Machine Learning (2021) 110:2729–2762
https://doi.org/10.1007/s10994-021-06010-w

Ordinal regression with explainable distance metric learning based on ordered sequences

Juan Luis Suárez[1] · **Salvador García**[1] · **Francisco Herrera**[1]

Received: 1 October 2020 / Revised: 22 May 2021 / Accepted: 25 May 2021 /
Published online: 7 July 2021
© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2021

Abstract
The purpose of this paper is to introduce a new distance metric learning algorithm for ordinal regression. Ordinal regression addresses the problem of predicting classes for which there is a natural ordering, but the real distances between classes are unknown. Since ordinal regression walks a fine line between standard regression and classification, it is a common pitfall to either apply a regression-like numerical treatment of variables or underrate the ordinal information applying nominal classification techniques. On a different note, distance metric learning is a discipline that has proven to be very useful when improving distance-based algorithms such as the nearest neighbors classifier. In addition, an appropriate distance can enhance the explainability of this model. In our study we propose an ordinal approach to learning a distance, called *chain maximizing ordinal metric learning*. It is based on the maximization of ordered sequences in local neighborhoods of the data. This approach takes into account all the ordinal information in the data without making use of any of the two extremes of classification or regression, and it is able to adapt to data for which the class separations are not clear. We also show how to extend the algorithm to learn in a non-linear setup using kernel functions. We have tested our algorithm on several ordinal regression problems, showing a high performance under the usual evaluation metrics in this domain. Results are verified through Bayesian non-parametric testing. Finally, we explore the capabilities of our algorithm in terms of explainability using the case-based reasoning approach. We show these capabilities empirically on two different datasets, experiencing significant improvements over the case-based reasoning with the traditional Euclidean nearest neighbors.

Keywords Distance metric learning · Ordinal regression · Nearest neighbors

Our work has been supported by the research project TIN2017-89517-P and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

Editors: João Gama, Alípio Jorge, Salvador García.

✉ Juan Luis Suárez
jlsuarezdiaz@decsai.ugr.es

Extended author information available on the last page of the article

1 Introduction

Ordinal regression is a machine learning task which aims to predict variables that take values in an ordered and non-numerical set. It is an interesting research topic with applications in many areas, such as weather forecasting (Guijo-Rubio et al., 2020), medicine (Kuráňová, 2016; Sánchez-Monedero et al., 2018; Beckham & Pal, 2017), psychology (Bürkner & Vuorre, 2019), social media mining (Chakraborty & Church, 2020) or computer vision (Antoniuk et al., 2016; Fu et al., 2018; Liu et al., 2017). This type of problem arises in many situations with human labelers, since they usually prefer to quantify data using categorical values rather than continuous values. The key element that differentiates ordinal regression from standard classification and regression is the presence of an order relation among the labels and of unquantifiable differences between two consecutive labels. This makes it necessary to adapt the traditional classification and regression techniques or to design new approaches to deal with ordinal regression problems in an appropriate way (Gutierrez et al., 2016).

Since ordinal regression walks a fine line between classification and regression, it was traditionally common to approach the problem from one of these naive perspectives. This could lead to suboptimal approaches, as they might not take adequate advantage of the ordinal information in the data. Over time, the problem of ordinal regression has become increasingly important, due to its presence in many real problems, and new methods have been developed. Frank and Hall (2001) propose to decompose the ordinal regression problem into binary subproblems considering the inequalities with each of the possible classes. For each subproblem a binary classifier is trained and finally the prediction probabilities are aggregated to obtain the final ordinal class. Cheng et al. (2008) propose a generalization of the perceptron for ordinal regression, modifying the output layer to handle ordinal labels. Chu and Keerthi (2007) extend support vector machines for ordinal regression problems by imposing constraints on the classes, both explicit and implicit. Cardoso and da Costa (2007) propose an original method to address this problem, which consists in adding additional dimensions to the data and replicating them in these dimensions. In this new scenario, the problem is transformed into a binary problem where the binary label assigned to each dimension will depend on the original ordinal class. The problem then can be solved using a binary classifier. Lin and Li (2009) extend the AdaBoost ensemble to adapt it to ordinal classes. More recently, Shi et al. (2019) propose an encoding and a kernel version of extreme learning machines to deal with ordinal regression problems, and Halbersberg et al. (2020) extend the Bayesian network classifiers by maximizing several metrics that better adapt to imbalanced and ordinal problems.

Learning using similarities between data is quite effective and adaptable to a wide variety of problems. It is inspired by the human ability to detect similarities among different objects, and it is one of the oldest practices carried out in machine learning (Cover and Hart 1967). Since similarity-based learning is based on human reasoning, it also allows us to develop explainable models for which their learned knowledge can be interpreted (Arrieta et al., 2020; Belle & Papantonis, 2020). Similarity-based approaches need to establish a similarity measure, or equivalently, a distance metric, among the data. Typically, standard distances, such as the Euclidean distance, are used for this purpose. However, standard distances may not fit our data as well as a distance that has been learned from the dataset itself. Learning a suitable distance that allows us to facilitate a subsequent knowledge extraction is the task of distance metric learning (Suárez et al., 2021). Distance metric learning has proven to be highly effective for

many different problems in machine learning, such as multi-dimensional classification (Ma & Chen, 2018) and multi-output learning (Liu et al., 2018).

Distance metric learning can be really useful for ordinal regression, since the learned distances can help to capture the ordinal information of the data. By learning a distance, we can get similar data to have similar labels as well, according to the order relation among the classes. In this way, the subsequent similarity learning will be more effective.

Several proposals to learn distances for ordinal regression problems have been formulated (Xiao et al., 2009; Fouad & Tiňo, 2013), but they use the differences between classes internally, which may make them more appropriate for standard regression problems.

A more recent proposal (Nguyen et al., 2018) attempts to overcome this drawback by drawing inspiration from one of the most popular distance metric learning algorithms, LMNN (Weinberger & Saul, 2009). This algorithm relies on the large margin principle to locally bring same-class data closer together, while preventing data from other classes from getting close to a specific limit. The extension to the ordinal case is made by adding further constraints so that, for a given sample, the samples with the most distant classes in the output space cannot be brought as close together in the input space as others with closer labels. This algorithm also supports a kernel version that is able to learn a distance in very high dimensional spaces. We will refer to the linear and kernel version of this proposal as LODML and KODML, respectively. However, seeking an airtight separation of classes, like this algorithm does, may be undesirable in many ordinal regression problems. Ordinal datasets are usually small and suffer from a high degree of subjectivity in their labeling (Agresti, 2010). Examples of this subjectivity are datasets that are generated from *likert scales* (Joshi et al., 2015) or pain intensity scales for medical purposes. In these cases the perception of the classes may be different for each labeler. This results in heterogeneous datasets with unclear class boundaries. The underlying order is what actually contains the information in the dataset, rather than the true classes assigned to each sample.

from which the existing distance metric learning algorithms tend to suffer when applied to ordinal regression are:

- The quantification of differences between classes during the operation of the algorithms.
- The assumption that classes in ordinal datasets can be easily separated.

In this paper we propose a new distance metric learning algorithm for ordinal regression that overcomes these drawbacks. Our algorithm, which we have called *chain maximizing ordinal metric learning* (CMOML), is based on the premise that no matter how heterogeneous our dataset is, a distance-based classifier can better predict a sample when labels gradually become more and more different as we move away from the sample in any direction. In order to support our proposal, Fig. 1 illustrates a good data layout for distance-based ordinal regression. This figure is a snapshot of a bidimensional dataset around the point $x \in \mathbb{R}^2$. We can see here that, although there is not a clear boundary between each class in the dataset, when we move away from x in most directions we notice that the labels begin to increase (marked with the red dashed arrows) or decrease (marked with the blue dashed arrows) gradually. Therefore, a distance-based classifier is expected to perform well around x. CMOML is based on this idea in order to transform the data such that a good arrangement for a subsequent distance-based learning can be achieved. If many same-class samples are found, they will move closer to one another. If there is more variety, an optimal local sorting will be sought like in Fig. 1.

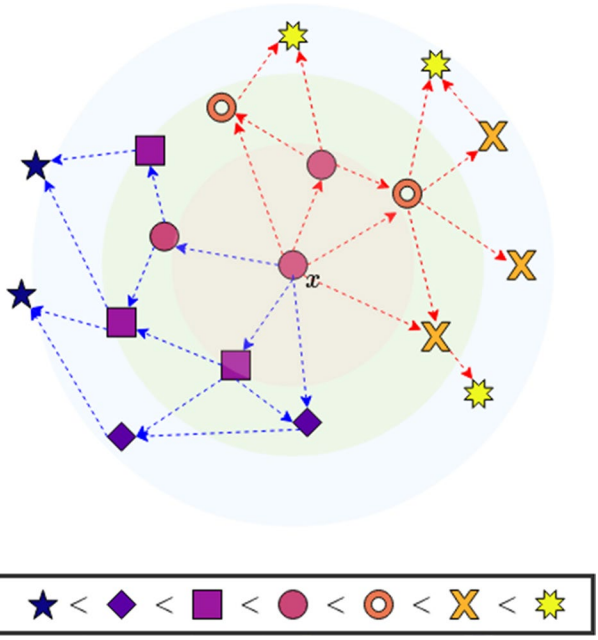

Fig. 1 A snapshot of a 2D ordinal dataset around the point x. When we move away from x in most directions, the labels gradually increase or decrease. This benefits similarity-based or distance-based classification around x (Color figure online)

CMOML looks for a distance that respects the local ordinal trend in our dataset as much as possible. This is achieved by introducing the concept of ordered sequences. For each sample we take a neighborhood from which we construct two types of sequences around the sample: sequences of samples, in the input space, and the corresponding sequences of labels, in the output space. An order relation can be established in both types of sequences, and when a pair of sequences is ordered in both the input and output space, we will refer to that pair as a chain. The goal of CMOML is to maximize the number of chains in the dataset. The configuration of the algorithm also allows it to be applied in high dimensional spaces with the support of the kernel functions. We will refer to the kernel version of CMOML as KCMOML.

The design of CMOML makes it possible to solve the problems of the previous proposals, since (a) the labels are not treated numerically, as the sequences will only consider the relative positions between them, and (b) the sequential approach fits the structure of the data regardless of the quality of the class separation.

In order to analyze the time complexity of CMOML we have performed a computational analysis of the algorithm, which includes a comparison with the time complexity of the state-of-the-art of metric learning for ordinal regression. The analysis shows that the compared algorithms have similar time complexities and, due to the nature of CMOML, it can highly benefit from parallelization.

To evaluate the performance of CMOML we have carried out several experiments on datasets for ordinal regression. The results, supported by a Bayesian statistical analysis, show that CMOML is an outstanding algorithm within the family of distance metric learning algorithms for ordinal regression, in addition to being competitive with the ordinal regression state-of-the-art.

CMOML also offers benefits in terms of explainability that the compared algorithms lack. The case-based reasoning approach (Lamy et al., 2019) allows nearest

neighbors-based algorithms to show their learned knowledge in an understandable way (Belle & Papantonis, 2020; Arrieta et al., 2020). By combining nearest neighbors with the distance that CMOML learns, we can make the neighbors we obtain much more intuitive to our human reasoning. We will show this empirically. To do this, we will analyze on two different datasets how the nearest neighbors behave with the Euclidean distance and with the distance learned by CMOML. Additionally we will see that CMOML can perform dimensionality reduction and we will explore its benefits in terms of understandability.

Our paper is organized as follows. Section 2 describes the distance metric learning and ordinal regression problems. Section 3 shows our proposal of distance metric learning for ordinal regression. Section 4 describes the experiments developed to evaluate the performance of our algorithm, and the results obtained. Section 5 shows how to combine CMOML and case-based reasoning to obtain explainable models and discusses the benefits of this approach on two different datasets. Finally, Sect. 6 ends with the concluding remarks.

2 Background

In this section we will describe the problems and tools that will be used throughout the paper.

2.1 Distance metric learning

Distance metric learning (Suárez et al., 2021) is a discipline of machine learning that aims to learn distances from the data, where distance refers to a map $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, with $\mathcal{X}$ a non-empty set, satisfying the following conditions:

1. Coincidence: $d(x, y) = 0 \iff x = y$, for every $x, y \in \mathcal{X}$.
2. Symmetry: $d(x, y) = d(y, x)$, for every $x, y \in \mathcal{X}$.
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$, for every $x, y, z \in \mathcal{X}$.

