Neural Networks 106 (2018) 294–302

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Incremental sparse Bayesian ordinal regression

Chang Li *, Maarten de Rijke

University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 8 February 2018
Received in revised form 9 June 2018
Accepted 25 July 2018
Available online 2 August 2018

Keywords:
Ordinal regression
Sparse Bayesian learning
Basis function-based method

ABSTRACT

Ordinal Regression (OR) aims to model the ordering information between different data categories, which is a crucial topic in multi-label learning. An important class of approaches to OR models the problem as a linear combination of basis functions that map features to a high-dimensional non-linear space. However, most of the basis function-based algorithms are time consuming. We propose an incremental sparse Bayesian approach to OR tasks and introduce an algorithm to sequentially learn the relevant basis functions in the ordinal scenario. Our method, called Incremental Sparse Bayesian Ordinal Regression (ISBOR), automatically optimizes the hyper-parameters via the *type-II maximum likelihood* method. By exploiting fast marginal likelihood optimization, ISBOR can avoid big matrix inverses, which is the main bottleneck in applying basis function-based algorithms to OR tasks on large-scale datasets. We show that ISBOR can make accurate predictions with parsimonious basis functions while offering automatic estimates of the prediction uncertainty. Extensive experiments on synthetic and real word datasets demonstrate the efficiency and effectiveness of ISBOR compared to other basis function-based OR approaches.

1. Introduction

The task of modeling ordinal data has attracted attention in various areas, including computer vision (Niu, Zhou, Wang, Gao, & Hua, 2016; Xiao, Liu, & Hao, 2017), information retrieval (Liu, 2009), recommender systems (Hu & Li, 2018) and machine learning (Gu et al., 2012; Gutiérrez, Perez-Ortiz, Sanchez-Monedero, Fernández-Navarro, & Hervas-Martinez, 2016; Gutiérrez, Tiňo, & Hervás-Martínez, 2014; Pérez-Ortiz, Gutiérrez, Carbonero-Ruz, & Hervás-Martínez, 2016; Tang & Tiňo, 2017). Because of the explicit or implicit relationship between labels, simple regression or multi-classification algorithms may fail to find optimal decision boundaries, which motivates the development of dedicated methods.

Generally, Ordinal Regression (OR) algorithms can be classified into three categories: naive approaches, ordinal binary decompositions, and threshold models (Gutiérrez et al., 2016). For naive approaches, OR tasks are simplified into traditional multi-classification or regression tasks, omitting ordering information, and solved by simple machine learning algorithms, e.g., Support Vector Machine (SVM) Regression (Smola & Schölkopf, 2004). For ordinal binary decomposition, the ordinal labels are decomposed into several binary pairs, which are then modeled by a single or multiple classifiers. For the threshold models, the OR problem is addressed by training a threshold model, which models the hidden score function and an implicit set of thresholds that derive the ordinal paradigm. Among these three categories, the third, threshold models, is the most popular way to model the OR problems (Gutiérrez et al., 2016). Thus, in this paper, we focus on threshold models.

Since data may lie in a low-dimensional space where data are not distinguishable by a linear combination of the features, basis functions are widely used in all three types of OR algorithm. The basis function can map features to highly non-linear spaces where the data can be distinguishable by a linear combination of basis functions (Vapnik, 1999). We call this kind of algorithms *basis function-based algorithms*. Most of the current basis function-based OR algorithms do not scale well, as they are batch methods and require access to the full training dataset.

To address this scalability problem, we propose Incremental Sparse Bayesian Ordinal Regression (ISBOR), which utilizes an incremental Bayesian approach to learning. We impose a zero-mean Gaussian prior over function parameters and utilize the ordinal likelihood (Chu & Ghahramani, 2005a), which is regarded as a probit function of OR to model the ordinal relationship between categories. Then we apply the Laplace method (MacKay, 1992) to derive a Maximum a Posteriori (MAP) estimate of the unknown parameters over the dataset. In order to derive a full Bayesian solution, we derive a type-II maximum likelihood optimization (Tipping, 2001), in which Incremental Sparse Bayesian Ordinal Regression (ISBOR) automatically optimizes the thresholds that determine the decision boundaries of ordering categories as hyper-parameters. Finally, to accelerate training, we follow the

* Corresponding author.
E-mail addresses: c.li@uva.nl (C. Li), derijke@uva.nl (M. de Rijke).

https://doi.org/10.1016/j.neunet.2018.07.015

idea of fast marginal likelihood learning (Tipping & Faul, 2003) and derive an incremental training strategy for ISBOR.

With this paper, we make an important step towards efficient ordinal regression based on basis functions. In particular, the main contributions are as follows:

- We propose a basis function-based sequential sparse Bayesian treatment for ordinal regression, ISBOR, which scales well with the number of training samples.
- We provide an experimental evaluation of ISBOR's performance against existing basis function-based OR algorithms in terms of efficacy, efficiency and sparseness.

The remainder of the paper is organized as follows. Section 2 revisits the related work. Section 3 presents ISBOR. Section 4 details the hyper-parameter optimization of ISBOR. We report on the experimental results in Section 5. The paper is concluded in Section 6.

2. Related work

In this paper, we focus on so-called basis function-based approaches to ordinal regression, which brings non-linear patterns to the linear decision functions and are well studied in machine learning. Three types of basis function-based approaches are widely used for the OR task: SVMs (Vapnik, 1999), Gaussian Processes (GP) (Rasmussen, 2004) and Sparse Bayesian Learning (SBL) (Tipping, 2001). SVM approaches convert the learning process to a convex optimization problem for which there are efficient algorithms, e.g., SMO (Keerthi, Shevade, Bhattacharyya, & Murthy, 2001), to find global minima. However, SVM is not equipped with a probabilistic interpretation, as a result of which it is hard to use expert or prior knowledge and make the probabilistic predictions with SVM. GP (Rasmussen, 2004) and SBL are Bayesian methods, which take expert knowledge as prior information and interpret the prediction with the posteriori distribution. In order to conduct Bayesian inference and model selection, most of them require one to compute the inverse of the basis function matrix, which leads to $\mathcal{O}(N^3)$ computational complexity, where N is the number of training samples.

In the following, we describe some of these algorithms to provide context for our work. The SVM-based Support Vector Ordinal Regression (SVOR) approach (Chu & Keerthi, 2007) is an accurate OR algorithm (Gutiérrez et al., 2016). SVOR is optimized using a sequential minimal optimization strategy, which brings the upper bound down to $\mathcal{O}(N^2 \log N)$. Solving SVOR in the dual problem boils down to optimizing with L2-regularization, which leads to a slightly sparse solution.

Incremental Support Vector Machine for Ordinal Regression (ISVOR) (Gu, Sheng, Tay, Romano, & Li, 2015) addresses the problem of basis function-based batch algorithms for OR. It decomposes the OR problem into ordinal binary classification and simultaneously builds decision boundaries with linear computational complexity. However, ISVOR suffers from the problem of stability and it doubles the problem size because of its binary decomposition approach. The main difference between the proposed ISBOR and SVM-based methods is that ISBOR can use prior knowledge and make probabilistic predictions.

