Fusing absolute and relative information for augmenting the method of nearest neighbors for ordinal classification

Mengzi Tang[a], Raúl Pérez-Fernández[a,b], Bernard De Baets[a]

[a] *KERMIT, Department of Data Analysis and Mathematical Modelling, Ghent University, Coupure links 653, 9000 Ghent, Belgium.*
[b] *Department of Statistics and O.R. and Mathematics Didactics, University of Oviedo, C/ Federico García Lorca 18, 3307 Oviedo, Spain.*

Abstract

Ordinal classification is a special case of multiclass classification in which there exists a natural order on the set of class labels. Due to the nature of the problem, datasets for ordinal classification are typically rather small, having a negative impact on performance. A possible way out is to look for additional information. In this paper, firstly, we make use of order relations for unlabeled examples to generate relative information. Secondly, we incorporate this relative information into the method of k nearest neighbors, thus exploiting absolute and relative information at the same time. More specifically, we bring together notions from the fields of information fusion and machine learning to integrate both types of information. Finally, we test the proposed method on some classical machine learning datasets. The experimental results show the effectiveness of our approach.

Keywords: Information fusion, Absolute information, Relative information, k nearest neighbors, Ordinal classification.

[1] Mengzi Tang is supported by the China Scholarship Council (CSC). Raúl Pérez-Fernández acknowledges the support of the Research Foundation of Flanders (FWO17/PDO/160) and the Spanish MINECO (TIN2017-87600-P).

[2] Email addresses: mengzi.tang@ugent.be (Mengzi Tang), raul.perezfernandez@ugent.be (Raúl Pérez-Fernández), bernard.debaets@ugent.be (Bernard De Baets)

1. Introduction

A classical approach for addressing ordinal classification problems begins with a data collection step, where a large number of examples are described by their feature vectors and are associated with a class label. The information provided by these datasets formed by examples with an explicitly given class label is usually referred to as absolute information. Unfortunately, in real life, it might be a difficult task to collect such datasets because, typically, it is a time-consuming and costly process. Moreover, for a dataset with a small amount of absolute information, the performance of an ordinal classifier could be very low. In order to improve the performance, it might be useful to consider additional side information, which is commonly used in recommender systems, marketing services and bioinformatics [1]. For example, De Bie et al. [2] used side information to learn a suitable distance metric and proposed an algorithm for related clustering tasks. Jonschkowski et al. [3] also validated the importance of side information and proposed many approaches that could solve different machine learning tasks, such as multi-task learning, multi-view learning and learning using privileged information.

In food science and more specifically in sensory analysis studies, some typical examples of side information are the number of storage days of the food samples, related chemical analysis and sensory evaluation tests [4, 5]. A prominent type of side information is that of relative information, in which examples are compared one to another and then ordered. Fortunately, gathering a large amount of relative information is an easy task that can be performed by inviting some novices and gathering their preferences over some examples (such as pairwise orders among samples based on the freshness of food). The new challenge that arises is how to use a small amount of absolute information and a large amount of relative information at the same time for ordinal classification.

The method of k nearest neighbors (k-NN) is one of the most fundamental and well-known machine learning methods and has been widely used for classification [6], clustering [7] and regression [8] in various domains of application

such as financial modelling [9], image interpolation [10] and bioinformatics [11]. It is a non-parametric method [12] that assigns to a test example the most frequent class label among its k nearest neighbors. The first formulation of k-NN is usually attributed to Fix and Hodges [13], although the work of Cover and Hart [14] also contributed to the popularity of the method.

Recently, some efforts have been made to improve the traditional k-NN. For instance, it is now widely known that the classical Euclidean distance metric, which is oftentimes considered the standard distance metric for determining the nearest neighbors, might not be adequate for dealing with some given datasets. Thus it might be necessary to learn a more appropriate distance metric. It seems natural for the learned distance metric to assign small distances to examples with the same class label and large distances to examples with different class labels [15]. Nguyen et al. [16] considered the Mahalanobis distance metric in the context of distance metric learning. The experimental results show a higher performance when compared to other distance metric learning methods. As another example of recent interest in k-NN, Datta et al. [17] defined a penalty-based dissimilarity metric and incorporated it into k-NN. This approach not only improves the performance, but also allows to directly process data with missing information.