Distance metric learning frequently focuses on learning Mahalanobis distances, since they are parameterized by matrices, and therefore they are easy to handle from a computational perspective. Given a positive semidefinite (also called metric) matrix M of dimension d, the Mahalanobis distance associated with M, for each $x, y \in \mathbb{R}^d$, is given by

$$d_M(x, y) = \sqrt{(x - y)^T M (x - y)}.$$

Since every metric matrix M can be decomposed as $M = L^T L$, where L is a matrix with the same number of columns as M, and verifying that $d_M(x, y) = \|L(x - y)\|_2$, a Mahalanobis distance can also be understood as the Euclidean distance after applying the linear map defined by the matrix L. Thus, distance metric learning comes down to learning a metric matrix M or to learning a linear map matrix L. Both approaches are equally valid, and each one has different advantages. For instance, learning M usually leads to convex optimization problems, while learning L can be used to guarantee a dimensionality reduction with no additional cost.

2.2 Ordinal regression and similarity approaches

Ordinal regression (Gutierrez et al., 2016) is a machine learning problem consisting in, similarly to classification and regression, predicting the output label $y \in \mathcal{Y}$ of an input vector $x \in \mathbb{R}^d$, given a training set of labeled samples $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$, with $x_1, \dots, x_N \in \mathbb{R}^d$ and $y_1, \dots, y_N \in \mathcal{Y}$. What differentiates ordinal regression from classification and regression is the nature of the set $\mathcal{Y}$. $\mathcal{Y}$ is a finite set $\mathcal{Y} = \{l_1, \dots, l_C\}$ of length $C \geq 3$, over which an order relation $<$ is defined so that $l_i < l_{i+1}$, for $i = 1, \dots, C-1$. In addition, although the values in $\mathcal{Y}$ are ordered, their differences are not quantifiable, that is, the operations $l_i - l_j$ are not defined, for any $i, j \in 1, \dots, C$. Therefore, the exclusive information that $\mathcal{Y}$ provides in ordinal regression problems is the relative positions between the labels. It is often common to represent $\mathcal{Y}$ as $\mathcal{Y} = \{1, \dots, C\}$ together with the order relation of the natural numbers, always taking into account that the differences between the values in $\mathcal{Y}$ should not be used in this problem.

In ordinal regression, committing a prediction error with a class close to the real one in $\mathcal{Y}$ is not as serious as committing an error predicting the class furthest away from the true one. Therefore, both the algorithms and the evaluation metrics have to be adapted to face this problem and evaluate their solutions adequately. For this purpose, traditional methods such as support vector machines have been adapted to this problem by imposing constraints in order to respect the ordering of the classes (Chu & Keerthi, 2007; Gu et al., 2020), or by reducing to binary problems in an appropriate way (Lin & Li, 2012). Other proposals try to adapt conventional loss or gain functions in order to handle ordinal data in the best possible way. These functions have been tested with algorithms such as decision trees (Singer et al., 2020), Bayesian networks (Halbersberg et al., 2020) or extreme learning machines (Shi et al., 2019), as well as with gradient-based learners (Fathony et al., 2017; Mensch et al., 2019), such as back-propagation neural networks. Finally, deep learning has also taken a leading role in this problem, with proposals that are mainly applied in fields related to computer vision (Vargas et al., 2020; Beckham & Pal, 2017; Fu et al., 2018).

In the context of similarity-based learning, the *k-nearest neighbors* (k-NN) approach (Cover & Hart, 1967) can be easily extended to the ordinal case. To do this, we have to consider the general aggregation function for a k-NN algorithm (Dudani, 1976; Calvo & Beliakov, 2010), given by

$$f(x) = \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^{k} w_j p(y_{i_j}, y),$$

where x is the sample to be predicted, $i_j, j = 1, \dots, k$ are the indices of the k-nearest neighbors of x in the training set, w_j are weights to be considered for each neighbor, and p is a penalty function that measures how wrong it might be to label x with each of the classes, according to the information given for each of the neighbors. Observe that if all the weights are equal and $p(y, y^*) = [\![y \neq y^*]\!]$, f becomes the classic majority-vote aggregation function of the k-NN (here, $[\![\cdot]\!]$ denotes the indicator function for the condition inside the brackets). When the labels are ordinal, we can use additional penalty functions that take into account the order relation between the labels. According to Tang et al. (2020), one of the most popular penalty functions is the L_1 penalty[1], given by $p(y, y^*) = |y - y^*|$. This penalty function

[1] Observe that, although this penalty function is considering the differences between the labels, their numerical values do not influence the final value of the aggregation function. In fact, the median or weighted-median resulting from the aggregation does not depend on the differences between the labels, but depends only on the weights w_j and the relative positions between the labels of the k-nearest neighbors.

leads to the *median-vote* or *weighted-median-vote* k-NN approaches, both of them being immediate extensions of this algorithm to ordinal regression.

In the context of distance metric learning for ordinal regression, Nguyen et al. (2018) recently proposed the distance metric learning approach LODML (*linear ordinal distance metric learning*). This approach is based on the classic *large margin nearest neighbors* (LMNN) algorithm for distance metric learning. LMNN searches for a distance that brings each sample as close as possible to a predefined set of same-class *target neighbors* while keeping data from other classes out of a *large margin* defined by the target neighbors. This distance can be obtained by optimizing the following objective function, defined for a positive semidefinite matrix M

$$\begin{aligned} & f_L(M) = \alpha \operatorname{tr}(M) + \sum_{(i,j,l)\in R_1} \xi_{ijl} \\ s.t. \; & : d_M^2(x_i, x_l) - d_M^2(x_i, x_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0 \\ & M \geq 0. \end{aligned}$$

In this function, the first term is a regularization term and ξ_{ijl} slack variables that measure to what degree the margins have been violated. The set R_1 is the set of triplets (i, j, l) such as x_j is a target neighbor of x_i and x_l is a sample with a different class to x_i. The first constraint is the *large margin constraint*, which enforces that, for each triplet $(i, j, l) \in R_1$, x_i and x_j are close, and x_l does not invade the large margin defined by x_j around x_i, as long as it is feasible.

The extension to the ordinal case that LODML performs is done by replacing the set of triplets R_1 with a new set $R = R_1 \cup R_2 \cup R_3$, where R_1 is the previous set of triplets, and R_2 and R_3 include triplets (i, j, l) so that $y_l < y_j < y_i$ or $y_i < y_j < y_l$, respectively, thus forcing the large margin criterion to also be applied with all the possible combinations of ordered classes. The algorithm can be extended to non-linear metric learning using kernel functions. The kernel version is referred to as *kernel ordinal distance metric learning* (KODML).

3 Chain maximizing ordinal metric learning

In this section we will describe the CMOML algorithm. This algorithm looks for a distance for which the dataset has a high number of chains. As such, it achieves an optimal configuration for the data, since close samples according to the distance obtained will also have close labels.

First, we will explain in detail the concepts needed to understand the algorithm, Then, we will show the objective function and how to proceed with its calculation. Afterwards, we will demonstrate how to extend the algorithm to high dimensional spaces using kernels. We will conclude the section by performing a complexity analysis of the proposed methods.

3.1 Preliminary definitions

Suppose that our training set is given by $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$, with corresponding labels $y_1, \ldots, y_N \in \mathcal{Y} = \{1, \ldots, C\}$. As already mentioned, given $x_i \in \mathcal{X}$, the goal of our

algorithm is to increase the difference between the labels as we move away from x_i in any direction, for each $x_i \in \mathcal{X}$. To do this we introduce several concepts. First, we will work with the distance defined by a linear map $L : \mathbb{R}^d \to \mathbb{R}^{d'}$, with $d' \leq d$. The value of d' is the desired dimension of our dataset in the transformed space, so we are able to perform a dimensionality reduction with this setup. The associated matrix, which we will also refer to as L, is of dimension $d' \times d$. A common value for d' is to use $d' = d$, so that the transformed data remains in the input space. If the transformation learned in this way is not full-rank, that is, all the elements in $\mathbb{R}^d$ are mapped to a subspace of dimension lower than d, this dimension can be used as a new value for d' to ensure that no information from the full-rank method is lost. Another possibility is to estimate d' from specialized dimensionality reduction methods. For example, to set d' according to the reconstruction errors provided by *principal components analysis* (Jolliffe, 2002), up to the amount of reconstruction error we are willing to admit for our data.

Then, for each $x_i \in \mathcal{X}$ we define a *neighborhood* $\mathcal{U}_L(x_i)$, which consists of the K_i nearest neighbors of x_i (including x_i), according to the distance determined by L. $K_i \in \mathbb{N}$ denotes the size of the neighborhood around x_i, and can be either a fixed value for every $i \in \{1, \dots, N\}$, a value estimated to make equal-radius neighborhoods that better adapts to the data densities, or any value that can be established with any kind of prior information. We will consider these neighbors as already projected onto the transformed space, that is, the elements of $\mathcal{U}_L(x_i)$ will be of the form Lx_j, with $j \in \{1, \dots, N\}$. We also fix $\kappa \in \mathbb{N}$, which will be the length of the sequences that we define below.

Definition 1 Let σ be an injective mapping from the set $\{1, \dots, \kappa\}$ to the set of the indices of the samples that belong to $U_L(x_i)$. A *sequence (of samples)* around $x_i \in \mathcal{X}$ (with respect to L) is a κ-tuple $(Lx_{\sigma(1)}, \dots, Lx_{\sigma(k)})$, with $Lx_{\sigma(1)}, \dots, Lx_{\sigma(\kappa)} \in \mathcal{U}_L(x_i)$. We refer to the tuple $(y_{\sigma(1)}, \dots, y_{\sigma(\kappa)})$ as the corresponding *sequence of labels*.

In other words, sequences of samples are successions of κ items taken from $U_L(x_i)$ without replacement, and their labels in the same order of choice form the corresponding sequence of labels. Since we are working on an ordinal regression problem, there is a natural order relation among the labels. We will say that a sequence of labels $(y_{\sigma(1)}, \dots, y_{\sigma(\kappa)})$ around x_i is *ordered* if either $y_i \leq y_{\sigma(1)} \leq \cdots \leq y_{\sigma(\kappa)}$ or $y_i \geq y_{\sigma(1)} \geq \dots y_{\sigma(\kappa)}$.

The next step is to establish a relation on the sequences of samples so that they can be paired with the corresponding order in the sequences of labels. As we are interested in increasing the difference between the labels while we are moving away from x_i, we define this (pre-)order relation as follows.

Definition 2 An *ordered sequence of samples* around $x_i \in \mathcal{X}$ (with respect to L) is a sequence of samples $(Lx_{\sigma(1)}, \dots, Lx_{\sigma(\kappa)})$ verifying that

$$\|Lx_{\sigma(1)} - Lx_i\| \leq \|Lx_{\sigma(2)} - Lx_i\| \leq \cdots \leq \|Lx_{\sigma(\kappa)} - Lx_i\|.$$

Definition 3 A *chain* around $x_i \in \mathcal{X}$ (with respect to L) is an ordered sequence of labels $(y_{\sigma(1)}, \dots, y_{\sigma(\kappa)})$ associated with an ordered sequence of samples $(Lx_{\sigma(1)}, \dots, Lx_{\sigma(\kappa)})$. We denote the total number of chains around x_i (with respect to L) as $S_L(x_i)$.

In short, chains represent sequences that are ordered in both the label space and the sample space. Observe that, if we assume that there are no points in $U_L(x_i)$ at the exactly same distance from x_i we have that

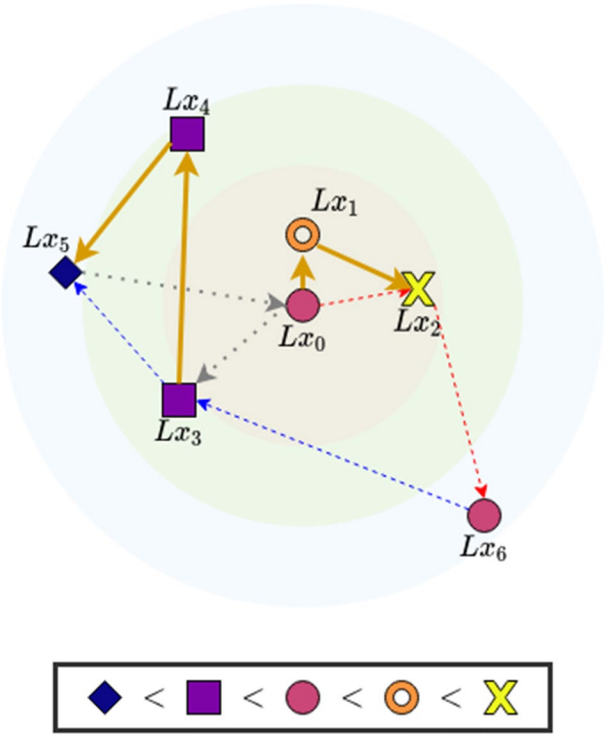

Fig. 2 A neighborhood of the sample Lx_0, for L of length $K_0 = 7$, with sequences of samples and labels (gray dotted arrows), ordered sequences of samples (red dashed arrows), ordered sequences of labels (blue dashed arrows) and chains (golden solid arrows) of length $\kappa = 3$ (Color figure online)

$$S_L(x_i) \leq \binom{K_i}{\kappa}, \tag{1}$$

since, as there is a unique ordering of the samples under this assumption, the total number of ordered sequences of samples is the total number of non-repeating combinations of κ items taken from $U_L(x_i)$, and $S_L(x_i)$ attains its maximum when every ordered sequence of samples is a chain. The case with equal distances will be discussed later on. Let us illustrate these concepts with an example.

