}^T\|$ is a positive constant, and ρ denotes the spectral norm of $\boldsymbol{C} - \frac{1}{M}\mathbf{1}\mathbf{1}^T$, which < 1 according to Assumption 1.

Since $J_m(\theta_m)$ is Lipschitz continuous, $\boldsymbol{G}_{\boldsymbol{\theta}}^k$ is bounded, so there exists a positive constant L satisfying $\|\boldsymbol{G}_{\boldsymbol{\theta}}^k\| \le L$. Thus,

$$\|\Delta\boldsymbol{\theta}^{k+1}\| \le \rho\|\Delta\boldsymbol{\theta}^k\| + \eta^k cL. \tag{A8}$$

Now we prove that $\lim_{k\to\infty}\|\Delta\boldsymbol{\theta}^k\| = 0$. To achieve this, we constructed an auxiliary variable u^k that satisfied

$$u^{k+1} = \rho u^k + \eta^k cL, \tag{A9}$$

and $u^0 = \|\Delta\boldsymbol{\theta}^0\| \ge 0$. If $u^k \ge \|\Delta\boldsymbol{\theta}^k\| \ge 0$, then

$$u^{k+1} = \rho u^k + \eta^k cL \ge \rho\|\Delta\boldsymbol{\theta}^k\| + \eta^k cL \ge \|\Delta\boldsymbol{\theta}^{k+1}\|. \tag{A10}$$

So $u^k \ge \|\Delta\boldsymbol{\theta}^k\| \ge 0$ for all $k \ge 0$. With $\rho < 1$ and $\lim_{k\to\infty}\eta^k = 0$, we have $\lim_{k\to\infty}u^k = 0$. Then

$$0 \le \lim_{k\to\infty}\|\Delta\boldsymbol{\theta}^k\| \le \lim_{k\to\infty}u^k = 0. \tag{A11}$$

So we have

$$\lim_{k\to\infty}\|\boldsymbol{\theta}^k - \frac{1}{M}\mathbf{1}\mathbf{1}^T\boldsymbol{\theta}^k\| = \lim_{k\to\infty}\|\Delta\boldsymbol{\theta}^k\| = 0. \tag{A12}$$

The proof of Theorem 1 is completed. □

Appendix B. Proof of Theorem 2

Proof. From Equation (20), we can obtain

$$\begin{aligned}\sum_{m=1}^{M}\theta_m^{k+1} &= \sum_{m=1}^{M}\sum_{n\in\mathcal{N}_m\cup\{m\}} c_{mn}\theta_n^k - \sum_{m=1}^{M}\eta^k\nabla_{\theta_m}J_m(\theta_m^k)\\ &= \sum_{m=1}^{M}\theta_m^k - \sum_{m=1}^{M}\eta^k\nabla_{\theta_m}J_m(\theta_m^k).\end{aligned} \tag{A13}$$

From Theorem 1, we have $\lim_{k\to\infty}\|\theta_m^k-\bar{\theta}^k\|=0$. So, for a sufficiently large $k=k_1$, the above equation can be written as follows:

$$\bar{\theta}^{k_1+1} = \bar{\theta}^{k_1} - \frac{\eta^{k_1}}{M}\sum_{m=1}^{M}\nabla_{\theta_m}J_m(\bar{\theta}^{k_1}), \tag{A14}$$

where $\nabla_{\theta_m}J_m(\bar{\theta}^{k_1})$ denotes the subgradient of $J_m(\theta_m)$ with respect to θ_m when $\theta_m=\bar{\theta}^{k_1}$.