As a special case of multiclass classification, ordinal classification [18] has become a popular research topic that has been considered in, for instance, economical modelling [19], social sciences [20] and computer vision [21]. Common approaches for addressing ordinal classification problems could be divided into naive methods, ordinal binary decomposition methods and threshold methods [22]. Naive methods simply address ordinal classification problems as if they were standard classification problems. The order between class labels is simply ignored, thus possibly compromising the performance of the method. Ordinal binary decomposition methods decompose the ordinal variables into several binary ones, and subsequently address different independent tasks, ultimately combining all binary outputs into one class label. Threshold methods assume that there exists an underlying one-dimensional space and learn thresholds that

partition the real line into several intervals. Each interval is ultimately identified with a class label.

Notably, most current contributions only use absolute information and neglect the fact that there is usually a lot of relative information available. Also, learning from absolute information and learning from relative information [23, 24] are usually considered as two separate problems. Several works [25, 26] have recommended the fusion of both absolute and relative information to achieve a complete understanding of datasets and give accurate evaluations of examples. Therefore, it is necessary to develop some strategies to fuse different types of information into a single ordinal classification model. To the best of our knowledge, there are only few related works discussing this problem. For example, Sader et al. [27] proposed an ordinal regression model for combining absolute evaluations from experts and relative evaluations from novices in order to predict the class label of test examples. This proposal solves a constrained convex optimization problem that contains many parameters to learn, which makes the model complex and hard to explain. Therefore, there is a need to come up with some novel and simpler approaches that incorporate relative information, are easy to explain and have a good classification performance.

In this paper, we propose a new method for ordinal classification based on k-NN that fuses absolute and relative information. More specifically, we bring together notions from the fields of information fusion and machine learning to integrate both types of information. We test our method on some classical machine learning datasets. The experimental performance shows the importance of considering absolute and relative information and the usefulness of our method. The remainder of this paper is structured as follows. Section 2 recalls the classical method of k-NN for multiclass classification. Section 3 discusses k-NN within the context of ordinal classification. In Section 4, we provide a method of k-NN for ordinal classification with absolute and relative information. Experimental results and a corresponding analysis of these results are presented in Section 5. We end with some conclusions and open problems in Section 6.

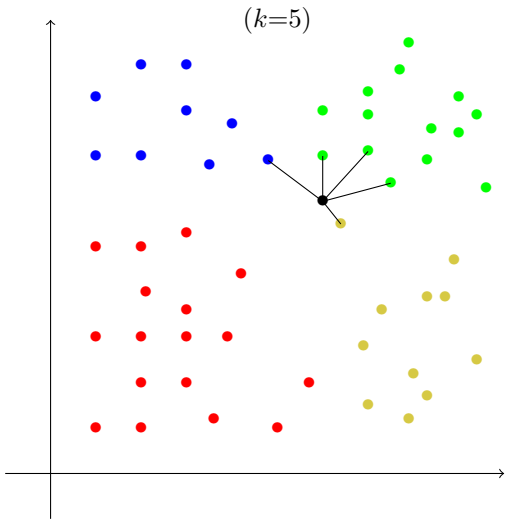


Figure 1: An example of multiclass classification using k-NN. Each color indicates a different class label. The class label of the black point is unknown. The solid black lines are connecting the black point with its five nearest neighbors.

2. k nearest neighbors for multiclass classification

The main principle of k-NN is that similar examples are distributed closely in the feature space. Thus the class label of a test example can be simply determined as the most frequent class label among those assigned to its k nearest neighbors [14, 28]. This process can be described by mapping the dataset onto a metric space. Typical distance metrics used are the Euclidean and Manhattan distance metrics [29].