Example 1 Suppose that we set $K_0 = 7, \kappa = 3$, and, for a linear map $L : \mathbb{R}^d \to \mathbb{R}^2$, we have a snapshot of a 2D ordinal dataset around the point Lx_0 as in Fig. 2. Under these assumptions we have that $U_L(x) = \{Lx_0, Lx_1, Lx_2, Lx_3, Lx_4, Lx_5, Lx_6\}$, and the distance orders in the neighborhood verify that $\|Lx_j - Lx\| < \|Lx_{j+1} - Lx\|$ for $j = 0, \ldots, 6$. The labels corresponding to each of the samples in $U_L(x_0)$ are: $y_0 =$ ●, $y_1 =$ ◎, $y_2 =$ X, $y_3 =$ ■, $y_4 =$ ■, $y_5 =$ ◆ and $y_6 =$ ●. We have that:

- The sequences of samples are any κ different items taken from $U_L(x)$ in any order. For example, (Lx_5, Lx_0, Lx_3) (the gray dotted arrows). This sequence is not ordered, since Lx_0 and Lx_3 are closer to Lx_0 than the first element of the sequence, Lx_5. Its correspond-

ing sequence of labels is $(y_5, y_0, y_3) = (\blacklozenge, \bullet, \blacksquare)$, which is also not ordered, since neither $y_5 \leq y_0 \leq y_3$ nor $y_5 \geq y_0 \geq y_3$.

- The sequence of samples (Lx_0, Lx_2, Lx_6) (the red dashed arrows) is ordered, since each term in the sequence is further away from Lx_0 than the previous one. However, it is not a chain, since its corresponding sequence of samples is $(y_0, y_2, y_6) = (\bullet, \times, \bullet)$, and this sequence is not ordered.
- The sequence of labels $(y_6, y_3, y_5) = (\bullet, \blacksquare, \blacklozenge)$ (the blue dashed arrows) is ordered, since it is decreasing ($\bullet \geq \blacksquare \geq \blacklozenge$) and all the elements in the sequence are lower than or equal to the label of the central sample of the neighborhood, $y_0 = \bullet$. However, it is not associated with an ordered sequence of samples, because in (Lx_6, Lx_3, Lx_5), the second item is closer to Lx_0 than the first one. Therefore, this sequence is not a chain.
- The sequences of samples (Lx_0, Lx_1, Lx_2) and (Lx_3, Lx_4, Lx_5) are chains, since they are ordered sequences of samples, and their corresponding sequences of labels are, respectively, ($\bullet$, $\circ$, $\times$) and ($\blacksquare$, $\blacksquare$, $\blacklozenge$), which are also ordered.
- Observe that the central point, Lx_0, may or may not belong to a chain. But when Lx_0 does not belong to a chain candidate the label y_0 is still used to compute if the sequence of labels is ordered. For example, although the sequence (Lx_2, Lx_4, Lx_5) is an ordered sequence of samples and its corresponding sequence of labels, $(y_2, y_4, y_5) = (\times, \blacksquare, \blacklozenge)$, is decreasing, it is not an ordered sequence of labels under our definition, since $y_0 \not\geq y_2 \geq y_4 \geq y_5$. Therefore, this sequence is not a chain.

The reason why we do not consider this last sequence a chain under our definition is because, from the point of view of distance-based ordinal regression, this chain does not move away from the base class of the neighborhood, y_0, gradually. The first element, y_2, is higher than y_0, while the second, y_4, is lower than y_0. Since in an ordinal regression problem the true distances between the class labels are unknown, to consider a sequence as a chain we are only interested in whether it ascends or descends from the base class of the neighborhood, y_0 in this case. Therefore, even though y_0 does not belong to the sequence, we impose this restriction to the definition of chain.

3.2 Optimization

Once the concept of chain has been defined, the problem that CMOML optimizes is

$$\max_{L \in \mathbb{R}^{d' \times d}} \sum_{i=1}^{N} S_L(x_i). \tag{2}$$

Observe that the objective function only takes integer values and therefore, it is not differentiable or even continuous. As a result, it is necessary to make use of black-box non-differentiable optimization methods in order to maximize Eq. (2). We will discuss the optimization method used in our experimental analysis in Sect. 4.2.

In order to avoid the brute force counting of $S_L(x_i)$, which would be computationally expensive according to the inequality in Eq. (1), it is possible to obtain the value of $S_L(x_i)$ using a dynamic programming approach. We will assume that there are no points in $U_L(x_i)$ at the exactly same distance from x_i. If this is not the case, the uniqueness in the order of the sequences of samples, which is needed for the dynamic programming approach, is lost. However, we will show at the end of the section that when we find several samples at the

same distance from x_i we can still apply this algorithm by following a *pessimistic optimization* approach.

First, let us consider the sequence of all the labels associated with all the samples in $\mathcal{U}_L(x_i)$, ordered by the distances of the corresponding samples to x_i; in other words, the sequence of length K_i, $u = (y_{\tau(1)}, \dots, y_{\tau(K_i)})$ so that $\mathcal{U}_L(x_i) = \{Lx_{\tau(1)}, \dots, Lx_{\tau(K_i)}\}$ and

$$\|Lx_{\tau(1)} - Lx_i\| < \|Lx_{\tau(2)} - Lx_i\| < \cdots < \|Lx_{\tau(K_i)} - Lx_i\|.$$

Observe that the inequality above holds when using the strict order ($<$) thanks to the assumption of unequal distances around x_i. We will focus on counting the ascending chains around x_i. The process for counting descending chains is similar. From the sequence u we construct a subsequence v containing only the items in u that are greater than y_i, that is,

$$v = (y_{\tau\delta(1)}, \dots, y_{\tau\delta(R)}),$$

where $R \leq K_i$ and $\delta : \{1, \dots, R\} \to \{1, \dots, K_i\}$ is a strictly increasing map so that $y_{\tau\delta(j)} \geq y_i$ for every $j \in \{1, \dots, R\}$, and δ is also surjective over the set $\{j \in \{1, \dots, K_i\} : y_{\tau(j)} \geq y_i\}$

Now, the problem of counting the ascending chains around x_i is reduced to finding all the increasing subsequences of length κ inside v, since v contains all the labels in the neighborhood of x_i which are greater than y_i, ordered by the distances to x_i. This new problem can be easily solved using dynamic programming, by iteratively filling a $\kappa \times R$ matrix, where the (l, m) entry represents the number of sequences of length l which end at the m-th item in v, and can be recovered from the entries in the row $l - 1$. This process can be done in $O(\kappa R^2)$.

The process of counting descending chains is done by constructing v with the elements of u that are lower than y_i, and counting the decreasing sequences in the new v. Observe that the constant chains are counted twice, since they appear in both ascending and descending chains. So, to obtain the final number of chains $S_L(x_i)$, we have to add the increasing chains and decreasing chains, and subtract the number of constant chains, which can be easily obtained as $\binom{H}{\kappa}$, where $H = |\{j \in \{1, \dots, K_i\} : y_{\tau(j)} = y_i\}|$.

Example 2 We return to the dataset of Example 1 to show how to construct the sequence needed for the dynamic programming algorithm. To facilitate the calculations, we will identify the classes in the dataset with integer numbers: ◆= 0, ■= 1, ●= 2, ◎= 3, ✕= 4. Since the samples in $U_L(x)$ verify that $\|Lx_j - Lx\| < \|Lx_{j+1} - Lx\|$ for $j = 1, \dots, 5$, we obtain the sequence u by adding the labels from y_0 to y_6, that is, $u = (2, 3, 4, 1, 1, 0, 2)$. To count the ascending chains, we compute v by removing the elements that are lower than y_0 from u, obtaining $v = (2, 3, 4, 2)$. Now we can apply the dynamic programming algorithm, obtaining one single chain: (2, 3, 4), associated with the samples (Lx_0, Lx_1, Lx_2) (the ascending chain obtained in Example 1). To count the descending chains we follow a similar process, removing the labels greater than y_0 from u, and obtaining $v = (2, 1, 1, 0, 2)$. Now we apply the dynamic programming algorithm, obtaining three descending chains: (2, 1, 1), (2, 1, 0) and (1, 1, 0). The last one is the descending chain shown in Example 1.

Finally, let us explain how to handle the case where there are examples at the same distance from x_i. Observe that, if the points that are at the same distance from x_i share the same label, they can be swapped in u without affecting the final value of $S_L(x_i)$, so it does not matter in what order they are added to u, since both positions will have the same class label.

For the general case, let us note, first of all, that this situation is dependent on L, and we can apply a small disturbance to it that would break this equality (unless the samples at the same distance to x_i and x_i are aligned)[2]. This means that when we have two different samples being mapped by L to points at the same distance to x_i, it is easy to find another map in an arbitrarily small neighborhood of L that sends them to different points at different distances to x_i.

Moreover, having two different samples from different classes mapped to points at the same distance to x_i is not a positive thing from the point of view of a distance-based classification. For example, consider a sample x_1, of class 1, for which its nearest neighbors (for the distance provided by L) are x_2 and x_3, with classes 2 and 3, respectively, and x_2 and x_3 are at the same distance to x_1. To classify x_1 using the information from x_2 and x_3, a distance-based classifier would consider that x_1 has the same probability to belong to classes 2 and 3. However, in an ordinal regression problem we would expect that class 2 had a greater probability, since it is closer to the real class of x_1 than class 3.

Since we are using a black-box optimizer we propose a pessimistic approach, in which in case of equality, the vector u is constructed by adding the farther classes first. Thus, the number of chains obtained by the optimizer will be lower than the real one, and therefore we will be forcing the optimizer to search for a distance that breaks the equality in the most suitable way. In the previous example, with x_2 and x_3 at the same distance to x_1, we would add to u the class 3 before the class 2. In this way, the objective function for L will get a lower number of chains than the objective function for the disturbances of L for which x_2 is closer to x_1 than x_3 (which will add to u the class 2 before the class 3). Therefore the optimizer will be guided to find distances in this second scenario, which is more appropriate for a distance-based ordinal regression.

In any case, the occurrence of equal distances in a dense space like $\mathbb{R}^d$ is a very unlikely case, even assuming the finite precision of the floating point variables. Note that the pessimistic approach also covers the case of duplicate samples with different labels, for which this approach also allows a unique ordering to be established when constructing u, although in this case the duplicates will always be mapped to equal points. In this case, preprocessing techniques to reduce label noise may be useful before the execution of the algorithm (Frénay & Verleysen, 2013; García et al., 2015).

To sum up, the core of CMOML consists in counting the number of chains for a given linear map matrix L, which can be done using the procedure described above. This evaluation will be repeated for different matrices by a black-box optimizer in order to find an optimum value of chains. The optimizer we have used in our experimental analysis will be discussed in Sec. 4.2.

[2] If x_q, x_r verify that $\|Lx_q - Lx_i\| = \|Lx_r - Lx_i\|$ and x_i, x_q, x_r are not aligned, then $x_q - x_i$ and $x_r - x_i$ are linearly independent, and therefore we can define a linear transformation that maps them to different length vectors, resulting in different distances to x_i. This transformation can be arbitrarily close to L, since we could construct such a transformation by defining the image of $x_q - x_i$ and $x_r - x_i$ as $\lambda L(x_q - x_i)$ and $L(x_r - x_i)$, respectively, with $1 - \varepsilon < \lambda < 1$, for any $\varepsilon > 0$.

If x_i, x_q and x_r are aligned and verifying $\|Lx_q - Lx_i\| = \|Lx_r - Lx_i\|$, then $x_q = x_r$ or x_i is the middle point between x_q and x_r. The first case may be caused by class noise, and is discussed in the text. For the second case, the pessimistic approach described in the text will cause the objective function not to count chains that contain opposing elements within the neighborhood. This is also desirable, since those opposing samples cannot be modified by a linear transformation.

3.3 Non-linear CMOML

Since learning a Mahalanobis distance is equivalent to learning a linear transformation, there are many problems where the inherent non-linearity of the data cannot be properly handled by these distances. To solve this problem, we can use kernel functions in order to be able to learn a linear transformation in a higher dimensional feature space generated by a non-linear mapping. This idea has been successfully applied to other distance metric learning algorithms (Torresani & Lee, 2007; Nguyen et al., 2017), as well as being a fundamental piece in the success of an algorithm of the relevance of the support vector machines (Burges, 1998). We can apply the kernel trick to CMOML to obtain its kernel version, KCMOML.