Gaussian Process Ordinal Regression (GPOR) (Chu & Ghahramani, 2005a) is the first GP algorithm that has been proposed for the OR task. GPOR employs a GP prior on the latent functions, and uses an ordinal likelihood, which is a generalization of the *probit* function, to estimate the distribution of ordinal data conditional on the model. To conduct model adaptation, GPOR applies two Bayesian inference techniques: Laplace approximation (MacKay, 1992) and expectation propagation approximation (Minka, 2001), respectively. Since approximate Bayesian inference methods require one to compute the inverse of an $N \times N$ matrix, the computational complexity of GPOR is $\mathcal{O}(N^3)$. The main differences between GPOR and ISBOR are twofold:

1. ISBOR is a sparse method, as a result of which the prediction is only based on the relevant samples. In contrast, GPOR makes predictions based on the whole training data.
2. ISBOR is an incremental learning algorithm, while GPOR is a batch algorithm: during training, GPOR needs to compute the matrix inverse of size $N \times N$, while ISBOR only computes the matrix inverse of size $M \times M$, where $M \ll N$ is the number of relevant samples.

Based on GPOR, various OR algorithms have been proposed (Chu & Ghahramani, 2005b; Srijith, Shevade, & Sundararajan, 2012a, b, 2013). However, they are all batch algorithms. In contrast, the proposed method, ISBOR, is an incremental learning algorithm and gets rid of computing the inverse of $N \times N$ matrix.

Based on SBL, Sparse Bayesian Ordinal Regression (SBOR) (Chang, Zheng, & Lin, 2009) builds a probabilistic solution to the OR problem. Here, "sparse" that means SBOR utilizes a sparseness assumption that enables it to make predictions based on a few relevant samples with a $\mathcal{O}(M^3)$ computational bound, where M is the number of relevant samples. However, SBOR is still a batch algorithm and requires one to handle matrix inversion on the full dataset during initial iterations. Other basis function-based batch OR algorithms include Kernel Discriminate Learning for Ordinal Regression (KDLOR) (Sun, Li, Wu, Zhang, & Li, 2010).

In summary, ISBOR differs from the above algorithms in the following ways. Instead of operating in batch, ISBOR utilizes an incremental way to sequentially choose relevant samples. Because of the sparsity assumption, during sequential training ISBOR only selects a small portion of the training data with linear computational complexity in each iteration. Moreover, instead of designing ordinal partitions like ISVOR, ISBOR directly learns the implicit thresholds and score function, which is a more natural way to reveal ordinal relations.

3. Incremental Sparse Bayesian Ordinal Regression

We start this section by defining the notation used in the paper. The training set is $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^d$ is the feature vector, $y_n \in \{1, 2, \ldots, r\}$ is the corresponding category; r is the number of categories. We use normal-face letters to denote scalar and boldface letters to denote vectors and matrices.

We present ISBOR in four steps: model specification, likelihood definition, prior assumption and maximum a posterior.

3.1. Model specification

As a threshold OR model (Gutiérrez et al., 2016), ISBOR chooses a linear combination of basis functions as the score function, $f(\mathbf{x}_n; \mathbf{w})$, which maps a sample from the d-dimensional feature space to a real number:

$$f(\mathbf{x}_n) = \sum_{i=1}^{N} \phi_i(\mathbf{x}_n) w_i = \boldsymbol{\phi}(\mathbf{x}_n)\mathbf{w}, \tag{1}$$

where $\mathbf{w} \in \mathcal{R}^N$ denotes the parameter vector[1] and $\boldsymbol{\phi}(\mathbf{x}_n) = [\phi_1(\mathbf{x_n}), \ldots, \phi_N(\mathbf{x_n})]$ is the basis function, e.g., the Gaussian Radial Basis Function (RBF):

$$\phi(\mathbf{x_n}, \mathbf{x_i}) = \exp(-\theta \|\mathbf{x}_n - \mathbf{x}_i\|_2^2). \tag{2}$$

[1] Here, w_n controls the relevance of the nth basis function $\phi_n(\mathbf{w})$: if $w_n = 0$, the nth basis function is irrelevant for the decision, which is equivalent to throw the nth sample away and retain the relevant basis functions.

After mapping, ISBOR exploits a set of thresholds, $[b_0, \ldots, b_r]$, to determine intervals of different categories. In order to represent the ordering information, these thresholds are chosen as a set of ascending numbers, e.g., $b_{i+1} > b_i$, and work with a set of positive auxiliary numbers, $[\Delta_2, \ldots, \Delta_{r-1}]$, with b_n defined as $b_n = b_1 + \sum_{i=2}^{n} \Delta_i$. During prediction, a sample $\mathbf{x}_n$ is classified to a target y_n if and only if $b_{y_n-1} < f(\mathbf{x}_n) \leq b_{y_n}$. We set $b_0 = -\infty$ and $b_r = \infty$.

3.2. Ordinal likelihood

To model ordinal data, we take the ordinal likelihood proposed in GPOR (Chu & Ghahramani, 2005a). The likelihood is the joint distribution of the samples conditional on the model parameters, and with the I.I.D. assumption; it is computed as follows:

$$p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n \mid \mathbf{X}, \mathbf{w}),$$

where $\mathbf{Y} = \{y_n\}_{n=1}^N$ and $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$. Following the standard probabilistic assumption (Chu & Ghahramani, 2005a), we assume that the outputs of a score function are contaminated with random Gaussian noise: $\hat{y}_n = f(\mathbf{x}_n) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. σ is the standard deviation of the noise distribution, which is learned by the model selection (Section 4.2). In this way, the score function is linked to the probabilistic output $p(\hat{y}_n \mid \mathbf{w}, \mathbf{x}_n, \epsilon) = \mathcal{N}(\hat{y}_n \mid f(\mathbf{x}_n), \sigma^2)$. And the likelihood over a sample is computed as follows:

$$p_{ideal}(y_n \mid \mathbf{x}_n, \mathbf{w}, \epsilon) = \begin{cases} 1 & \text{if } b_{y_n-1} < \hat{y}_n \leq b_{y_n}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Since $b_{i+1} = b_i + \delta_{i+1}$ and $\delta_{i+1} > 0$, $[b_0, \ldots, b_r]$ divide the real line into r ordinal intervals. Thus, with these intervals, the ideal likelihood maps the real value output $f(x)$ to ordinal categories. However, because of the uniform distribution, Eq. (3) is not differentiable, and hence we cannot implement Bayesian inference. To tackle this issue, we integrate out the noise term and obtain a differentiable likelihood as follows:

$$\begin{aligned} p(y_n \mid \mathbf{x}_n, \mathbf{w}, \sigma) &= \int_{\epsilon} p_{ideal}(y_n \mid \mathbf{x}_n, \mathbf{w}, \epsilon) \mathcal{N}(\epsilon \mid 0, \sigma^2) d\epsilon \\ &= \psi(z_{n,1}) - \psi(z_{n,2}), \end{aligned} \tag{4}$$

where

$$z_{n,1} = \frac{b_{y_n} - f(\boldsymbol{x}_n)}{\sigma} \text{ and } z_{n,2} = \frac{b_{y_n-1} - f(\mathbf{x}_n)}{\sigma},$$

and $\psi(z)$ is the Gaussian cumulative distribution function. Based on Eq. (4), maximum likelihood estimation is equivalent to maximizing the area under the standard Gaussian distribution between $z_{n,1}$ and $z_{n,2}$, which is differentiable.