Supposing $\theta^*=\arg\min J$, we have

$$\begin{aligned}\|\bar{\theta}^{k_1+1}-\theta^*\|^2 &= \|\bar{\theta}^{k_1} - \frac{\eta^{k_1}}{M}\sum_{m=1}^{M}\nabla_{\theta_m}J_m(\bar{\theta}^{k_1})-\theta^*\|^2\\ &= \|\bar{\theta}^{k_1}-\theta^*\|^2 + \|\frac{\eta^{k_1}}{M}\sum_{m=1}^{M}\nabla_{\theta_m}J_m(\bar{\theta}^{k_1})\|^2\\ &\quad - 2\frac{\eta^{k_1}}{M}\sum_{m=1}^{M}\nabla_{\theta_m}J_m(\bar{\theta}^{k_1})^T(\bar{\theta}^{k_1}-\theta^*).\end{aligned} \tag{A15}$$

Since $J_m(\theta_m)$ is Lipschitz continuous for all m, there exists a positive constant L satisfying $\|\frac{1}{M}\sum_{m=1}^{M}\nabla_{\theta_m}J_m(\theta_m)\|^2\le L^2$. Thus,

$$\|\bar{\theta}^{k_1+1}-\theta^*\|^2 \le \|\bar{\theta}^{k_1}-\theta^*\|^2 + (\eta^{k_1})^2L^2 - 2\frac{\eta^{k_1}}{M}\sum_{m=1}^{M}\nabla_{\theta_m}J_m(\bar{\theta}^{k_1})^T(\bar{\theta}^{k_1}-\theta^*). \tag{A16}$$

Because $J_m(\theta_m)$ is convex, we have

$$\nabla_{\theta_m}J_m(\bar{\theta}^{k_1})^T(\bar{\theta}^{k_1}-\theta^*) \ge J_m(\bar{\theta}^{k_1}) - J_m(\theta^*). \tag{A17}$$

Then

$$\begin{aligned}\|\bar{\theta}^{k_1+1}-\theta^*\|^2 &\le \|\bar{\theta}^{k_1}-\theta^*\|^2 + (\eta^{k_1})^2L^2 - 2\frac{\eta^{k_1}}{M}\sum_{m=1}^{M}(J_m(\bar{\theta}^{k_1}) - J_m(\theta^*))\\ &= \|\bar{\theta}^{k_1}-\theta^*\|^2 + (\eta^{k_1})^2L^2 - 2\frac{\eta^{k_1}}{M}(J(\bar{\theta}^{k_1}) - J^*).\end{aligned} \tag{A18}$$

If $\lim_{k\to\infty}\sum_{m=1}^{M}J_m(\theta_m^k)\neq J^*$, $\exists\varepsilon>0, k_2>0$, $\forall k\ge k_2, J(\bar{\theta}^{k_2})-J^*>\varepsilon$. Let $k_3=\max\{k_1,k_2\}$. Then

$$\|\bar{\theta}^{k_3+1}-\theta^*\|^2 < \|\bar{\theta}^{k_3}-\theta^*\|^2 + (\eta^{k_3})^2L^2 - 2\frac{\eta^{k_3}}{M}\varepsilon. \tag{A19}$$

Taking the summation of both sides of the above equation over $k=k_3,\ldots,k_3+k^*$, we obtain

$$\|\bar{\theta}^{k_3+k^*}-\theta^*\|^2 < \|\bar{\theta}^{k_3}-\theta^*\|^2 + \sum_{k=k_3}^{k_3+k^*}(\eta^k)^2L^2 - \sum_{k=k_3}^{k_3+k^*}2\frac{\eta^k}{M}\varepsilon. \tag{A20}$$

Since $\|\bar{\theta}^{k_3+k^*} - \theta^*\|^2 \geq 0$, we have

$$\|\bar{\theta}^{k_3} - \theta^*\|^2 + \sum_{k=k_3}^{k_3+k^*} (\eta^k)^2 L^2 > \sum_{k=k_3}^{k_3+k^*} 2\frac{\eta^k}{M}\varepsilon. \tag{A21}$$

Thus,

$$\frac{\|\bar{\theta}^{k_3} - \theta^*\|^2 + L^2 \sum_{k=k_3}^{k_3+k^*} (\eta^k)^2}{\frac{2}{M}\sum_{k=k_3}^{k_3+k^*} \eta^k} > \varepsilon. \tag{A22}$$