We consider a non-linear mapping $\phi : \mathbb{R}^d \to \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space of high dimension (or even infinite), denoted as the *feature space*. To learn a distance given by $L : \mathcal{F} \to \mathbb{R}^d$ using CMOML in the feature space we only need to compute the distances between the data after sending them to $\mathcal{F}$. Then the ordered sequences can be obtained in the same way as in the linear case once the distances are known. Assuming that a kernel function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ associated with ϕ is known, if we search for the components of L in the span of $\{\phi(x_1), \dots, \phi(x_N)\}$, we can express L in terms of $\mathcal{K}$ and a new matrix A, of dimension $d' \times N$ (Suárez et al., 2021) as

$$L\phi(x) = A \begin{pmatrix} \mathcal{K}(x, x_1) \\ \dots \\ \mathcal{K}(x, x_N) \end{pmatrix}.$$

From this identity, we can compute the distances $\|L\phi(x_i) - L\phi(x_j)\|$, as well as applying L to any point in $\mathcal{F}$. Therefore, by optimizing the matrix A in order to find the maximum number of chains in the feature space, using again a black-box non-differentiable optimization method (which will be discussed in Sect. 4.2) we obtain the kernelized version of CMOML.

3.4 Complexity analysis of CMOML and comparison with LODML

In this section we will compare the computational time complexity required by the CMOML and LODML methods and their kernel versions.

The parameters that determine the time complexity of LODML are: the number of samples N, the number of features d, the number of target neighbors k, the neighborhood size ν, and the size of the set of triplets $m = |R|$ (Nguyen et al., 2018). Both the target neighbors and the neighborhoods can be built at the first stage of the algorithm using a linear nearest neighbors search, which can be performed in $\mathcal{O}(kN^2 + dN^2)$ and $\mathcal{O}(\nu N^2 + dN^2)$, respectively. The second stage of the algorithm is the optimization procedure, which is an iterative projected gradient descent method consisting of two parts: the gradient computation and its projection onto the positive semidefinite cone. The gradient computation requires the calculation of outer products involving all the constraints in R, which, in the worst case, can be performed in $\mathcal{O}(md^2)$. However, in practice, only a few outer products are required to be computed, which are those corresponding to the constraints that switched their activation state between consecutive iterations of the gradient update. Finally, the projection of the obtained gradient onto the positive semidefinite cone is performed in $\mathcal{O}(d^3)$ and

the metric update is performed in $\mathcal{O}(d^2)$, thus the overall time complexity of LODML is $\mathcal{O}(d^3 + md^2)$ per iteration (Nguyen et al., 2018).

In the case of CMOML, the parameters that determine its complexity are: the number of samples N, the number of features d, the sizes of the neighborhoods $\{K_i : i = 1, \dots, N\}$, the sequence length κ and the output dimension d'. For the worst case estimation, let us assume that $d' = d$ and $K = \max\{K_i : i = 1, \dots, N\}$. The objective function of CMOML requires transforming the dataset using L, which can be performed in $\mathcal{O}(Nd^2)$, and the computation of the pairwise distances for the transformed dataset, which can be performed in $\mathcal{O}(N^2 d)$. Then, computing the number of chains in each neighborhood requires first the neighborhood calculation, which can be performed in $\mathcal{O}(KN)$ since all the pairwise distances in the transformed data were already computed. Then, the sequences u and v can be built in $O(K)$ as shown in Sect. 3.2, and finally the counting operation can be performed in $\mathcal{O}(kK^2)$ (also shown in Sect. 3.2 and using that $R \leq K$ in any neighborhood). This is done for each $x_i \in \mathcal{X}$, resulting in an overall time complexity of the objective function of

$$\mathcal{O}(Nd^2 + N^2 d + N(KN + K + \kappa K^2)) = \mathcal{O}(N(d^2 + \kappa K^2) + N^2(d + K)).$$

To the objective function cost of CMOML we have to add the overhead of the differential evolution steps, which consists of $\mathcal{O}(Pd^2)$ after every P evaluations of the objective function (Das and Suganthan 2010), where P is the population size.

With respect to the kernel versions, both methods include an initial kernel matrix computation, of size $N \times N$, which can be performed in $\mathcal{O}(N^2 d)$. Then, all the dimension-dependent (d) complexities in LODML scale to samples-dependent (N) complexities in KODML due to the kernel trick, resulting in an overall complexity of $\mathcal{O}(N^3 + mN^3)$ for the gradient iterations of KODML. For KCMOML, since we still count the chains after applying the transformation, the output dimension d remains in this stage and only the transformation operation and the differential evolution overhead scale to N, resulting in an overall complexity of $\mathcal{O}(N(Nd + \kappa K^2) + N^2(d + K))$ for the objective function, and $\mathcal{O}(PN^2)$ for the differential evolution overhead.

In summary, we see that both methods run in cubic orders of complexity with respect to (N, d), which is common in most metric learning methods, since at a minimum they usually require operations such as matrix products or decompositions.

Within the cubic order, LODML seems to be a lighter algorithm, and its gradient computation depends only on d. However, CMOML has the advantage that the evaluations of the objective function can be highly parallelized, and a whole generation can be evaluated simultaneously. In contrast, LODML is inevitably iterative as every iteration depends on the completion of the previous one. Finally, KCMOML scales somewhat better than KODML since it still depends on d in many terms of its overall complexity, rather than N.

4 Experiments

In this section we describe the experiments we have developed with CMOML and KCMOML, and the results we have obtained. The results obtained show a high performance of CMOML with respect to the compared algorithms.

First we describe the experiments, datasets and parameters we have used to evaluate the performance of CMOML. Then, we perform a distance metric learning comparison framework that involves CMOML, the Euclidean distance and the state-of-the-art of distance metric learning for ordinal regression. We compare these algorithms using always the same

nearest neighbors classifier. We do this for the linear case and for the kernelized versions of the algorithms. We show the results and verify them using Bayesian statistical analysis. Finally, we show the position of CMOML with respect to the state-of-the-art for ordinal regression, beyond distance metric learning and nearest neighbors classifiers.

4.1 Experimental framework

We have evaluated the distance learned by CMOML using a distance-based classifier. Since we will consider ordinal datasets in the experiments, the classifier used in the experiments is the median-vote k-neighbors classifier (Tang et al., 2020), which is the natural adaptation for the common majority-vote k-neighbors classifier to ordinal regression, as discussed in Sect. 2.2. We have used a number of neighbors $k = 7$. This number of neighbors was chosen due to the fact that starting from this value the median and the mode of the neighbor labels begin to differ significantly. For smaller values they will be very similar in most cases. We have compared our algorithm with the results that the same classifier obtains when using the Euclidean distance, and the distance learned by LODML.

The different distances used by the classifier will be evaluated by a stratified 5-fold cross validation, that is, a cross validation that preserves the original class proportions in each fold. We have used 36 datasets, 23 of them being real-world ordinal regression datasets. The list of datasets has been completed by adding equal-frequency discretized regression datasets. All these datasets are numeric, without missing values, and have been *min-max* normalized to the interval [0, 1] prior to the execution of the experiments. The datasets, their dimensions and the sources from which they have been gathered are shown in Table 1.

Let $h : \mathbb{R}^d \to \mathcal{Y}$ be the hypothesis obtained by the classifier. To evaluate it, we have used the *concordance index* (C-Index), which is one of the most popular metrics for ordinal regression (Cruz-Ramírez et al., 2014; Gönen & Heller, 2005). C-Index measures the ratio between the number of ordered pairs in both true labels and predictions and the number of all comparable pairs, that is

$$C = \frac{\sum_{i,j:y_i<y_j} \left([\![h(x_i) < h(x_j)]\!] + \frac{1}{2}[\![h(x_i) = h(x_j)]\!] \right)}{\#\{y_i, y_j : y_i < y_j\}}$$

where $[\![\cdot]\!]$ denotes the indicator function, that takes the value 1 if the condition inside is satisfied and 0 otherwise. C-Index is not influenced by the numerical representation of the class labels and can be understood as a generalization of the area under the ROC curve (Hanley & McNeil, 1982) for the ordinal case.

To the results on the different datasets we add the average score obtained for each metric, and an average ranking that has been calculated by assigning integer values starting from 1, according to the relative quality of the different algorithms for each dataset.

For CMOML and KCMOML we have used a fixed neighborhood size $K_i = 20$ for every sample in the dataset, a sequence length $\kappa = 7$, an output dimension $d' = d$ and a differential evolution optimizer that is described in detail in Sect. 4.2. For LODML and KODML we have used a number of 7 target neighbors. Target neighbors are set, for each sample, as the nearest same-class neighbors to the sample for the Euclidean distance. Both the sequence length in CMOML and KCMOML and the number of target neighbors in LODML and KODML are set in order to match the number of nearest neighbors used in the subsequent k-NN classification.

Table 1 Datasets used in the experiments

Dataset	# Samples	# Features	# Classes	Source
affairs	265	17	6	Gagolewski (2020)
automobile	202	71	5	Gutierrez et al. (2016)
autoMPG8	392	7	5	Triguero et al. (2017)
auto-riskness	157	15	5	Gagolewski (2020)
balance	625	4	3	Triguero et al. (2017)
boston-housing	506	13	5	Gagolewski (2020)
car	1728	6	4	Triguero et al. (2017)
cement-strength	998	8	5	Gagolewski (2020)
cleveland	297	13	5	Triguero et al. (2017)
conctact-lenses	25	6	3	Gutierrez et al. (2016)
eucalyptus	736	91	5	Gutierrez et al. (2016)
glass	213	9	6	Gagolewski (2020)
newthyroid	215	5	3	Triguero et al. (2017)
pasture	36	25	3	Gutierrez et al. (2016)
squash-stored	52	26	3	Gutierrez et al. (2016)
squash-unstored	52	25	3	Gutierrez et al. (2016)
tae	151	54	3	Triguero et al. (2017)
winequality-red	1359	11	6	Gutierrez et al. (2016)
winsconsin-breast-ord	194	32	5	Gagolewski (2020)
ERA	1000	4	9	Gutierrez et al. (2016)
ESL	482	4	7	Gutierrez et al. (2016)
LEV	1000	4	5	Gutierrez et al. (2016)
SWD	1000	10	4	Gutierrez et al. (2016)
baseball [Discretized]	337	16	5	Triguero et al. (2017)
dee [Discretized]	365	6	5	Triguero et al. (2017)
ele-1 [Discretized]	495	2	5	Triguero et al. (2017)
forestFires [Discretized]	517	12	5	Triguero et al. (2017)
machineCPU [Discretized]	209	6	5	Triguero et al. (2017)
pyrim [Discretized]	74	26	5	Gutierrez et al. (2016)
stock [Discretized]	950	9	5	Gutierrez et al. (2016)
abalone [Discretized]	4177	11	5	Gutierrez et al. (2016)
bank1 [Discretized]	8192	8	5	Gutierrez et al. (2016)
bank2 [Discretized]	8192	32	5	Gutierrez et al. (2016)
computer1 [Discretized]	8192	12	5	Gutierrez et al. (2016)
computer2 [Discretized]	8192	21	5	Gutierrez et al. (2016)
calhousing [Discretized]	20,640	8	5	Gutierrez et al. (2016)

Given that the purpose of this study is to draw a fair comparison between the algorithms and assess their robustness in a common environment with multiple datasets, we have not included a tuning step to maximize any particular performance metric.

The implementations of LODML, CMOML and their kernel versions used in this experimental analysis can be found in our Python library, `pydml` (Suárez et al., 2020).

4.2 Black-box optimizer in CMOML and KCMOML

In order to run the CMOML and KCMOML algorithms it is necessary to establish some kind of black-box optimizer that can evaluate its objective function. In this experimental analysis we have used a differential evolution (Storn & Price, 1997) algorithm. This evolutionary model is updated by taking differences between individuals in order to find a global optimum. Since the function to be optimized is not of high complexity, we have opted for this standard model of differential evolution, in both CMOML and KCMOML. The linear map matrices that define the distances are codified as a real vector of dimension $d' \times d$ that contains all the entries in the matrix added by rows (in the kernel version the matrix A is codified in the same way, but in this case we obtain a $d' \times N$ vector). We have taken the differential evolution implementation available in Scikit-Learn[3] and the parameters we have used are:

- A population size of 100, initialized using the latin hypercube strategy, in order to maximize the coverage of the parameter space (Park, 1994).
- The algorithm has been executed during 150 generations.
- The strategy used to generate new candidates has been *best1bin*, in which the best solution is updated, depending on the recombination probability, with the difference of a pair of individuals taken randomly.
- The decision of updating each of the parameters is made using a binomial distribution with a recombination probability of 0.7. The differential weight that determines how much a parameter is updated in case of recombination is taken randomly, for each generation, from the interval [0.5, 1[. When the new individual is constructed, its fitness is calculated using Eq. (2) (after rewriting it as a matrix) and it replaces the original individual if the new fitness is higher.