3.3. Priori assumption

For large scale datasets, if we directly learn parameters by maximum likelihood estimation, we may easily encounter severe over-fitting. To avoid this, we add an additional constraint on parameters: the regularization term. In Bayesian learning, we achieve this by introducing a zero-mean Gaussian prior for $\mathbf{w}$: $p(w_n \mid \alpha_n) = \mathcal{N}(w_n; 0, \alpha_n^{-1})$. Assuming that each parameter is mutually independent, the prior over parameters is computed as follows:

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{n=1}^{N} \mathcal{N}(w_n \mid 0, \alpha_n^{-1}), \tag{5}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]$ and α_n, the inverse of variance, serves as the regularization term. If the value of α_n is large, the posterior of w_n will be mainly constrained by the prior and w_n will be bound to a small neighborhood of 0.[2] To complete the definition of the sparse prior, we define a set of flat Gamma hyper-priors over $\boldsymbol{\alpha}$, which together with Gaussian priors result in Student's-t prior and work as L_1 regularization (Tipping, 2001).

3.4. Maximum a posterior

Having defined the prior and likelihood, ISBOR proceeds by computing the posterior over all training data, based on Bayes' rule:

$$p(\mathbf{w} \mid \mathbf{D}) = \frac{p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}, \sigma) p(\mathbf{w} \mid \boldsymbol{\alpha})}{p(\mathbf{D} \mid \boldsymbol{\eta})}, \tag{6}$$

where $\mathbf{D}$ is the training dataset, $p(\mathbf{w} \mid \alpha)$ defined in Eq. (5) is the prior, $p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}, \sigma)$ defined in Eq. (4) is the likelihood, the denominator $p(\mathbf{D} \mid \boldsymbol{\eta}) = \int p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}, \sigma) p(\mathbf{w} \mid \boldsymbol{\alpha}) d\mathbf{w}$ is the marginal likelihood, which we use for model selection and hyper-parameter optimization in the next section. To simplify our notation, we collect all the hyper-parameters, including noise level σ, thresholds and $\boldsymbol{\alpha}$, into $\boldsymbol{\eta}$.

We prefer the $\mathbf{w}^*$ with the highest posterior probability, and formulate the MAP point estimate as $\mathbf{w}^* = \max_{\mathbf{w}} p(\mathbf{w} \mid \mathbf{D})$. However, we cannot integrate $\mathbf{w}$ out in the marginal likelihood analytically. In our MAP estimation we use the fact that $p(\mathbf{w} \mid \mathbf{D}) \propto p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}, \sigma) p(\mathbf{w} \mid \boldsymbol{\alpha})$ and work with the logarithm of the posterior:

$$\begin{aligned} \ln p(\mathbf{w} \mid \mathbf{D}) &= \ln p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{w}, \sigma) + \ln p(\mathbf{w} \mid \boldsymbol{\alpha}) + const \\ &\approx \sum_{n=1}^{N} \ln(\psi(z_{n,1}) - \psi(z_{n,2})) - \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w}, \end{aligned} \tag{7}$$

where $\mathbf{A}$ is a diagonal matrix with diagonal elements $[\alpha_1, \ldots, \alpha_N]$, *const* is a term independent of $\mathbf{w}$. The first part of the last line, from the likelihood, works as the loss term; the second part, from the prior, acts as the regularization term.

Next, the Newton–Raphson method (Ypma, 1995) is applied to compute the MAP estimate. First, we compute the first and second order derivatives of the first term (log-likelihood part), $\mathcal{L} = \ln p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w})$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= -\sum_{n=1}^{N} \frac{1}{\sigma} \frac{\mathcal{N}(z_{n,1} \mid 0, 1) - \mathcal{N}(z_{n,2} \mid 0, 1)}{\psi(z_{n,1}) - \psi(z_{n,2})} \boldsymbol{\phi}_n \\ &= \boldsymbol{\Phi}^T \delta \end{aligned} \tag{8}$$

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{w} \partial \mathbf{w}^T} = -\boldsymbol{\Phi}^T \mathbf{H} \boldsymbol{\Phi}, \tag{9}$$

where

$$\begin{aligned} \delta_n &= \frac{1}{\sigma} \frac{\mathcal{N}(z_{n,1} \mid 0, 1) - \mathcal{N}(z_{n,2} \mid 0, 1)}{\psi(z_{n,1}) - \psi(z_{n,2})} \\ H_{nn} &= \frac{1}{\sigma^2} \left[\left(\frac{N(z_{n,1} \mid 0, 1) - N(z_{n,2} \mid 0, 1)}{\psi(z_{n,1}) - \psi(z_{n,2})} \right)^2 \right. \\ &\quad \left. \times \frac{z_{n,1} N(z_{n,1} \mid 0, 1) - z_{n,2} N(z_{n,2} \mid 0, 1)}{\psi(z_{n,1}) - \psi(z_{n,2})} \right]. \end{aligned}$$

Then, combining Eqs. (7)–(9), we obtain the derivative of the log-posterior as

$$\frac{\partial^2 \log p(\mathbf{w} \mid \mathbf{D})}{\partial \mathbf{w} \partial \mathbf{w}^T} = -\boldsymbol{\Phi}^T \mathbf{H} \boldsymbol{\Phi} - \mathbf{A}.$$

[2] Practically, when w_n is smaller than a value, e.g., 10^{-3}, we will consider it to be 0, which boils down to throwing away the corresponding sample.

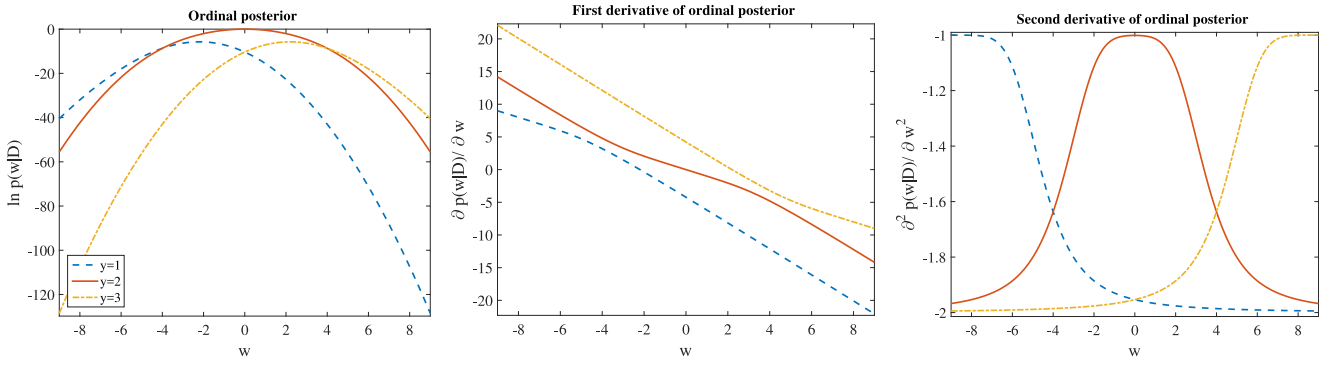


Fig. 1. The ordinal posterior and its first and second derivatives.