Since $\sum_{k=0}^{+\infty} \eta^k = +\infty$ and $\sum_{k=0}^{+\infty} (\eta^k)^2 < +\infty$,

$$\lim_{k^*\to\infty} \frac{\|\bar{\theta}^{k_3} - \theta^*\|^2 + L^2 \sum_{k=k_3}^{k_3+k^*} (\eta^k)^2}{\frac{2}{M}\sum_{k=k_3}^{k_3+k^*} \eta^k} = 0, \tag{A23}$$

which conflicts with Equation (A22). Thus,

$$\lim_{k\to\infty} \sum_{m=1}^{M} J_m(\theta_m^k) = J^*. \tag{A24}$$

The proof of Theorem 2 is completed. □

References

1. Doyle, O.M.; Westman, E.; Marqu, A.F.; Mecocci, P.; Vellas, B.; Tsolaki, M.; Kłoszewska, I.; Soininen, H.; Lovestone, S.; Williams, S.C.; et al. Predicting progression of alzheimer's disease using ordinal regression. *PLoS ONE* **2014**, *9*, e105542. [CrossRef]
2. Allen, J.; Eboli, L.; Mazzulla, G.; Ortúzar, J.D. Effect of critical incidents on public transport satisfaction and loyalty: An Ordinal Probit SEM-MIMIC approach. *Transportation* **2020**, *47*, 827–863. [CrossRef]
3. Gutiérrez, P.A.; Salcedo-Sanz, S.; Hervás-Martínez, C.; Carro-Calvo, L.; Sánchez-Monedero, J.; Prieto, L. Ordinal and nominal classification of wind speed from synoptic pressurepatterns. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1008–1015. [CrossRef]
4. Cao, W.; Mirjalili, V.; Raschka, S. Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognit. Lett.* **2020**, *140*, 325–331. [CrossRef]
5. Hirk, R.; Hornik, K.; Vana, L. Multivariate ordinal regression models: An analysis of corporate credit ratings. *Stat. Method. Appl.* **2019**, *28*, 507–539. [CrossRef]
6. Zhao, X.; Zuo, M.J.; Liu, Z.; Hoseini, M.R. Diagnosis of artificially created surface damage levels of planet gear teeth using ordinal ranking. *Measurement* **2013**, *46*, 132–144. [CrossRef]
7. Kotsiantis, S.B.; Pintelas, P.E. A cost sensitive technique for ordinal classification problems. In Proceedings of the 3rd Hellenic Conference on Artificial Intelligence, Samos, Greece, 5–8 May 2004; pp. 220–229.
8. Tu, H.-H.; Lin, H.-T. One-sided support vector regression for multiclass cost-sensitive classification. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 49–56.
9. Harrington, E.F. Online ranking/collaborative filtering using the perceptron algorithm. In Proceedings of the 20th International Conference on Machine Learning, Washington, DC, USA, 21–24 August 2003; pp. 250–257.
10. Gutiérrez, P.A.; Perez-Ortiz, M.; Sanchez-Monedero, J.; Fernez-Navarro, F.; Hervas-Martinez, C. Ordinal regression methods: Survey and experimental study. *IEEE Trans. Knowl. Data Eng.* **2015**, *28*, 127–146. [CrossRef]
11. Chu, W.; Keerthi, S.S. New approaches to support vector ordinal regression. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 145–152.
12. Chu, W.; Keerthi, S.S. Support vector ordinal regression. *Neural Comput.* **2007**, *19*, 792–815. [CrossRef]
13. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
14. Liu, H.; Tu, J.; Li, C. Distributed Ordinal Regression Over Networks. *IEEE Access* **2021**, *9*, 62493–62504. [CrossRef]
15. McCullagh, P. Regression models for ordinal data. *J. Royal Stat. Soc. Ser. B Methodol.* **1980**, *42*, 109–142. [CrossRef]
16. Williams, R. Understanding and interpreting generalized ordered logit models. *J. Math. Sociol.* **2016**, *40*, 7–20. [CrossRef]
17. Wang, H.; Shi, Y.; Niu, L.; Tian, Y. Nonparallel Support Vector Ordinal Regression. *IEEE Trans. Cybern.* **2017**, *47*, 3306–3317. [CrossRef]
18. Jiang, H.; Yang, Z.; Li, Z. Non-parallel hyperplanes ordinal regression machine. *Knowl.-Based Syst.* **2021**, *216*, 106593. [CrossRef]
19. Li, L.; Lin, H.-T. Ordinal regression by extended binary classification. *Adv. Neural Inf. Process. Syst.* **2006**, *19*, 865–872.