4.3 Results for the linear case

Table 2 shows the C-Index results obtained by the algorithms. According to these results, we see a clear dominance of CMOML over the other algorithms. This shows once again the main ability of our algorithm, since a high C-Index indicates that the data are ordered in both the input and the output space.

In order to evaluate to what extent CMOML outperforms the compared algorithms and vice-versa, we have conducted a series of Bayesian statistical tests. We have prepared several Bayesian sign tests (Benavoli et al., 2017) that compare CMOML with each of the other algorithms. These tests take into account the differences between the C-Index scores obtained from the two algorithms being compared, assuming that their prior distribution is a Dirichlet Process (Benavoli et al., 2014), defined by a prior strength $s = 1$ and a prior pseudo-observation $z_0 = 0$. After reading the score observations obtained for each dataset, the tests produce a posterior distribution. This distribution provides us with the probabilities that each of the compared algorithms will outperform the other. The tests also introduce a *region of practically equivalent (rope)* performance, where it is assumed that neither of the algorithms is better than the other. We have designated the rope region as the one where the score differences

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html.

Table 2 C-Index score for the linear version of the algorithms in each dataset

	Euclidean	LODML	CMOML
affairs	**0.569363**	0.565897	0.554959
automobile	0.808623	**0.880245**	0.815365
autoMPG8	**0.927234**	0.895400	0.917576
auto-riskness	0.614520	0.627593	**0.651275**
balance	0.925787	0.960857	**0.976474**
boston-housing	0.795998	0.825344	**0.846487**
car	0.972751	**0.980798**	0.978543
cement-strength	0.738981	0.734096	**0.844991**
cleveland	0.783723	**0.788992**	0.767258
contact-lenses	0.700000	0.700000	**0.828571**
eucalyptus	0.823725	**0.881438**	0.804636
glass	0.774106	0.752447	**0.794482**
newthyroid	0.889352	**0.945833**	0.943981
pasture	**0.867593**	0.801852	0.845370
squash-stored	0.665731	0.721389	**0.773538**
squash-unstored	0.721429	0.772857	**0.803571**
tae	**0.686458**	0.636642	0.677773
winequality-red	0.701366	0.649910	**0.703374**
wisconsin-breast-ord	**0.651560**	0.621095	0.589265
ERA	0.689033	**0.693902**	0.680311
ESL	0.912828	0.906105	**0.917878**
LEV	0.808138	0.792477	**0.810350**
SWD	**0.744839**	0.749787	0.748572
baseball	0.863565	0.766930	**0.877246**
dee	0.875449	0.846101	**0.879377**
ele-1	**0.878193**	0.855523	0.877938
forestFires	0.519372	0.519587	**0.540831**
machineCPU	0.854202	0.860928	**0.871658**
pyrim	0.764274	**0.805812**	0.800085
stock	0.962147	0.688421	**0.962486**
abalone	0.802877	0.724529	**0.805626**
bank1	0.773963	0.942145	**0.949363**
bank2	0.587407	0.770863	**0.784161**
computer1	0.903586	0.784361	**0.910165**
computer2	0.906698	0.855022	**0.909880**
calhousing	0.866630	0.778105	**0.910898**
AVG SCORE	0.786986	0.780091	**0.815398**
AVG RANK	2.229730	2.256757	**1.513514**

The results of the best algorithm for each dataset are highlighted in bold

are in the interval $[-0.01, 0.01]$. In summary, from the posterior distribution we obtain three probabilities: the probability that the first algorithm will outperform the second,

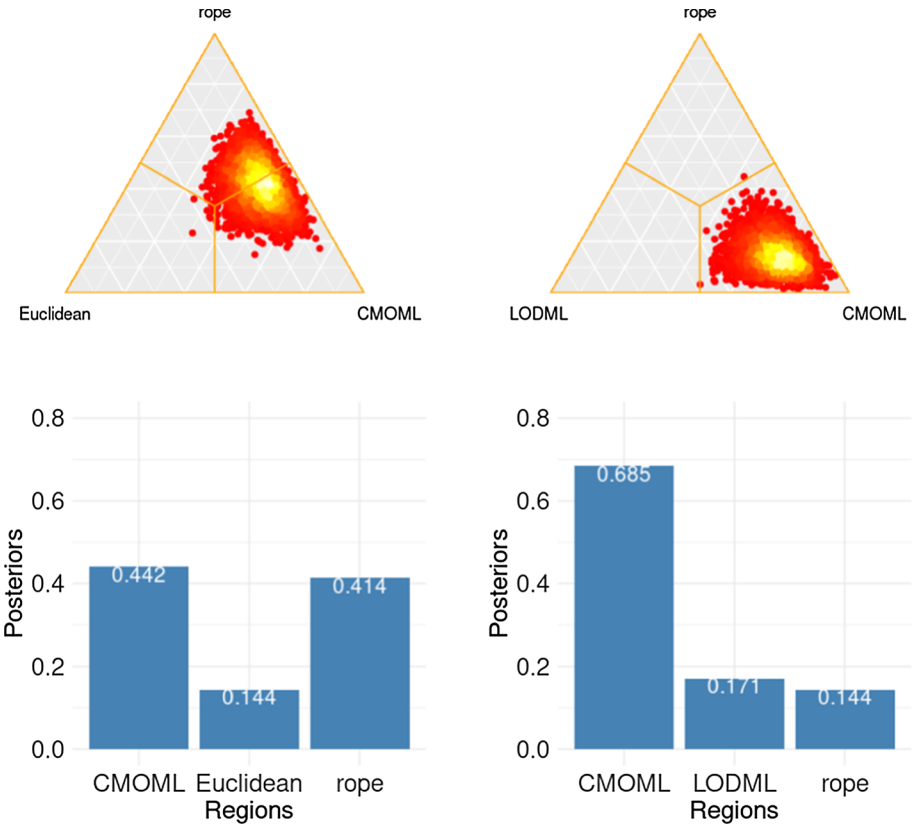

Fig. 3 Bayesian sign test results for the comparison between CMOML with Euclidean distance and LODML using C-Index. Simplex diagrams and posterior distributions are shown (Color figure online)

the probability that the second algorithm will outperform the first, and the probability that both algorithms will have an equivalent performance. The distribution can be displayed in a simplex plot for a sample of the posterior distribution, where a greater tendency of the points towards one of the regions will represent a greater probability.

To carry out the Bayesian sign tests we have used the R package `rNPBST` (Carrasco et al., 2017). Figure 3 shows the results of the comparisons using the C-Index metric.

By examining the results of the tables and the corresponding Bayesian analyses, we can draw several conclusions. The Bayesian analysis shows us that there is a very low probability that Euclidean distance will outperform CMOML, while CMOML has a greater chance of outperforming Euclidean distance, although there is also a remarkable probability that both distances have an equivalent performance. In the comparison between LODML and CMOML we observe again that CMOML has a very high probability of outperforming LODML, with low probabilities for the reciprocal and rope cases.

In general, according to the results observed with this ordinal regression metric, we can conclude that CMOML could be an interesting alternative when looking for distances aimed at ordinal regression problems.

4.4 Results for the non-linear case

Finally, we will show the results of the experiments with the kernelized versions of CMOML and LODML. We have evaluated KCMOML and KODML over the datasets previously described in Table 1, using the same classifier, validation strategies, parameters and metrics. Since both kernel versions learn linear transformations that scale quadratically with the number of samples, which can be computationally expensive for large datasets (Nguyen et al., 2018), we have set a time limit of one week for the kernel experiments. We have evaluated both algorithms using two of the most popular kernels, namely:

- The *polynomial kernel*, $\mathcal{K}(x, x') = \gamma\langle x, x'\rangle^p$. We have used $p = 2$, thus we have obtained a quadratic transformation of the data.
- The *radial basis function kernel*, or RBF. It is defined as $\mathcal{K}(x, x') = \exp(-\gamma\|x - x'\|^2)$. The image of the non-linear mapping associated with this kernel is an infinite-dimensional space.

The value γ in both kernels has been tuned by cross-validation using the set of values $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000\}$.

The results of the kernelized versions using the C-Index score are shown in Table 3. The symbol (−) in a cell of the table indicates that the corresponding algorithm has exceeded the established time limit. Observing these results we can see that the kernel versions are able to obtain better results than the linear versions in most of the datasets. The improvement observed for KODML with respect to LODML is higher than the improvement of KCMOML with respect to CMOML, but KCMOML still obtains better average results, with the RBF kernel obtaining the best average scores and the polynomial kernel obtaining the best average rankings. Analyzing the results of the Bayesian tests[4] from Fig. 4 we can see there is pretty level playing field between the two algorithms when using the polynomial kernel, with a very high probability that the algorithms will have equivalent performances, while with the RBF kernel the probability distribution in the three regions is very even, although slightly biased towards KCMOML. This tells us that KCMOML has a slightly higher probability of performing better than KODML, although a similar probability for the KODML region implies that the algorithms are able to complement each other well for different datasets. It is also interesting to observe that KCMOML with the RBF kernel outstands in most of the real-world ordinal regression datasets, while its performance on the discretized regression datasets slightly worsens. Finally, it should be noted that KCMOML only times out in 2 of the large datasets used in the experiments, while KODML times out in 5 of them. This confirms what the complexity analysis showed about KCMOML scaling better than KODML with respect to the number of samples.

4.5 Comparison with state-of-the-art methods for ordinal regression

Since CMOML has proven to be an outstanding algorithm within the family of distance metric learning algorithms for ordinal regression, in this section we compare the proposed

Table 3 C-Index score for the kernel version of the algorithms in each dataset

	KODML		KCMOML	
	POLY-2	RBF	POLY-2	RBF
affairs	0.586422	0.545643	0.567910	**0.601115**
automobile	0.814357	0.802392	**0.862178**	0.851728
autoMPG8	0.927905	0.908938	0.921637	**0.928632**
auto-riskness	0.664561	**0.708308**	0.660411	0.671843
balance	0.965443	0.986929	**0.993742**	0.991133
boston-housing	**0.834244**	0.814329	0.814222	0.807763
car	0.978700	0.973873	0.984611	**0.985772**
cement-strength	0.784070	0.760296	**0.818427**	0.806169
cleveland	0.797910	**0.814295**	0.778811	0.785053
contact-lenses	0.857143	0.857143	0.871429	**0.885714**
eucalyptus	**0.874315**	0.851291	0.827302	0.793716
glass	0.774331	0.774331	0.797583	**0.802359**
newthyroid	0.948843	0.949769	0.943981	**0.966435**
pasture	0.861111	0.846296	**0.867593**	0.831481
squash-stored	0.753582	0.718582	0.771637	**0.858450**
squash-unstored	0.780714	**0.815000**	0.757857	**0.815000**
tae	0.700905	**0.725865**	0.675628	0.700214
winequality-red	0.703518	0.681436	**0.705966**	0.702759
wisconsin-breast-ord	0.656351	**0.659343**	0.616383	0.603888
ERA	0.702709	0.703945	**0.706104**	0.702813
ESL	0.923134	0.920086	**0.926032**	0.920399
LEV	0.813103	0.808132	**0.813145**	0.810783
SWD	0.755780	**0.757707**	0.752039	0.756966
baseball	0.863850	0.805774	**0.866594**	0.865433
dee	0.882748	0.861711	**0.889650**	0.888016
ele-1	**0.881161**	0.877794	0.879256	0.870785
forestFires	0.538324	**0.541348**	0.520120	0.517618
machineCPU	**0.877991**	0.864206	0.872821	0.854135
pyrim	0.779829	0.815556	**0.837949**	0.782308
stock	**0.963650**	0.955048	0.958975	0.961427
abalone	0.731883	0.736535	**0.768793**	0.756074
bank1	–	–	0.844700	**0.884079**
bank2	–	–	–	–
computer1	–	–	0.888985	**0.899613**
computer2	–	–	0.865846	**0.913115**
calhousing	–	–	–	–
AVG SCORE[a]	0.805760	0.801351	0.807379	**0.808903**
AVG RANK[a]	2.532778	2.859521	**2.217482**	2.390218

The results of the best algorithm for each dataset are highlighted in bold

[a]In order to conduct a fair comparison between the two methods, the average score, the average ranking and the Bayesian test results exclude the datasets *bank1*, *bank2*, *computer1*, *computer2* and *calhousing* in the kernel experiments