Consider a set of input examples $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i = (x_{i1}, ..., x_{id})$ belongs to the input space $\mathcal{X} \subseteq \mathbb{R}^d$. The absolute information is denoted as $\mathcal{A} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, where the class labels y_i belong to the output space $\mathcal{Y} = \{C_1, C_2, ..., C_r\}$. We use the Euclidean distance metric to determine the nearest neighbors of a test example $\mathbf{x}^*$. We recall that the Euclidean distance $d(\mathbf{u}, \mathbf{v})$ between two examples $\mathbf{u}$ and $\mathbf{v}$ is computed as

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2} \ . \tag{1}$$

For the test example $\mathbf{x}^*$, its nearest neighbor $\mathbf{x}_{i_1}$ within the given set $\mathcal{D}$ is determined by

$$\arg\min_{\mathbf{x}_i \in \mathcal{D}} d(\mathbf{x}^*, \mathbf{x}_i) \ . \tag{2}$$

The second nearest neighbor $\mathbf{x}_{i_2}$ is the one that minimizes $d(\mathbf{x}^*, \mathbf{x}_i)$ aside of $\mathbf{x}_{i_1}$, and so on. By repeating this process, we find the k nearest neighbors gathered in a set $\mathcal{D}_k = \{\mathbf{x}_{i_j}\}_{j=1}^k$. We introduce the notation $\mathbf{y}^* = (y_{i_1}, y_{i_2}, ..., y_{i_k})$ for referring to the class labels associated with the k nearest neighbors. After finding all these neighbors, we need to aggregate their class labels by means of a function $f : \mathcal{Y}^k \to \mathcal{Y}$ that transforms $\mathbf{y}^*$ into a unique class label. A popular choice of such function is the mode. More precisely, the class label y^* for $\mathbf{x}^*$ is typically determined by computing the mode as follows:

$$y^* = f(\mathbf{y}^*) = \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^{k} \delta(y \neq y_{i_j}) \ , \tag{3}$$

where $\delta(y \neq y_{i_j})$ is the indicator function, which takes the value one if $y \neq y_{i_j}$ and zero if $y = y_{i_j}$. If there is more than one minimizer, we choose the class label y_{i_j} of the neighbor $\mathbf{x}_{i_j}$ that is associated with the smallest value of j among those verifying that y_{i_j} is a minimizer.

Example 1. *Consider the multiclass classification problem in Figure 1. We set $k = 5$ and compute the five nearest neighbors of the test example $\mathbf{x}^*$. The number of examples with the class label green (three) is greater than the number of examples with any other class label (zero or one). Thus, the class label green is assigned to the test example $\mathbf{x}^*$.*

From the example above, we can see that the value of k is very important since it might determine the performance of k-NN [30]. For example, if we would set $k = 1$ in Figure 1, the class label yellow would be assigned to $\mathbf{x}^*$ and we would get a totally different result. The choice of a good value of k depends on the data. In general, a large value of k can deal with noisy datasets, whereas a small value of k can avoid relying too strongly on far neighbors. The latter

mentioned problem could be avoided by considering a distance-weighted version of k-NN, originally proposed by Dudani [31]. The main idea of this distance-weighted method is to assign a larger weight to closer neighbors and a smaller weight to more distant neighbors. The weight w_j for the j-th nearest neighbor of $\mathbf{x}^*$ is defined as follows:

$$w_j = \begin{cases} \frac{d(\mathbf{x}^*,\mathbf{x}_{i_k}) - d(\mathbf{x}^*,\mathbf{x}_{i_j})}{d(\mathbf{x}^*,\mathbf{x}_{i_k}) - d(\mathbf{x}^*,\mathbf{x}_{i_1})} & , \quad \text{if } d(\mathbf{x}^*,\mathbf{x}_{i_k}) \neq d(\mathbf{x}^*,\mathbf{x}_{i_1}) \ , \\ 1 & , \quad \text{otherwise} \ . \end{cases} \tag{4}$$

The weight is inversely proportional to the distance between the test example and the corresponding example. The nearest neighbor is assigned the largest weight, $w_1 = 1$, while the farthest neighbor is assigned the smallest weight, $w_k = 0$. The weights for the other neighbors are scaled linearly. Finally, the class label y^* is determined using a weighted version of the mode:

$$y^* = f(\mathbf{y}^*) = \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^{k} w_j \, \delta(y \neq y_{i_j}) \ . \tag{5}$$

Other methods for selecting the weights have been proposed. For instance, Hechenbichler et al. [32] used a large variety of possible kernel functions to get different weighting schemes according to the distances to the nearest neighbors.