Note that $\Phi^T\mathbf{H}\Phi$ is a quadratic form and $\mathbf{A}$ is a diagonal matrix with positive diagonal elements, so

$$-\frac{\partial^2 \log p(\mathbf{w} \mid \mathbf{D})}{\partial \mathbf{w} \partial \mathbf{w}^T}$$

is a positive definite matrix, which implies that MAP estimation is a concave programming problem, with a global maximum.

Having found the MAP point $\mathbf{w}^*$, we use the Laplace method to approximate the posterior distribution by a Gaussian distribution $\mathcal{N}(\mathbf{w} \mid \mathbf{w}^*, \boldsymbol{\Sigma})$, where $\mathbf{w}^*$ and $\boldsymbol{\Sigma}$ are the mean and variance and computed as follows:

$$\boldsymbol{\Sigma} = (\mathbf{A} + \Phi^T\mathbf{H}\Phi)^{-1} \tag{10}$$

$$\mathbf{w}^* = \boldsymbol{\Sigma}\Phi^T\mathbf{H}\hat{\mathbf{t}}, \tag{11}$$

where $\hat{\mathbf{t}} = \mathbf{H}^{-1}\boldsymbol{\delta} + \Phi\mathbf{w}^*$.

Using a local Gaussian at the MAP point to represent the posterior distribution over weights is often considered as a weakness of the Bayesian treatment, especially for complex models. However, as pointed out by Tipping (2001), a log-concave posterior implies a much better accuracy and no heavier sparsity than L1-regularization. As we discussed above, the posterior of ISBOR has the feature of log-concavity. We report the plots of the log-posterior as well as its first and second order derivatives in Fig. 1 and see that $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ is monotonically decreasing w.r.t. w, while $\frac{\partial^2 \mathcal{L}}{\partial^2 \mathbf{w}}$ is always smaller than 0. So the MAP here is essentially a log-concave optimization problem, which implies that the Laplace approximation in ISBOR enjoys the same features of accuracy and sparsity as in the Relevance Vector Machine (RVM) (Tipping, 2001).

4. Hyper-parameter optimization

ISBOR uses various hyper-parameters, including $\boldsymbol{\alpha}$ in the prior estimation (Eq. (5)), the noise variance σ in Eq. (4), and the thresholds $[b_1, \ldots, b_r]$. In this section we detail how to learn these hyper-parameters.

4.1. Marginal likelihood

As a fully Bayesian framework, hyper-parameters are optimized by maximizing the posterior mode of hyper-parameters $p(\boldsymbol{\eta} \mid \mathbf{D}) \propto p(\mathbf{D} \mid \boldsymbol{\eta})p(\boldsymbol{\eta})$, where $\boldsymbol{\eta}$ contains all hyper-parameters. As we assume a non-informative Gamma hyper-prior, the optimization is equivalent to maximizing the marginal likelihood $p(\mathbf{D} \mid \boldsymbol{\eta})$, which is computed as $p(\mathbf{D} \mid \boldsymbol{\eta}) = \int p(\mathbf{D} \mid \mathbf{w}, \sigma)p(\mathbf{w} \mid \boldsymbol{\alpha})d\mathbf{w}$. As there is no closed form for this equation, again, we apply Laplace approximation and get the following approximations:

$$\begin{aligned} p(\mathbf{D} \mid \boldsymbol{\eta}) &= p(\mathbf{Y} \mid \mathbf{w}^*)p(\mathbf{w}^* \mid \boldsymbol{\alpha})(2\pi)^{n/2}\boldsymbol{\Sigma}^{1/2} \\ \ln p(\mathbf{D} \mid \boldsymbol{\eta}) &= \mathcal{L} - \frac{1}{2}\mathbf{w}^{*T}\mathbf{A}\mathbf{w}^* + \frac{1}{2}\ln|\mathbf{A}| + \frac{1}{2}\ln|\boldsymbol{\Sigma}|. \end{aligned} \tag{12}$$

In the rest of this section, we deal with the log-marginal likelihood, and maximize Eq. (12) with respect to each hyper-parameter.

4.2. Threshold and noise hyper-parameters

For the threshold hyper-parameters, we only need to determine $r-1$ values: b_1 and $[\Delta_2, \ldots, \Delta_{r-1}]$. Since we cannot compute these analytically, we exploit gradient descent (ascent, actually) to iteratively choose these parameters. The derivatives of the log-marginal likelihood, Eq. (12), with respect to b_1 and Δ_i, are computed as follows:

$$\frac{\partial \ln p(\boldsymbol{D} \mid \boldsymbol{\eta})}{\partial b} = -\boldsymbol{\delta}^*, \tag{13}$$

$$\frac{\partial \ln p(\boldsymbol{D} \mid \boldsymbol{\eta})}{\partial \Delta_i} = \begin{cases} -\delta_n^* & \text{if } y_n > i \\ \dfrac{1}{\sigma}\dfrac{\mathcal{N}(z_1; 0, 1)}{\Psi(z_1) - \Psi(z_2)} & \text{if } y_n = i \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Based on these two equations, we use gradient descent to search for proper thresholds.

For the noise term σ, setting the derivative

$$\frac{\ln p(\mathbf{D} \mid \boldsymbol{\eta})}{\sigma} = 0,$$

we obtain an update rule for the noise term:

$$\sigma^2 = \frac{\|\hat{\mathbf{t}} - \Phi\mathbf{w}\|^2}{N - \sum_n (1 - \alpha_n \Sigma_{nn})}, \tag{15}$$

where $\hat{\mathbf{t}} = \mathbf{H}^{-1}\boldsymbol{\delta} + \Phi\mathbf{w}^*$.

4.3. Fast marginal learning

We compute the contribution of the sparsity hyper-parameter $\boldsymbol{\alpha}$ to the marginal likelihood as follows:

$$\ln p(\mathbf{D} \mid \boldsymbol{\alpha}) = \mathcal{L} - \frac{1}{2}\ln|\mathbf{C}| - \frac{1}{2}\hat{t}\mathbf{C}^{-1}\hat{t}, \tag{16}$$

where we compute $\mathbf{C}$ as follows:

$$\begin{aligned} \mathbf{C} &= \mathbf{H}^{-1} + \Phi\mathbf{A}^{-1}\Phi^T \\ &= \mathbf{H}^{-1} + \sum_{n \neq j} \alpha_n \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T + \alpha_j^{-1}\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T. \end{aligned} \tag{17}$$

Since computing $\mathbf{C}$ requires matrix inversion, it is impractical to maximize it for large scale training sets. Fortunately, Tipping & Faul (2003) proposed a sequential way to maximize the marginal likelihood. We take this strategy and optimize $\boldsymbol{\alpha}$ as follows:

- First, we use the established matrix determinant and inverse identities (Petersen, Pedersen, et al., 2008) to compute the

determination and inverse of $\mathbf{C}$ as follows:

$$|\mathbf{C}| = |\mathbf{C}_{/j}| \, |\mathbf{I} + \alpha_j^{-1}\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T|$$
$$\mathbf{C}^{-1} = \mathbf{C}_{/j}^{-1} - \frac{\mathbf{C}_{/j}^{-1}\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\mathbf{C}_{/j}^{-1}}{\alpha_j + \boldsymbol{\phi}_j^T\mathbf{C}_{/j}^{-1}\boldsymbol{\phi}_j}, \tag{18}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{C}_{/j}$ denotes $\mathbf{C}$ without the contribution of the jth sample.