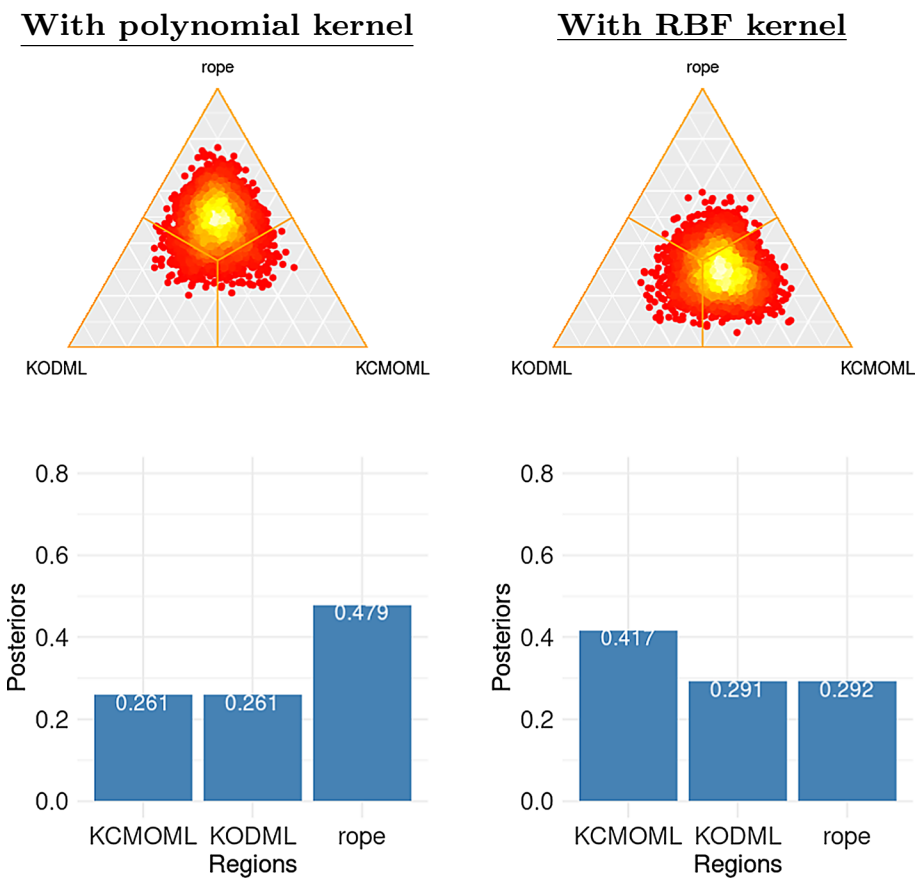

Fig. 4 Bayesian sign test results for the comparison between KCMOML and KODML, with polynomial and RBF kernels, using C-Index. Simplex diagrams and posterior distributions are shown (Color figure online)

algorithm to other state-of-the-art algorithms for ordinal regression. The experiments were performed using the two methods that obtained the best results in the experimental analysis of Gutierrez et al. (2016), namely, SVOREX (Chu & Keerthi, 2007) and REDSVM (Lin & Li, 2012). These two algorithms are adaptations of support vector machines for ordinal regression. The first adds explicit constraints concerning adjacent classes for threshold determination, while the second reduces the ordinal regression problem to be applied to binary SVM classifiers.

In addition to these SVM variants we have included two more recent proposals for ordinal regression in our comparison. The first is an adaptation of the extreme learning machines, *kernel extreme learning machine for ordinal regression* (KELMOR) (Shi et al., 2019). The second is an extension of the Bayesian network classifiers (Halbersberg et al., 2020). This extension learns a Bayesian network that jointly maximizes accuracy and mutual information, in order to better adapt to imbalanced and ordinal problems. We will refer to this algorithm as EBNC (*extended Bayesian network classifier*).

Both SVMs and KELMOR have been used with a Gaussian kernel and an adjusting parameter $C = 10$. For the Bayesian network classifier, the features were discretized to a

Table 4 C-Index score for the state-of-the-art methods and CMOML in each dataset

	REDSVM	SVOREX	EBNC	KELMOR	CMOML
affairs	0.543764	0.479077	0.544441	**0.570416**	0.554959
automobile	**0.853380**	0.759912	0.776466	0.838471	0.815365
autoMPG8	0.921465	**0.933516**	0.915756	0.918324	0.917576
auto-riskness	0.696078	**0.763215**	0.740496	0.696718	0.651275
balance	0.988387	**0.994224**	0.930498	0.949678	0.976474
boston-housing	0.838319	0.838910	0.763829	**0.851915**	0.846487
car	0.956863	**0.982058**	0.974778	0.952840	0.978543
cement-strength	0.836725	**0.864907**	0.702598	0.799276	0.844991
cleveland	0.820070	0.794660	0.733414	**0.835190**	0.767258
contact-lenses	0.700000	0.685714	0.614286	**0.885714**	0.828571
eucalyptus	**0.882360**	0.854718	0.853683	0.868132	0.804636
glass	0.717674	0.773195	0.642967	0.734655	**0.794482**
newthyroid	0.750463	0.857407	0.833796	0.776157	**0.943981**
pasture	0.827778	0.757407	0.846296	**0.854630**	0.845370
squash-stored	0.777310	0.811784	**0.830044**	0.766608	0.773538
squash-unstored	0.761429	0.775000	0.710000	0.775714	**0.803571**
tae	0.604379	**0.684285**	0.636560	0.594199	0.677773
winequality-red	0.723094	**0.730985**	0.726721	0.720977	0.703374
wisconsin-breast-ord	**0.615969**	0.599485	0.516718	0.604256	0.589265
ERA	0.710939	0.680576	0.692489	**0.720854**	0.680311
ESL	0.929687	**0.932802**	0.908091	0.929938	0.917878
LEV	0.811428	**0.822995**	0.801398	0.809583	0.810350
SWD	0.744378	0.761199	**0.780408**	0.735707	0.748572
baseball	**0.881180**	0.856130	0.830042	0.875346	0.877246
dee	**0.900075**	0.894234	0.868829	0.889131	0.879377
ele-1	**0.885381**	0.884788	0.848548	0.870766	0.877938
forestFires	0.515009	0.512263	0.499984	0.523941	**0.540831**
machineCPU	**0.885147**	0.882376	0.777610	0.865265	0.871658
pyrim	**0.852821**	0.732393	0.729316	0.792393	0.800085
stock	0.922424	0.959384	0.940706	0.924979	**0.962486**
abalone	0.813377	**0.823557**	0.769140	0.813790	0.805626
bank1	**0.953372**	0.952471	0.714695	0.923420	0.949363
bank2	**0.824870**	0.773617	0.708871	0.801277	0.784161
computer1	0.899223	**0.910422**	0.808520	0.899727	0.910165
computer2	0.910846	**0.927012**	0.880130	0.914840	0.909880
calhousing	0.862286	0.874078	0.801631	0.857212	**0.910898**
AVG SCORE	0.808832	0.808910	0.768160	0.809501	**0.815398**
AVG RANK	2.729730	**2.405405**	4.162162	2.891892	2.810811

The results of the best algorithm for each dataset are highlighted in bold

maximum of ten values per attribute, using equal-length bins, so that they could be handled by the network. CMOML has been used with the same settings as in Sect. 4.3. As in the previous experiments, we have carried out a stratified 5-fold cross validation. We have

taken the implementations of the SVMs from Sánchez-Monedero et al. (2019) and the code of EBNC provided by Halbersberg et al. (2020). We also provide an implementation of KELMOR[4]. The C-Index scores obtained by CMOML and each of the state-of-the-art classifiers are shown in Table 4.

Looking at the results, we can observe that CMOML is slightly behind the SVMs in terms of rankings. However, although CMOML only ranks first in 6 of the datasets in C-Index, it achieves outstanding wins in datasets such as *newthyroid*, *glass* or *calhousing*, with convincing wins in the rest of its top positions as well. Moreover, in most of the cases where CMOML performs worse than the other algorithms, its results are still competitive with the results of the rest of algorithms. This translates into the best average C-Index.

In summary, CMOML has the ability to excel in several ordinal datasets where other state-of-the-art methods do not perform well, in addition to having decent overall performance. Finally, as we will see in the next section, CMOML stands out from the compared methods in terms of explainability, regarding case-based reasoning, dimensionality reduction and visualization. Both support vector machines and extreme learning machines are considered by design complex black-box learning algorithms (Arrieta et al., 2020; Adadi & Berrada, 2018). Their opaque structure turns them into undesirable algorithms for high-risk automated tasks. In contrast, the transparency of the nearest neighbors-based algorithms and the information provided by the neighbors themselves make them much more useful in these tasks (Lamy et al., 2019). We will see in the next section that the distance learned by CMOML, besides considerably improving the performance of the k-NN, makes the interpretable information produced by the classifier much more intuitive.

5 Nearest neighbors and metric learning for an explainable learning process

In this section we will explore the explainability possibilities offered by CMOML when it is applied together with the nearest neighbors classifier. *Explainable artificial intelligence (XAI)* (Arrieta et al., 2020; Belle & Papantonis, 2020) has recently gained new relevance as a research topic as a consequence of the growing need for transparency and interpretability in the large amount of highly complex models, such as ensembles or deep neural networks, that currently dominate machine learning.

We saw in the previous section that CMOML obtains results close to those of the state-of-the-art for ordinal regression. The purpose of this section is to show that, in addition to the foregoing, CMOML has the advantage over the state-of-the-art algorithms in that, since it can be used in combination with a distance-based algorithm such as the nearest neighbors classifier, it can benefit from the strengths that k-NN provides in terms of explainability, and even polish them in several ways, as we will see throughout the section.

The nearest neighbors classifier, like most similarity-based classifiers, can be interpreted in terms of *case-based reasoning* (Lamy et al., 2019). The k-NN makes its decisions based on the similarity with previous experiences (the k nearest neighbors in the learned training set). Therefore, it is possible to analyze these experiences to decide to what extent the decision made by the algorithm can be trusted. This is very similar to human decision making, which often relies on previous experiences. In addition, when our data is low-dimensional

[4] https://github.com/jlsuarezdiaz/KELMOR

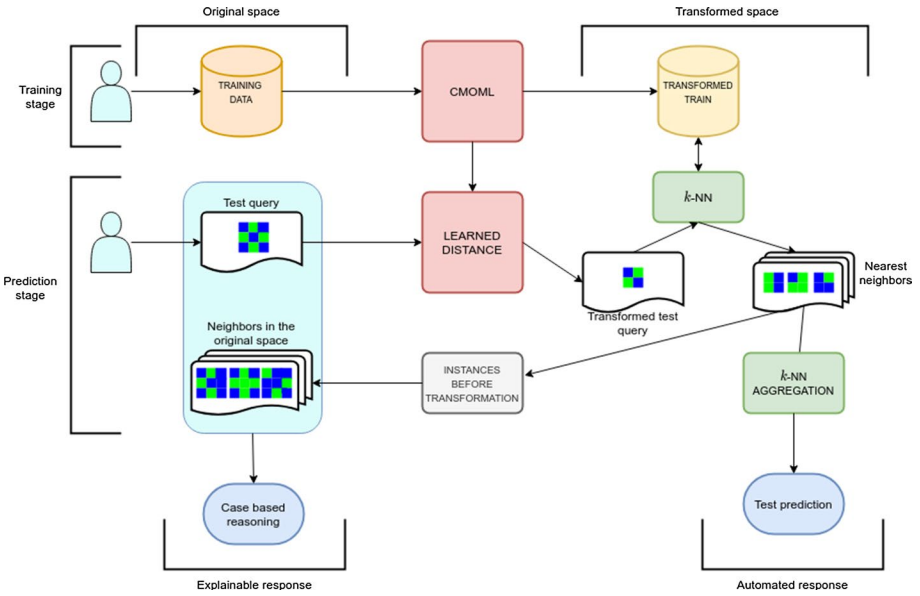

Fig. 5 A case-based reasoning approach with nearest neighbors and CMOML (Color figure online)

it is possible to visualize why the *k*-NN has decided to make a certain choice, thanks to the simplicity of the "nearest neighbor" concept.

If we want to classify a new sample using the *k*-NN and we fit an appropriate distance to our training data, the nearest neighbors of the sample may change and become even more similar to the sample. In this way, these neighbors will be more informative when interpreting a decision. This is possible with distance metric learning. In particular, while the CMOML optimization algorithm improves the layout of the data to perform better under ordinal regression metrics, it also modifies nearby instances to be more related to the sample to be predicted. In addition, we can pick the dimension to which we want CMOML to project the data.

Therefore, in terms of explainability, CMOML improves the traditional *k*-NN in two aspects:

- *Comprehensibility* The knowledge the *k*-NN with CMOML learns can be represented by the nearest neighbors, and these neighbors will be much more informative than the neighbors the Euclidean distance would obtain.
- *Understandability* The ability to reduce the dimensionality of CMOML allows data to be represented in spaces of lower complexity where it is easier to understand why the classifier makes a certain decision.

We will test these CMOML capabilities in two different datasets: *balance* and *newthyroid*. In these datasets, CMOML outperforms the Euclidean distance and also obtains good results compared to the state-of-the-art algorithms, as shown in Table 4. CMOML clearly outperforms all the algorithms in *newthyroid* and it is very close to the SVMs in *balance*. In this last case, the gains in interpretability that we are going to show may compensate the minimal loss in the ordinal metrics.