3. *k* nearest neighbors for ordinal classification

In real life, many classification problems actually come with a natural order on the set of class labels. For example, the freshness of food samples is usually evaluated as *spoiled* $\prec$ *marginal* $\prec$ *satisfactory* $\prec$ *fresh* $\prec$ *very fresh*, in which $\prec$ represents that the former class label is less preferred than the latter class label. In the case in which there is an underlying order on the set of class labels, the classical multiclass classification problem becomes an ordinal classification problem. Although the weighted mode probably is the most natural choice when there is no order associated with the considered set of class labels, when dealing with an ordinal classification problem, new meaningful families of aggregation

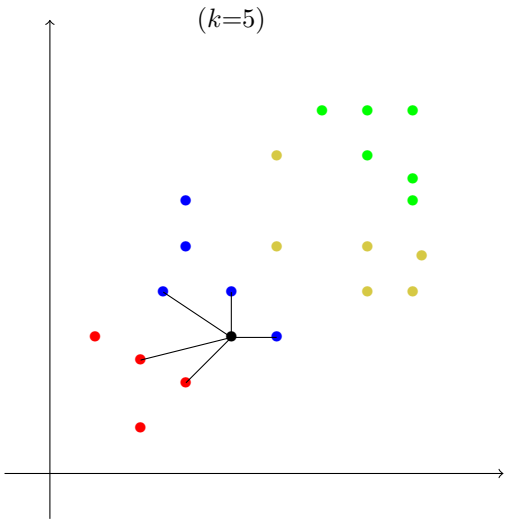


Figure 2: An example of ordinal classification using k-NN. Each color indicates a different class label. The following order is given: *red* $\prec$ *blue* $\prec$ *yellow* $\prec$ *green*. The class label of the black point is unknown. The solid black lines are connecting the black point with its five nearest neighbors.

functions should be considered, as will be explained next.

Assume that the underlying order on $\mathcal{Y} = \{C_1, C_2, ..., C_r\}$ is $C_1 \prec C_2 \prec ... \prec C_r$. Given a set of k nearest neighbors $\mathcal{D}_k = \{\mathbf{x}_{i_j}\}_{j=1}^{k}$ and associated class labels $\mathbf{y}^* = (y_{i_1}, y_{i_2}, ..., y_{i_k})$ with $\mathbf{y}^* \in \mathcal{Y}^k$, we need to select the class label $y^* \in \mathcal{Y}$ to be assigned to the test example $\mathbf{x}^*$. There is a large literature on how to solve such problem centered on the notion of an aggregation function. Aggregation functions have many applications in fuzzy systems, pattern recognition, information retrieval, to name just a few [33, 34]. Common aggregation functions for the case in which $\mathcal{Y}$ is a set of real numbers include weighted quasi-arithmetic means, ordered weighted averaging (OWA) operators and many others [35, 36, 37]. The literature is sparser in the setting in which an ordinal (linguistic) scale is considered and usually builds upon the notion of a median. Some interesting works by García-Lapresta et al. [38, 39] follow a different direction by introducing the notion of an ordinal proximity measure, which assigns an ordinal degree of proximity to each couple of class labels of the

ordinal scale.

In this paper, we consider the penalty-based aggregation approach proposed by Calvo and Beliakov [40]. We recall that, if minimal conditions are required for the penalty, penalty-based aggregation amounts to idempotent aggregation. Formally, a penalty-based function is defined by

$$f(\mathbf{y}^*) = \arg\min_{y \in \mathcal{Y}} P(\mathbf{y}^*, y) \ , \tag{6}$$

where $P : \mathcal{Y}^k \times \mathcal{Y} \to \mathbb{R}^+$ is a penalty function satisfying some desirable properties (see, e.g., [41, 42]).

Since they form the most prominent family of penalty functions, we restrict our attention to *faithful* penalty functions, i.e., functions P defined by

$$P(\mathbf{y}^*, y) = \sum_{j=1}^{k} w_j \, p(y_{i_j}, y) \ , \tag{7}$$

where the weights w_j could be defined as in the previous section and $p : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a (dissimilarity) function with the properties:

(1) $p(C, s) = 0$ if and only if $C = s$ and

(2) $p(C_1, s) \geq p(C_2, s)$ whenever $C_1 \geq C_2 \geq s$ or $C_1 \leq C_2 \leq s$.