- Second, we define two auxiliary variables:

$$s_j = \phi_j^T C_{/j}^{-1}\phi_j, \; q_j = \phi_j^T C_{/j}^{-1}\hat{t}. \tag{19}$$

Combining Eqs. (16), (18) and (19), we isolate the contribution of sample j to the marginal likelihood as follows:

$$\ln p(\boldsymbol{D} \mid \alpha_j) = \frac{1}{2}[\ln\alpha_j - \ln|\alpha_j + s_j| + \frac{q_j^2}{s_j + \alpha_j}]. \tag{20}$$

For simplicity, we define $g(\alpha_j) = \ln p(\boldsymbol{D} \mid \alpha_j)$.

- However, we still need to compute the inverse of $\mathbf{C}_{/j}$ in Eq. (19). To speed up the computation, we define the follow auxiliary variables:

$$Q_j = \boldsymbol{\phi}_j^T\mathbf{C}^{-1}\hat{t} = \boldsymbol{\phi}_j^T\mathbf{H}\hat{t} - \boldsymbol{\phi}_j^T\mathbf{H}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{H}\hat{t}$$
$$S_j = \boldsymbol{\phi}_j^T\mathbf{C}^{-1}\boldsymbol{\phi}_j = \boldsymbol{\phi}_j^T\mathbf{H}\boldsymbol{\phi}_j - \boldsymbol{\phi}_j^T\mathbf{H}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{H}\boldsymbol{\phi}_j,$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{M\times M}$ is the covariance of the posterior distribution (Eq. (10)).[3] Then, we can compute

$$s_j = \frac{\alpha_j S_j}{\alpha_j - S_j} \text{ and } q_j = \frac{\alpha_j Q_j}{\alpha_j - S_j}.$$

- Finally, setting

$$\frac{\partial g(\alpha_j)}{\partial \alpha_j} = 0,$$

we get the closed form solution for α_j:

$$\alpha_j = \frac{s_j^2}{q_j^2 - s_j}. \tag{21}$$

Since $\alpha_j \geq 0$, the denominator of Eq. (21), denoted as $f_j = q_j^2 - s_j > 0$, which works as an important criterion for determining the relevant samples.

4.4. ISBOR

We summarize the pseudo-code of ISBOR in Algorithm 1. We provide brief comments on three ingredients. First, we initialize ISBOR (line 4) by randomly picking a sample from each category as the initial relevant samples. Based on these r samples, we initialize $\mathbf{Q}$, $\mathbf{S}$ and $\mathbf{f}$. On line 6, we compute the delta marginal likelihood for the samples not yet considered. As to the call to Estimate() (line 13), we update $\mathbf{w}$ based on Eq. (11); update $\boldsymbol{\alpha}$ based on Eq. (21); update $\mathbf{ml}$ based on Eq. (12) and use gradient search to update threshold $\mathbf{b}$ based on Eqs. (13) and (14).

4.5. Computational analysis

The maximization rule for marginal likelihood is based on the MAP estimate which, in Eq. (10), requires the inversion of a matrix with $\mathcal{O}(M^3)$ computational complexity and $\mathcal{O}(M^2)$ memory. However, as we constructively maximize the marginal likelihood,

[3] Because of the sparse assumption, $M \ll N$, and thus computing the inverse of $\boldsymbol{\Sigma}$ is much faster than that of $\mathbf{C}$.

Algorithm 1 Incremental Sparse Bayesian Ordinal Regression (ISBOR)

1: **Input: D** $= \{\mathbf{x}, \mathbf{y}\}$, θ, maxIts and minDelta.
2: **Output: w**, **b** and σ.
3: Φ = basis($\mathbf{x}$, θ);
4: $\mathbf{w}, \boldsymbol{\phi}, \boldsymbol{\alpha}, \sigma, \mathbf{b}, \mathbf{Q}, \mathbf{S}, \mathbf{f}$ = Initialize($\Phi, \mathbf{y}$);
5: **for** $i = 1, 2, \ldots,$ maxIts **do**
6: deltaML = $[g(\alpha_1), \ldots, g(\alpha_n)]$;
7: $\boldsymbol{\phi_n} \leftarrow$ max(deltaML);
8: **if** $\boldsymbol{\phi_n} \in \boldsymbol{\phi}$ and $f_n < 0$ **then**
9: $\{\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\phi}\} \leftarrow \{\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\phi}\} - \{w_n, \alpha_n, \phi_n\}$;
10: **else if** $f_n > 0$ **then**
11: $\{\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\phi}\} \leftarrow \{\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\phi}\} \cup \{w_n, \alpha_n, \phi_n\}$;
12: **end if**
13: $\mathbf{w}, \boldsymbol{\alpha}, \mathbf{b}, \mathbf{ml}$ = Estimate($\mathbf{w}, \boldsymbol{\alpha}, \mathbf{b}, \Phi, \boldsymbol{\phi}, \sigma$);
14: compute σ based on Eq. (15);
15: compute $\mathbf{Q}, \boldsymbol{S}$ based on Eq. (21);
16: compute $\boldsymbol{q}, \boldsymbol{s}, \boldsymbol{f}$;
17: **if** abs($\mathbf{ml} - \mathbf{ml}_{old}$) < minDelta **then**
18: break;
19: **end if**
20: $\mathbf{ml}_{old} = \mathbf{ml}$;
21: **end for**

$M \ll N$, first, we choose one sample from each category to initialize the algorithm; second, we benefit from the sparse learning, as the scale of M remains small (around a few dozen based on our experiments). In this case, matrix inversion is not the main computational bottle-neck for each iteration.

Although we apply an incremental strategy to train ISBOR, we have to compute the basis function matrix in the initialization step, which has $\mathcal{O}(N^2)$ computational complexity and $\mathcal{O}(N^2)$ memory. Combining these two parts, the total computational complexity of ISBOR is $\mathcal{O}(N^2 + M^3)$ and the memory complexity $\mathcal{O}(N^2)$. However, we should mention that the basis function matrix can be computed in the pre-training session, so the computational complexity is essentially $\mathcal{O}(N + M^3)$. For comparison, we report the computational and space complexity of SBOR and other state-of-the-art methods in Table 1. We see that ISBOR has the best computational complexity, and thus, ISBOR is more efficient than others, at least theoretically.

As computing the posterior covariance requires the inverse of the Hessian matrix, $(\mathbf{A} + \boldsymbol{\Phi}^T\mathbf{H}\boldsymbol{\Phi})^{-1}$, it is inevitable to encounter the singular values. Theoretically speaking, $\mathbf{H}$ and $\mathbf{A}$ are the diagonal matrices with positive elements, $\boldsymbol{\Phi}^T\mathbf{H}\boldsymbol{\Phi}$ is the quadratic form. However, there still exist singular problems, especially when some α are extremely large. In order to avoid ill-conditioning, we manually prune training samples with large α values.