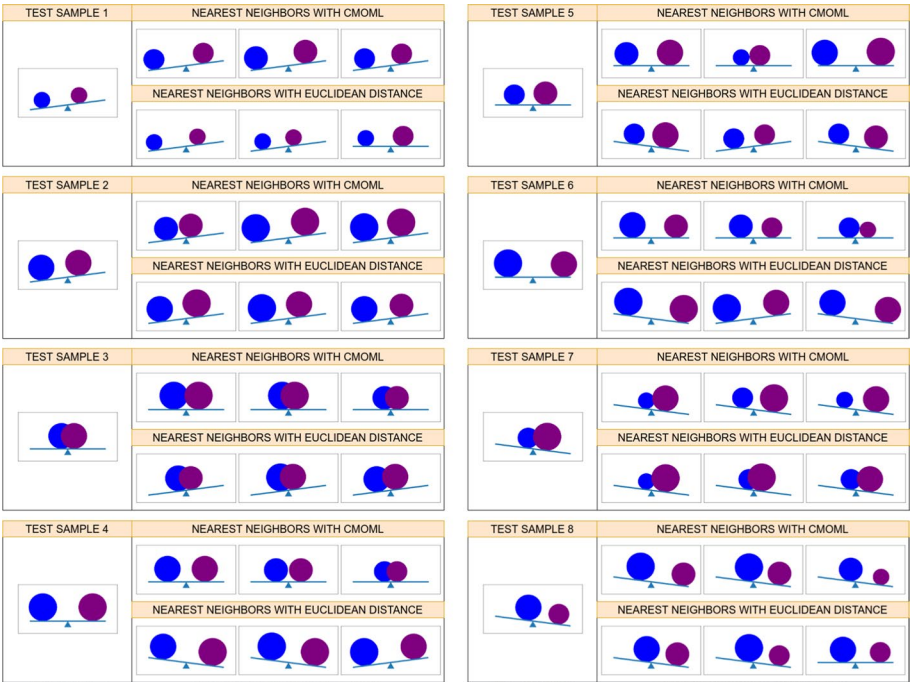

Fig. 6 Nearest neighbors of some test samples in `balance` using CMOML and Euclidean distance (Color figure online)

5.1 Knowledge representation with CMOML

Here we will analyze how the distance learned by CMOML improves knowledge representation for a case-based reasoning, with the neighbors obtained by the k-NN. To do this, we follow the outline in Fig. 5. In short, once the distance is learned, we use it to transform the new sample that is to be classified. In the transformed space we retrieve its nearest neighbors. With them, on the one hand, the classifier makes its prediction, aggregating the classes of the neighbors (for the ordinal case, we use the median class here again). On the other hand, we retrieve the neighbors in the original space and visualize them together with the case to be predicted.

Below we perform this procedure with several examples in *balance* and *newthyroid*.

5.1.1 Balance dataset analysis

Balance is a dataset whose examples represent two objects placed at the sides of a scale. Its attributes consist of the weight and the position of the left item and the weight and the position of the right item. The goal is to predict whether the scale tilts to one side (*right* or *left*) or whether it stays in balance. Due to the continuity of the scale movement it is logical to assume the relation of order in the output variable $left < balance < right$. This dataset is very useful to help visualize how CMOML improves the neighbors that the k-NN retrieves.

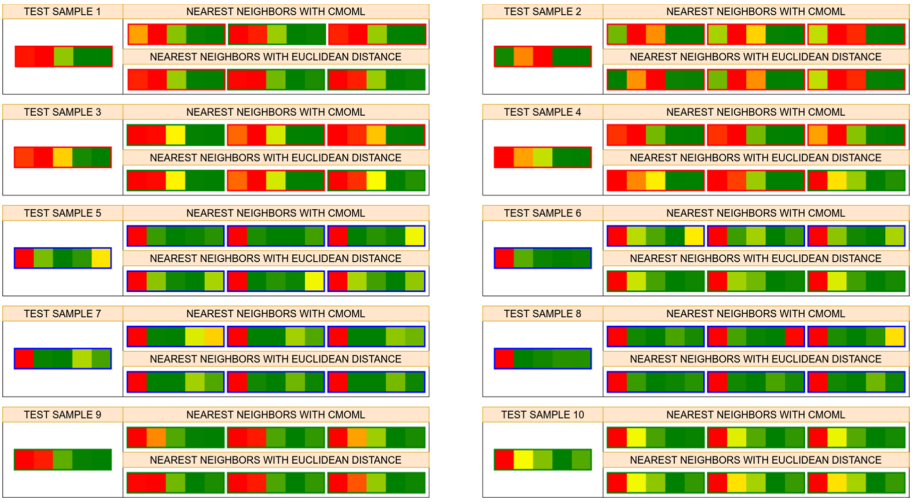

Fig. 7 Nearest neighbors of some test samples in `newthyroid` using CMOML and Euclidean distance (Color figure online)

Indeed, we will see that these neighbors are very similar to what a human being would consider, if prompted to decide how the scale will move based on previous experiences.

We train CMOML with 80 % of the dataset and use the rest for testing. For the test data, we retrieve the 3 nearest neighbors obtained by CMOML, and we also retrieve the nearest neighbors that the Euclidean distance would obtain. Figure 6 shows the results for some of the test samples.

From the results obtained we can draw several conclusions. On the one hand, for the test data that are not in balance, the neighbors that the two distances obtain are often same-class neighbors, especially for CMOML. This is reasonable due to the geometric properties of the dataset. Observe that the true labels are determined by the sign of `left_weight * left_distance - right_weight * right_distance`, so the *left* and *right* classes are determined by half-spaces. Even so, we can see that the neighbors obtained by CMOML are more intuitive than those obtained by the Euclidean distance. For example, we can see, for the test sample 1, that the neighbors that CMOML obtains are always objects of the same weight, which also keep ratios between the distances to the center of the scale. Presumably, if we, as humans, had to decide where the scale would tilt to based on the same previous experiences, we would have chosen those very same neighbors. With the Euclidean distance this does not happen so clearly.

On the other hand, for the test data that are in balance, the Euclidean distance often tends to choose neighbors that tilt to one of the sides. This is due to the fact that this distance is not optimized and the *balance* class is a hyperplane between the other two classes, so as soon as a training sample of the *left* or *right* class is close to this border it can easily interfere with the neighbors. Instead, the neighbors that CMOML obtains are mostly in the *balance* position. And not only that, but again they are much more intuitive. Some of the clearest cases are the test samples 3 and 4. There, both objects weigh the same and are at the same distance from the center of the scale. All their neighbors share this property. Therefore, the answer is clear when analyzing why these neighbors have been chosen.

5.1.2 Newthyroid dataset analysis

Newthyroid is a dataset whose input variables are the measurements of certain hormones in the human body: `T3 resin`, `thyroxin`, `triiodothyronine`, `thyroidstimulating` and `TSH`. The goal is to predict, with these measures, whether the individual being evaluated suffers from hyperthyroidism, hypothyroidism or is healthy. As both disorders can be considered opposed, it is assumed that the ordinal relationship of the output variable is *hypothyroidism* < *normal* < *hyperthyroidism*.

Once again, we train CMOML with 80 % of the dataset, use the rest for testing, and retrieve the 3 nearest neighbors obtained by CMOML and by the Euclidean distance for the test data. In Fig. 7 we show the neighbors obtained for some of the most remarkable cases. Here, we represent an instance by a heatmap, where each cell is each of the input variables in the data set, in the same order as described above. These cells range from the lowest possible value for the attribute (green) to its highest possible value (red). We also highlight the border of the heatmaps to specify the true labels of each instance. Blue borders represent instances of class *hypothyroidism*, green borders represent instances of class *normal* and red borders represent instances of class *hyperthyroidism*.

From the properties we have observed we can conclude several facts. First of all, it is very common to always find neighbors of the class *normal* for the test samples in the *normal* class. This is reasonable, since the normal class is the most frequent and quite dense, as we will see in the next section. It is also the least important class of the problem, since, as with most medical-related problems, false positives in disease diagnosis are not as serious as false negatives. That is why in Fig. 7 we choose to focus on the possible positives (both *hypothyroidism* and *hyperthyroidism*) to perform a case-based reasoning, taking only the samples 9 and 10 from the *normal* class.

Focusing now on the samples with *hyperthyroidism* (samples 1-4) and *hypothyroidism* (samples 5-8), we can see that the neighbors obtained by CMOML are much more label-accurate than the neighbors obtained by the Euclidean distance. In fact, in some cases like test samples 3 and 6, the *k*-NN with the Euclidean distance would misclassify the samples, since the median class in these cases would be *normal*. CMOML can still classify these samples correctly because most of the neighbors it obtains are from the correct class.

Finally, we analyze how the distance learned by CMOML influences the values of the features of the nearest neighbors. We can observe that, while the Euclidean distance has the limitation of only providing neighbors whose features are all very similar one-by-one to the features of the test sample, CMOML can go a step further and provide neighbors with more distinguishing properties. For example, it is known that low values of `thyroxin` combined with very low or high values of `TSH` are usually a symptom of *hypothyroidism*. For the test sample 6 we observe a low value of `TSH` and a medium-low value of `thyroxin`. The neighbors obtained by the Euclidean distance show `thyroxin` values that are not low enough and, additionally, they are tagged as *normal*. However, CMOML provides neighbors with lower values of `thyroxin`, and values of `TSH` that are both low and high. That is, the algorithm is discovering that both low and high values of `TSH` influence *hypothyroidism*, and finds a new similarity between the data that the Euclidean distance overlooks.

Table 5 C-Index scores with CMOML in `balance` and `newthyroid` after applying a dimensionality reduction

Dimension	C-INDEX	
	Balance	Newthyroid
MAX	0.976474	0.943981
3	0.973122	0.943981
2	0.970433	0.943981

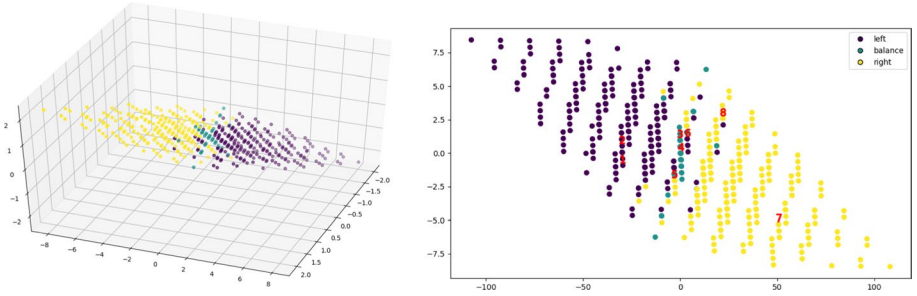

Fig. 8 3D and 2D projections of `balance` learned by CMOML. The 2D projection shows the position of the transformed test samples in Fig. 6 (Color figure online)

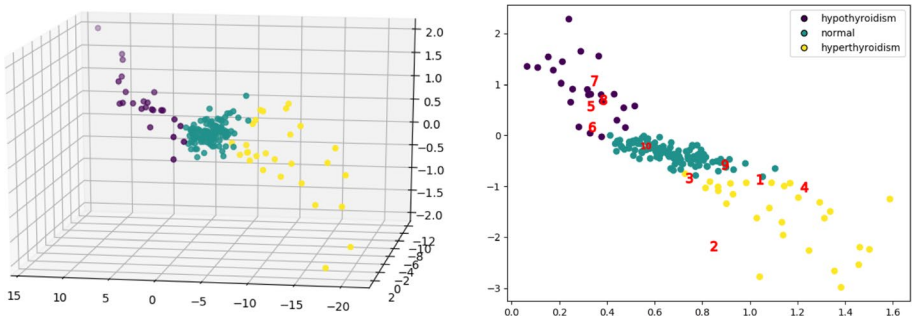

Fig. 9 3D and 2D projections of `newthyroid` learned by CMOML. The 2D projection shows the position of the transformed test samples in Fig. 7 (Color figure online)

5.2 Dimensionality reduction and visualization

To conclude, we will analyze how the dimensionality reduction applied by CMOML allows for the visualization of the transformed data, so that the decision made by the nearest neighbors classifier can be understood. To do this, we learn a distance with CMOML for each dataset and impose a dimension of 3 and 2 in the transformed space. In this way, the algorithm will obtain a projection of the data in the three- and bi-dimensional spaces that will be adequate from an ordinal perspective, as shown throughout the paper. With this

dimensionality reduction, the results of the cross validation over *balance* decrease by less than a hundredth for the 3D projection, using C-Index, and two hundredths more in the 2D projection. In *newthyroid* the projections maintain the same results as at maximum dimension, as it is shown in Table 5. Therefore, we can conclude that the reduction does not significantly affect the quality of the data, and thus the transformed data can be used to extract human-understandable information.