20. Liu, X.; Fan, F.; Kong, L.; Diao, Z.; Xie, W.; Lu, J.; You, J. Unimodal regularized neuron stick-breaking for ordinal classification. *Neurocomputing* **2020**, *388*, 34–44. [CrossRef]
21. Cattivelli, F.S.; Sayed, A.H. Diffusion LMS strategies for distributed estimation. *IEEE Trans. Signal Process.* **2009**, *58*, 1035–1048. [CrossRef]
22. Li, C.; Shen, P.; Liu, Y.; Zhang, Z. Diffusion information theoretic learning for distributed estimation over network. *IEEE Trans. Signal Process.* **2013**, *61*, 4011–4024. [CrossRef]
23. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **2011**, *3*, 1–122. [CrossRef]
24. Yang, T.; Yi, X.; Wu, J.; Yuan, Y.; Wu, D.; Meng, Z.; Hong, Y.; Wang, H.; Lin, Z.; Johansson, K.H. A survey of distributed optimization. *Annu. Rev. Control* **2019**, *47*, 278–305. [CrossRef]
25. Shen, P.; Li, C. Distributed information theoretic clustering. *IEEE Trans. Signal Process.* **2014**, *62*, 3442–3453. [CrossRef]
26. Olfati-Saber, R. Distributed Kalman filtering for sensor networks. In Proceedings of the 46th Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007; pp. 5492–5498.
27. Miao, X.; Liu, Y.; Zhao, H.; Li, C. Distributed online one-class support vector machine for anomaly detection over networks. *IEEE Trans. Cybern.* **2018**, *49*, 1475–1488. [CrossRef]
28. Rahimi, A.; Recht, B. Random features for large-scale kernel machines. *Adv. Neural Inf. Process. Syst.* **2007**, *20*, 1177–1184.
29. Cover, T.M. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Trans. Electron. Comput.* **1965**, *14*, 326–334. [CrossRef]
30. Vedaldi, A.; Zisserman, A. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 480–492. [CrossRef]
31. Scholkopf, B.; Smola, A.J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
32. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. Royal Stat. Soc. Ser. B* **2005**, *67*, 301–320. [CrossRef]
33. Bertsekas, D. *Convex Optimization Algorithms*; Athena Scientific: Belmont, MA, USA, 2015.
34. Xiao, L.; Boyd, S. Fast linear iterations for distributed averaging. *Syst. Control Lett.* **2004**, *53*, 65–78. [CrossRef]
35. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035.
36. Cerrada, M.; Sánchez, R.V.; Li, C.; Pacheco, F.; Cabrera, D.; de Oliveira, J.V.; Vásquez, R.E. A review on data-driven fault severity assessment in rolling bearings. *Mech. Syst. Signal Process.* **2018**, *99*, 169–196. [CrossRef]
37. Smith, W.A.; Randall, R.B. Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study. *Mech. Syst. Signal Process.* **2015**, *64*, 100–131. [CrossRef]
38. Zhang, X.; Liang, Y.; Zhou, J. A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM. *Measurement* **2015**, *69*, 164–179. [CrossRef]