4.6. Sparsity analysis

The simple Gaussian prior working as an L2-regularization in the posterior model leads to a non-sparse MAP estimate. However, with the Gamma hyper-prior, the real prior over $\mathbf{w}$ follows a Student's t distribution which is considered as a sparse prior with a sharp peak at 0 (Tipping, 2001, Section 5.1). During inference, we do not integrate out $\boldsymbol{\alpha}$, which implies that $\boldsymbol{\alpha}$ is the direct factor to sparsity, which in turn means that for irrelevant vectors the corresponding $\boldsymbol{\alpha}$ should be large. However, the learned $\boldsymbol{\alpha}$ in the sequential model are relatively small: we only add sample whose $\boldsymbol{\alpha}$ is essentially small to the model as the relevant sample. There is no reason to learn $\boldsymbol{\alpha}$ of samples excluded from the model, which has large values.

Table 1
Computational and space complexity of ordinal regression algorithms. N and M represent the number of training samples and the number of relevant and/or support samples, respectively.

	KDLOR/GPOR /SVOR/SBOR	ISVOR	ISBOR
Computational complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(2N + 8M^3)$	$\mathcal{O}(N + M^3)$
Space complexity	$\mathcal{O}(N^2)$	$\mathcal{O}(4N^2)$	$\mathcal{O}(N^2)$

Table 2
Benchmarks: Detailed information.

Dataset	# training	# test	# features	# categories
BS	468	157	4	3
SWD	750	250	10	4
Marketing	6,744	2249	74	9
Bank	8,000	50	8	5
Computer	8,092	100	12	5
CalHouse	20,490	150	8	5
Census	22,584	200	16	5

5. Experimental evaluation

Our experimental evaluation aims at addressing the following three research questions.

RQ1 Efficacy: Is the generalization performance of the proposed algorithm, ISBOR, comparable to other baselines?
RQ2 Efficiency: Does fast marginal analysis reduce ISBOR's computational complexity compared to baselines?
RQ3 Sparseness: Can ISBOR achieve the competitive predictions only based on a small subset of the training set?

5.1. Experimental design

The research questions listed above lead us to two experimental designs. The first involves a synthetic dataset to give us an understanding of the efficacy, effectiveness and sparsity. The second is on benchmark datasets, i.e., 7 widely used ordinal datasets to extensively evaluate the performance of ISBOR.

5.1.1. Datasets

Synthetic dataset. To create a synthetic dataset we follow the data-generating strategy in da Costa, Alonso, and Cardoso (2008). First, 21,000 two-dimensional points are sampled within the square area $[0, 10] \times [0, 10]$ under a uniform distribution. Second, each point is assigned a score by the function $f(\mathbf{x}) = 10(x_1 - 0.5)(x_2 - 0.5) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.5^2)$ acts as a Gaussian random noise. Finally, we choose six thresholds $\{-\infty, -60, -9, 15, 60, +\infty\}$, and each point is attached with a category by computing:

$$y = \underset{r \in \{1,2,3,4,5\}}{\arg\min}\ b_{r-1} \leq 10(x_1 - 0.5)(x_2 - 0.5) + \epsilon \leq b_r.$$

In this manner, we generate a five-category dataset and the numbers of data points assigned to each category are 4431, 4535, 3949, 3780 and 4305, respectively. We choose 10 different sizes of training sets: 1000, 2000, …, 10 000 and use the rest of the data as test sets. For each size training sets, we randomly generate 30 different partitions. Then, the experiments are conducted on all 30 partitions.

Benchmark datasets. We also compare ISBOR with five algorithms on seven benchmark datasets.[4] The details of the benchmark datasets are summarized in Table 2. Each benchmark dataset is randomly split into 20 partitions.

[4] http://www.uco.es/grupos/ayrna/ucobigfiles/datasets-orreview.zip.

5.1.2. Metrics

We use Mean Absolute Error (MAE) to measure the efficacy:

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|y_n - \hat{y}_n|,$$

where $\hat{y}_n$ is the predicted category. As for efficiency, we choose running time (in seconds) as the measurement.

5.1.3. Methods used for comparison

We choose KDLOR, GPOR, SVOR, SBOR and ISBOR discussed in the related work section as baselines. We use the ORCA package (Gutiérrez et al., 2016) (in MATLAB)[5] for KDLOR. The authors of SVOR and GPOR provide a publicly available implementation in C.[6] We use a MATLAB implementation of ISVOR shared by the authors. SBOR and ISBOR are implemented in MATLAB.

5.1.4. Settings and parameters

We choose the Gaussian RBF in Eq. (2) as the basis function for each algorithm. We initialize ISBOR by setting $\boldsymbol{\alpha} = 10^{-3}$, $\sigma = 1$.[7] We select the kernel width via 5-fold cross-validation on the training set within the values of $\theta \in \{10^{-2}, 10^{-1}, \ldots, 10\}$. GPOR automatically learns the hyper-parameters, which does not require any pre-selection process. For other methods, we follow the model selection process in Gutiérrez et al. (2016) and use a nested 5-fold cross-validation on the training set to search for the best hyper-parameters. Specifically, we choose $\theta \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$ for every algorithm. The additional regularization parameters of SVOR and ISVOR are chosen within the values of $c \in \{10^{-1}, \ldots, 10^{3}\}$. For KDLOR, we choose the regularization parameter within the range of $c \in \{0.1, 1, 10\}$, since the regularization parameter of KDLOR presents a different interpretation from the one in SVM. Additionally, KDLOR requires another singularity-avoiding parameter, which is chosen in the range of $u \in \{10^{-6}, 10^{-5}, \ldots, 10^{-1}\}$.

Cross-validation is conducted using MAE. That is, once the hyper-parameters with the lowest MAE are obtained, we apply them to the whole training set and then validate them on the test sets.

The experiments are run on a server with Intel(R) Xeon(R) CPU E5-2683 v3 2.00 GHz (16 Cores) and 32 Gigabyte.