In Figs. 8 and 9 we show how the data from *balance* and *newthyroid*, respectively, look like when projected on three and two dimensions. In the 2D-projection we also include the test samples from Figs. 6 and 7, to facilitate the understanding of the choices the classifier makes. The test samples are marked with red numbers. Each number corresponds to the sample number they had in the aforementioned figures.

By analyzing the 3D and 2D projections of *balance* we can see why the *left* and *right* classes are so easy to classify and why the *balance* class presents a major difficulty. Indeed, the latter class acts as a hyperplane that separates the other classes. Therefore, if the training set is not very populated in this class, it is easy for neighbors of the *left* and *right* classes to decrease the quality of the classification of true *balance* instances. In any case, the projections learned by CMOML, especially at maximum dimension and to the 3D-space, have proven to be good enough on the central class. Moreover, on the 2D projection we can observe that, indeed, the test samples of Fig. 6 of the class *balance* always fall on training samples of the class *balance*.

If we now analyze the projections of *newthyroid*, we see that the *normal* class stands in the center between the two disorders, while the points of the other classes usually take values that shoot up in some dimension. In general, there is a fairly clear separation among the classes, with the exception of certain border points for which some doubts may arise when it comes to classification. This can be corroborated with the test samples from Fig. 7 that are plotted in the 2D-projection: it is clear that most of them can be correctly classified using the scatter plot, while the test samples 1 and 3 can cause some more confusion, as can be seen in Fig. 7.

It should be noted that the projections respect the ordinal principle by which CMOML is guided: that, as we move farther away from any sample, the classes will gradually become more different. This happens with the two datasets analyzed. Finally, we have to mention that 2D or 3D projections are not always possible with CMOML without a significant loss of information. In any case, a dimensionality reduction even to dimensions greater than 2 and 3 can be beneficial in terms of efficiency and noise reduction. In addition, it is always possible to use other visualization methods in larger dimensions (Maaten & Hinton, 2008), which can perform better if we first apply CMOML properly.

6 Conclusions

In this paper we have developed a new distance metric learning algorithm for ordinal regression, following a new approach based on the optimization of ordered sequences. This approach is purely ordinal, since it makes an extensive use of the relative positions among the labels and it has proven to be effective in situations where other proposals on the same topic cannot operate properly.

According to how the proposal has been developed and how the results have supported it, we can conclude that CMOML is a promising algorithm with strong capabilities in numerous ordinal regression problems. The sequence based approach provides CMOML

with a different and effective way of handling the ordinal regression problems, compared to the previous distance metric learning proposals in the subject. In the general context of ordinal regression, CMOML has also proven to be competitive and more explainable that the best performing methods, and it improves significantly the explainablity provided by the traditional Euclidean nearest neighbors, with additional advantages such as dimensionality reduction and understandability-focused visualization.

As future work we plan to further investigate the CMOML optimization method, in order to handle even larger datasets than those used in this work. In particular, image datasets for ordinal regression are the order of the day (Li et al., 2019; Diaz & Marathe, 2019), and dealing with them at the pixel level using Mahalanobis distances is computationally challenging. In this situation it may be interesting to explore the behaviour of the algorithm when it receives the feature maps extracted by a convolutional network (Min et al., 2009). For the purpose of improving the optimization method, we will also explore both the use of optimizers with higher convergence speed and programming paradigms and architectures that maximize the parallel evaluation of the algorithm (Muñoz et al., 2015).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access, 6,* 52138–52160.

Agresti, A. (2010). *Analysis of Ordinal Categorical Data* (Vol. 656). New York: John Wiley & Sons.

Antoniuk, K., Franc, V., & Hlaváč, V. (2016). V-shaped interval insensitive loss for ordinal classification. *Machine Learning, 103*(2), 261–283.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion, 58,* 82–115.

Beckham, C., & Pal, C. (2017) Unimodal probability distributions for deep ordinal classification. In: Proceedings of the 34th International Conference on Machine Learning, pp 411–419

Belle, V., & Papantonis, I. (2020) Principles and practice of explainable machine learning. arXiv preprint arXiv:200911698

Benavoli, A., Corani, G., Mangili, F., Zaffalon, M., & Ruggeri, F. (2014) A bayesian wilcoxon signed-rank test based on the dirichlet process. In: International Conference on Machine Learning, pp 1026–1034

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research, 18*(1), 2653–2688.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121–167.

Bürkner, P. C., & Vuorre, M. (2019). Ordinal regression models in psychology: A tutorial. *Advances in Methods and Practices in Psychological Science, 2*(1), 77–101.

Calvo, T., & Beliakov, G. (2010). Aggregation functions based on penalties. *Fuzzy Sets and Systems, 161*(10), 1420–1436.

Cardoso, J. S., & da Costa, J. F. P. (2007). Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research, 8*(50), 1393–1429.

Carrasco, J., García, S., del Mar Rueda, M., & Herrera, F. (2017). rnpbst: An r package covering non-parametric and bayesian statistical tests. In: International Conference on Hybrid Artificial Intelligence Systems, Springer, pp 281–292

Chakraborty, S., & Church, E. M. (2020). Social media hospital ratings and hcahps survey scores. *Journal of Health Organization and Management*.
Cheng, J,, Wang, Z., & Pollastri, G, (2008) A neural network approach to ordinal regression. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, pp 1279–1284
Chu, W., & Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation, 19*(3), 792–815.
Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21–27.
Cruz-Ramírez, M., Hervás-Martínez, C., Sánchez-Monedero, J., & Gutiérrez, P. A. (2014). Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing, 135,* 21–31.
Das, S., & Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*(1), 4–31.
Diaz, R., & Marathe, A. (2019) Soft labels for ordinal regression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4738–4747
Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems Man and Cybernetics, 4,* 325–327.
Fathony, R., Bashiri, MA., & Ziebart, B. (2017) Adversarial surrogate losses for ordinal regression. In: Advances in Neural Information Processing Systems, pp 563–573
Fouad, S., & Tiňo, P. (2013) Ordinal-based metric learning for learning using privileged information. In: The 2013 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8
Frank, E., & Hall, M. (2001) A simple approach to ordinal classification. In: European Conference on Machine Learning, Springer, pp 145–156
Frénay, B., & Verleysen, M. (2013). Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems, 25*(5), 845–869.
Fu, H., Gong, M., Wang, C., Batmanghelich, K., & Tao, D. (2018) Deep ordinal regression network for monocular depth estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2002–2011
Gagolewski, M. (2020) Ordinal regression benchmark data. https://www.gagolewski.com/resources/data/ordinal-regression/, accessed: 2020-09-10 (YYYY-MM-DD)
García, S., Luengo, J., & Herrera, F. (2015). *Data Preprocessing in Data Mining*. Berlin: Springer.
Gönen, M., & Heller, G. (2005). Concordance probability and discriminatory power in proportional hazards regression. *Biometrika, 92*(4), 965–970.
Gu, B., Geng, X., Shi, W., Shan, Y., Huang, Y., Wang, Z., & Zheng, G. (2020). Solving large-scale support vector ordinal regression with asynchronous parallel coordinate descent algorithms. *Pattern Recognition, 109*(107592).
Guijo-Rubio, D., Casanova-Mateo, C., Sanz-Justo, J., Gutiérrez, P., Cornejo-Bueno, S., Hervás, C., & Salcedo-Sanz, S. (2020). Ordinal regression algorithms for the analysis of convective situations over madrid-barajas airport. *Atmospheric Research, 236*.
Gutierrez, P. A., Perez-Ortiz, M., Sanchez-Monedero, J., Fernandez-Navarro, F., & Hervas-Martinez, C. (2016). Ordinal regression methods: Survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering, 28*(1), 127–146.
Halbersberg, D., Wienreb, M., & Lerner, B. (2020). Joint maximization of accuracy and information for learning the structure of a bayesian network classifier. *Machine Learning, 109,* 1039–1099.
Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology, 143*(1), 29–36.
Jolliffe, I. (2002). Principal Component Analysis. Springer Series in Statistics, Springer
Joshi, A., Kale, S., Chandel, S., & Pal, DK. (2015). Likert scale: Explored and explained. *Current Journal of Applied Science and Technology*, pp 396–403
Kuráňová, P. (2016). Modelling the results of the phadiatop test using the logistic and ordinal regression. In: Applications of Computational Intelligence in Biomedical Technology, Springer, pp 103–118
Lamy, J. B., Sekar, B., Guezennec, G., Bouaud, J., & Séroussi, B. (2019). Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. *Artificial Intelligence in Medicine, 94,* 42–53.
Li, W., Lu, J., Feng, J., Xu, C., Zhou, J., & Tian, Q. (2019). Bridgenet: A continuity-aware probabilistic network for age estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1145–1154
Lin, HT., & Li, L. (2009). Combining ordinal preferences by boosting. In: Proceedings ECML/PKDD 2009 Workshop on Preference Learning, pp 69–83
Lin, H. T., & Li, L. (2012). Reduction from cost-sensitive ordinal ranking to weighted binary classification. *Neural Computation, 24*(5), 1329–1367.

Liu, W., Xu, D., Tsang, I. W., & Zhang, W. (2018). Metric learning for multi-output tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 41*(2), 408–422.
Liu, Y., Kong, AWK., Goh, CK. (2017). Deep ordinal regression based on data relationship for small datasets. In: Proceedings of the 26th International Joint Conferences on Artificial Intelligence, pp 2372–2378
Ma, Z., & Chen, S. (2018). Multi-dimensional classification via a metric approach. *Neurocomputing, 275,* 1121–1131.
Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research, 9,* 2579–2605.
Mensch, A., Blondel, M., Peyré, G. (2019). Geometric losses for distributional learning. In: Proceedings of the 36th International Conference on Machine Learning, pp 4516–4525
Min, R., Stanley, DA., Yuan, Z., Bonner, A., Zhang, Z. (2009). A deep non-linear feature mapping for large-margin knn classification. In: 2009 Ninth IEEE International Conference on Data Mining, IEEE, pp 357–366
Muñoz, M. A., Sun, Y., Kirley, M., & Halgamuge, S. K. (2015). Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences, 317,* 224–245.
Nguyen, B., Morell, C., & De Baets, B. (2017). Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition, 64,* 215–225.
Nguyen, B., Morell, C., & De Baets, B. (2018). Distance metric learning for ordinal classification based on triplet constraints. *Knowledge Based Systems, 142,* 17–28.
Park, J. S. (1994). Optimal latin-hypercube designs for computer experiments. *Journal of Statistical Planning and Inference, 39*(1), 95–111.
Sánchez-Monedero, J., Pérez-Ortiz, M., Saez, A., Gutiérrez, P. A., & Hervás-Martínez, C. (2018). Partial order label decomposition approaches for melanoma diagnosis. *Applied Soft Computing, 64,* 341–355.
Sánchez-Monedero, J., Gutiérrez, P. A., & Pérez-Ortiz, M. (2019). Orca: A matlab/octave toolbox for ordinal regression. *Journal of Machine Learning Research, 20*(125), 1–5.
Shi, Y., Li, P., Yuan, H., Miao, J., & Niu, L. (2019). Fast kernel extreme learning machine for ordinal regression. *Knowledge Based Systems, 177,* 44–54.
Singer, G., Anuar, R., & Ben-Gal, I. (2020). A weighted information-gain measure for ordinal classification trees. *Expert Systems with Applications, 152*(113375).
Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
Suárez, J. L., García, S., & Herrera, F. (2020). pydml: A python library for distance metric learning. *Journal of Machine Learning Research, 21*(96), 1–7.
Suárez, J. L., García, S., & Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing, 425,* 300–322.
Tang, M., Pérez-Fernández, R., & De Baets, B. (2020). Fusing absolute and relative information for augmenting the method of nearest neighbors for ordinal classification. *Information Fusion, 56,* 128–140.
Torresani, L., & Lee, Kc. (2007) Large margin component analysis. In: Advances in Neural Information Processing Systems, pp 1385–1392
Triguero, I., González, S., Moyano, J. M., García, S., Alcalá-Fdez, J., Luengo, J., et al. (2017). Keel 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems, 10*(1), 1238–1249.
Vargas, V. M., Gutiérrez, P. A., & Hervás-Martínez, C. (2020). Cumulative link models for deep ordinal classification. *Neurocomputing, 401,* 48–58.
Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research, 10,* 207–244.
Xiao, B., Yang, X., Xu, Y., & Zha, H. (2009). Learning distance metric for regression by semidefinite programming with application to human age estimation. In: Proceedings of the 17th ACM International conference on Multimedia, ACM, pp 451–460

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Juan Luis Suárez[1] · **Salvador García**[1] · **Francisco Herrera**[1]

Salvador García
salvagl@decsai.ugr.es

Francisco Herrera
herrera@decsai.ugr.es

[1] Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, Granada 18071, Spain