5.2. Experimental results

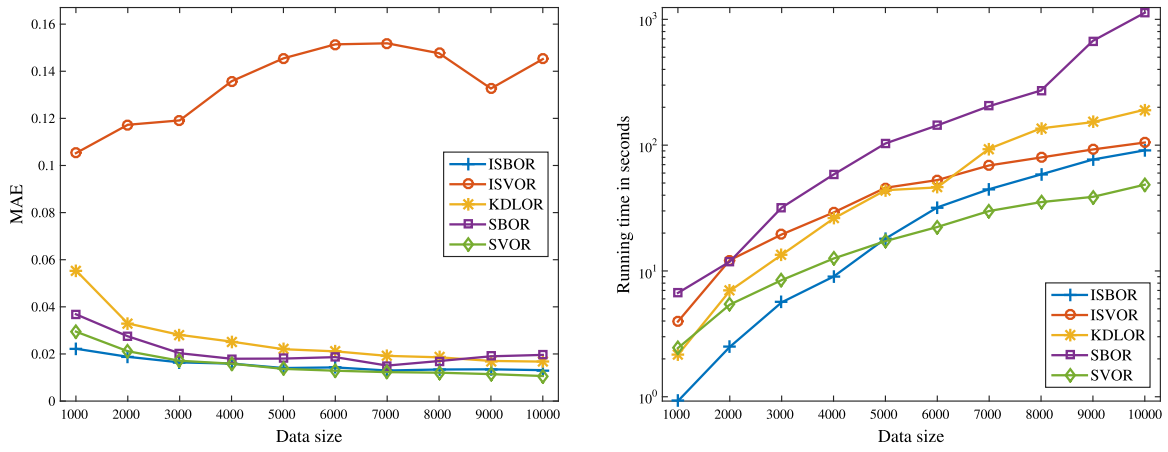
5.2.1. Efficacy

We begin by addressing **RQ1** concerning efficacy. We first consider the results on the synthetic dataset. Fig. 2(a) shows the performance in terms of MAE on the synthetic dataset. From the figure, we see that other than ISVOR, all the algorithms work well on the Synthetic datasets, in terms of efficacy. Specifically, ISBOR and SVOR are the two best performing algorithms. When the data sizes are larger than 5000, SVOR outperforms ISBOR, but the gaps are small.

[5] https://github.com/ayrna/orca.

[6] http://www.gatsby.ucl.ac.uk/~chuwei/#software.

[7] This is a heuristic setup inspired by Chu and Ghahramani (2005a), although the better way to choose the starting points is by trying different values and selecting the best combination.



(a) MAE. (b) Running time.

Fig. 2. MAE and running time of OR algorithms on the synthetic dataset.

Table 3
Benchmark results: MAE and running time. Standard deviations (of MAE) indicated in brackets. Failure to complete all runs in 24 h is indicated with '–'; best results are marked in **boldface**, second best in *italics*.

MAE	BS	SWD	Market	Bank	Computer	CalHouse	Census
KDLOR	0.17 (0.03)	0.58 (0.03)	1.60 (0.03)	0.21 (0.07)	0.39 (0.03)	**0.46 (0.05)**	0.63 (0.06)
GPOR	0.03 (0.02)	*0.41 (0.03)*	–	–	–	–	–
SVOR	**0.00 (0.00)**	**0.41 (0.03)**	**0.83 (0.01)**	*0.20 (0.06)*	0.40 (0.03)	0.63 (0.05)	–
SBOR	0.04 (0.06)	0.52 (0.06)	1.45 (0.03)	0.34 (0.19)	0.44 (0.12)	–	**0.60 (0.30)**
ISVOR	0.36 (0.53)	0.56 (0.04)	*1.20 (0.08)*	0.80 (0.10)	**0.36 (0.04)**	0.87 (0.07)	0.65 (0.06)
ISBOR	*0.02 (0.01)*	0.43 (0.02)	1.74 (0.05)	**0.19 (0.05)**	*0.37 (0.05)*	*0.50 (0.04)*	*0.63 (0.05)*
Running time	BS	SWD	Market	Bank	Computer	CalHouse	Census
KDLOR	**0.08**	**0.27**	159.92	97.27	96.86	1369.14	1696.18
GPOR	359.94	205.60	–	–	–	–	–
SVOR	**0.08**	0.83	44.49	932.59	2682.19	4350.30	–
SBOR	0.96	2.96	93.73	986.12	204.85	–	3713.62
ISVOR	1.53	*0.77*	*65.22*	**73.89**	**73.61**	*907.95*	*774.22*
ISBOR	0.64	1.35	**62.76**	*91.02*	*94.84*	**810.22**	**710.84**

Table 4
Wilcoxon tests for the MAE results obtained using the benchmark datasets and reported in Table 3.

Method	# wins	# draws	# losses
GPOR	4	6	25
SVOR	24	7	4
SBOR	11	12	12
KDLOR	11	8	16
ISVOR	11	11	13
ISBOR	17	10	8

Next, we turn to the benchmark datasets. The MAE scores are presented in Table 3 (top half). The results are averaged over 20 partitions.

To determine the significance of observed differences, we use the Wilcoxon test (Demšar, 2006; Wilcoxon, 1945) and compare the efficacy of each pair of algorithms. Since we compare 6 algorithms, there are 30 comparisons for each dataset in total. We choose the significance level $\alpha = 0.1$ and take the number of comparisons into account, and obtain the corrected significance level as $\alpha = 0.1/30 \approx 0.0033$. For each algorithm, we record the number of statistically significant wins, losses (or failures in finishing the training on time) and draws. The Wilcoxon test results are reported in Table 4.

Based on the top half of Tables 3 and 4, we find that SVOR is the best performing ordinal regression algorithm in terms of MAE. Specifically, SVOR wins 24 times out of 35 pair-wise comparisons. ISBOR, the second best performing algorithm, wins 17 comparisons. Because of the time limitation, GPOR fails to complete the experiments on 5 datasets and performs worse. The rest of algorithms perform similarly with each others and win 11 times.

To sum up, these results answer **RQ1** as follows: although SVOR has the best generalization performance, ISBOR outperforms other baselines and is comparable to SVOR.

5.2.2. Efficiency

We turn to **RQ2**. We report the running time of competing algorithms on the synthetic dataset with different data scales in Fig. 2(b). Generally, the implementations in C run much faster than those in pure MATLAB. To suppress this effect, we compare the running times on a logarithmic scale. We omit plotting the results of GPOR, because after running for 24 hours GPOR failed to complete any run on any partition.

Considering Fig. 2(b), when it comes to efficiency, ISBOR is faster than all algorithms except for SVOR, which is implemented in C. Comparing to SBOR, which can be regarded as the offline version of ISBOR, the gaps between ISBOR and SBOR are getting larger with the size of data increasing. On 10 000-size data, ISBOR is about 10 times faster than SBOR. These results demonstrate that incremental learning together with the sparseness assumption can accelerate the training speed of ISBOR. In summary, Fig. 2 shows that ISBOR can be an efficient ordinal regression algorithm while preserving a comparable prediction accuracy to SVOR.

From the bottom part of Table 3, we notice that on the small datasets, ISBOR does not show any advantages in running time.

Table 5
Relevant and support samples used on the benchmark datasets. Best results marked in **boldface**, second best in *italics*.

Dataset	SVOR	ISVOR	SBOR	ISBOR
BS	60.2	283.3	**9.0**	*17*
SWD	718.9	454.1	*104.8*	**58.5**
Marketing	3,756.0	10,185.1	*51.3*	**51.1**
Bank	5,685.1	8,128.5	**16.6**	*44.8*
Computer	3,373.1	7,739.5	*3,056.1*	**30.9**
CalHouse	12,788.3	*23,919.8*	–	**84.8**
Census	–	28,348.5	*1,001.0*	**73.0**

However, on the large datasets, ISBOR outperforms the baselines. Specifically, we can see a trend that the larger scale of the dataset is, the bigger the gaps between ISBOR and the batch algorithms are. This trend provides an answer to **RQ2**: the incremental setting makes ISBOR a faster OR algorithm.

5.2.3. Sparseness

Finally, we address **RQ3**. Since GPOR and KDLOR make predictions based on all training samples, in Table 5, we only report the number of support or relevant samples of SVOR, ISVOR, SBOR and ISBOR so as to answer the sparseness question (**RQ3**).

Analyzing Table 5, we notice that the sparse Bayes based SBOR and ISBOR employ much smaller numbers of training samples to make predictions than the SVM-based SVOR and ISVOR.[8] Among the seven benchmark datasets, ISBOR wins 5 times and SBOR wins 2 times, which supports our claim that ISBOR is a parsimonious ordinal regression algorithm and can make effective predictions based on a small subset of the training set. This finding answers **RQ3** on sparseness.

6. Conclusion

We have presented a novel incremental ordinal regression algorithm within an efficient sparse Bayesian learning framework. Instead of processing the whole training set in one go, the proposed algorithm can incrementally learn from representations of training samples and has linear computational complexity in the training data size. Our empirical results show that Incremental Sparse Bayesian Ordinal Regression (ISBOR) is comparable or superior to state-of-the-art OR algorithms based on basis functions in terms of efficacy, efficiency and sparseness.

We hope that this work paves the way for research into large-scale ordinal regression. We believe that the design of ISBOR can be improved in multiple directions. From a Bayesian viewpoint, a more elegant way to optimize the hyper-parameters would be to maximize $p(\boldsymbol{\eta} \mid \mathbf{D})$ rather than $p(\mathbf{D} \mid \boldsymbol{\eta})$ with additional hyper-assumptions. This is achievable via other approximation inference methods like variational Bayes and expectation propagation (Bishop, 2006, Chapter 10). From an application view, we can equip ISBOR with other sparse Bayesian architectures and adapt it to other problems like semi-supervised learning (Pérez-Ortiz et al., 2016; Srijith et al., 2013; Xiao, Liu, & Hao, 2016) and feature selection (Jiang, Li, Chen, Yao, & de Rijke, 2016; Li & Chen, 2014). From a ranking viewpoint, higher positions are more important. So far, ISBOR ignores pair-wise preferences and considers each position equally important, which amounts to a point-wise approach. Another promising future direction, therefore, is to take pair-wise position information into account and apply ISBOR to ranking problems.

[8] Notice how ISVOR uses more samples than the ground truth provides due to binary decomposition, as explained in Section 2.

Code and data

To facilitate reproducibility of the results in this paper, we are sharing the code and the data used to run the experiments in this paper at https://github.com/chang-li/SBOR.

Acknowledgments

We thank our anonymous reviewers for their valuable feedback and suggestions.

This research was partially supported by Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, the China Scholarship Council, the Criteo Faculty Research Award program, Elsevier, the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement no. 312827 (VOX-Pol), the Google Faculty Research Awards program, the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nos. CI-14-25, 652.002.001, 612.001.551, 652.001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
Chang, X., Zheng, Q., & Lin, P. (2009). Ordinal regression with sparse Bayesian. In *International conference on intelligent computing* (pp. 591–599). Springer.
Chu, W., & Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *Journal of Machine Learning Research (JMLR)*, *6*(Jul), 1019–1041.
Chu, W., & Ghahramani, Z. (2005). Preference learning with Gaussian processes. In *Proceedings of the 22nd international conference on machine learning* (pp. 137–144). ACM.
Chu, W., & Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation*, *19*(3), 792–815.
da Costa, J. F. P., Alonso, H., & Cardoso, J. S. (2008). The unimodal model for the classification of ordinal data. *Neural Networks*, *21*(1), 78–91.
Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research (JMLR)*, *7*(Jan), 1–30.
Gu, B., Sheng, V. S., Tay, K. Y., Romano, W., & Li, S. (2015). Incremental support vector learning for ordinal regression. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(7), 1403–1416.
Gu, B., Wang, J. D., Yu, Y. C., Zheng, G. S., Huang, Y. F., & Xu, T. (2012). Accurate on-line ν-support vector learning. *Neural Networks*, *27*, 51–59.
Gutiérrez, P. A., Perez-Ortiz, M., Sanchez-Monedero, J., Fernández-Navarro, F., & Hervas-Martinez, C. (2016). Ordinal regression methods: Survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, *28*(1), 127–146.
Gutiérrez, P. A., Tiňo, P., & Hervás-Martínez, C. (2014). Ordinal regression neural networks based on concentric hyperspheres. *Neural Networks*, *59*, 51–60.
Hu, J., & Li, P. (2018). Collaborative filtering via additive ordinal regression. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 243–251). New York, NY, USA: ACM.
Jiang, B., Li, C., Chen, H., Yao, X., & de Rijke, M. (2016). Probabilistic Feature Selection and Classification Vector Machine, arXiv preprint arXiv:1609.05486.
Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, *13*(3), 637–649.
Li, C., & Chen, H. (2014). Sparse Bayesian approach for feature selection. In *2014 IEEE symposium on computational intelligence in big data* (pp. 1–7).
Liu, T. Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, *3*(3), 225–331.
MacKay, D. J. (1992). The evidence framework applied to classification networks. *Neural Computation*, *4*(5), 720–736.
Minka, T. P. (2001). *A family of algorithms for approximate bayesian inference* (Ph.D. thesis), Cambridge, MA, USA: Massachusetts Institute of Technology.
Niu, Z., Zhou, M., Wang, L., Gao, X., & Hua, G. (2016). Ordinal regression with multiple output CNN for age estimation. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 4920–4928).
Pérez-Ortiz, M., Gutiérrez, P. A., Carbonero-Ruz, M., & Hervás-Martínez, C. (2016). Semi-supervised learning for ordinal Kernel discriminant analysis. *Neural Networks*, *84*, 57–66.
Petersen, K. B., Pedersen, M. S., et al. (2008). The matrix cookbook. *Technical University of Denmark*, *7*(15), 510.

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning* (pp. 63–71). Springer.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Srijith, P., Shevade, S., & Sundararajan, S. (2012). A probabilistic least squares approach to ordinal regression. In *Australasian joint conference on artificial intelligence* (pp. 683–694). Springer.

Srijith, P., Shevade, S., & Sundararajan, S. (2012). Validation-based sparse Gaussian processes for ordinal regression. In *International conference on neural information processing* (pp. 409–416). Springer.

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

Sun, B. Y., Li, J., Wu, D. D., Zhang, X. M., & Li, W. B. (2010). Kernel discriminant learning for ordinal regression. *IEEE Transactions on Knowledge and Data Engineering*, *22*(6), 906–910.

Tang, F., & Tiňo, P. (2017). Ordinal regression based on learning vector quantization. *Neural Networks*, *93*, 76–88.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research (JMLR)*, *1*(Jun), 211–244.

Tipping, M. E., & Faul, A. C. (2003). Fast marginal likelihood maximisation for sparse Bayesian models. In *AISTATS*.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, *10*(5), 988–999.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83.

Xiao, Y., Liu, B., & Hao, Z. (2016). A maximum margin approach for semisupervised ordinal regression clustering. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(5), 1003–1019.

Xiao, Y., Liu, B., & Hao, Z. (2017). Multiple-instance ordinal regression. *IEEE Transactions on Neural Networks and Learning Systems*, *PP*(99), 1–16.

Ypma, T. J. (1995). Historical development of the Newton–Raphson method. *SIAM Review*, *37*(4), 531–551.