A prominent example of faithful penalty function is given by the sum of the L_1-distances, where the L_1-distance $d_\perp$ between two class labels C_i and C_j is defined as $d_\perp(C_i, C_j) = |i - j|$. Note that the L_1-distance metric treats all labels of the ordinal scale as if they were equidistant, something that is not always advisable depending on the nature of $\mathcal{Y}$. The final aggregation function becomes

$$y^* = f(\mathbf{y}^*) = \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^{k} w_j \, d_\perp(y_{i_j}, y) \ . \tag{8}$$

The process above is equivalent to computing the median when identical weights $w_1 = ... = w_k$ are considered. A weighted version of the median arises when the weights proposed by Dudani (discussed in Section 2) are considered.

Example 2. *Consider the ordinal classification problem in Figure 2. We find*

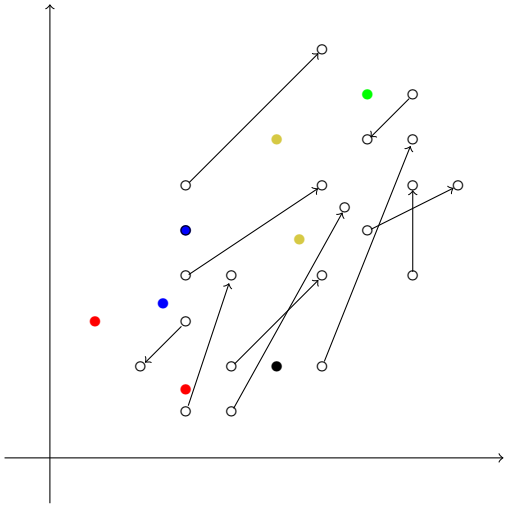


Figure 3: An example of ordinal classification with absolute and relative information. Each color indicates a different class label. The white examples connected by solid lines with an arrow only have relative information without a class label. The arrow represents that the example to which the arrow points is preferred to the example from which the arrow starts. The aim is to give a correct class label to the black example $\mathbf{x}^*$.

the five nearest neighbors of the test example. The median of the class labels of the five nearest neighbors is blue. Note that the minimizer of Eq.(8) *in case the weights proposed by Dudani are considered is also blue for this test example.*

Note that the method in this section is different from that presented in Section 2. In order to avoid any potential misunderstanding, we will refer to the method used in Section 2 in which Eq. (5) is considered as *weighted-mode-based k-NN*, and to the method used in Section 3 in which Eq. (8) is considered as *weighted-median-based k-NN*. It must be remarked that weighted-mode-based k-NN is the only possibility in case we are dealing with non-ordinal classification, whereas the use of weighted-median-based k-NN is encouraged in case we are dealing with ordinal classification.

4. k nearest neighbors for ordinal classification with absolute and relative information

4.1. Motivation and problem formulation

Classical ordinal classification problems only contain absolute information where the data includes examples with input features and associated ordinal class labels. However, in real-life datasets, there often exists relative information concerning the data without explicitly expressing a class label for all examples.

An example of such combined setting arises in the field of sensory analysis in food science, where experts typically describe the freshness of different food samples on an ordinal scale. However, because the number of experts is typically small and it is usually expensive to train additional experts on how to identify some spoilage indicators (and thus on how to properly use a given ordinal scale), only a small number of labeled examples (absolute information) is usually available. Fortunately, more data with relative information can be provided by untrained novices. This relative information can make up for the limitation regarding the few available absolute evaluations.

In order to make the motivation more clear, we use a toy example to illustrate our problem setting. In Figure 3, there are just a few labeled examples. The four classes of colored examples have been identified with four different class labels with the order *red* $\prec$ *blue* $\prec$ *yellow* $\prec$ *green*. The white examples connected by solid lines with arrows represent relative information concerning two unlabeled examples and express that the example to which the arrow points is preferred to the example from which the arrow starts. The aim is to assign a class label to the test example by also considering this latter type of information.

Formally, the data includes two different types of information: absolute information and relative information. The first type of information (absolute information) is collected in a set $\mathcal{A} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$ with a set of input examples $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where the input examples $\mathbf{x}_i = (x_{i1}, ..., x_{id})$ belong to the input space $\mathcal{X} \subseteq \mathbb{R}^d$ and the class labels y_i belong to the output space $\mathcal{Y} = \{C_1, C_2, ..., C_r\}$. The class labels are assumed to be ordered as

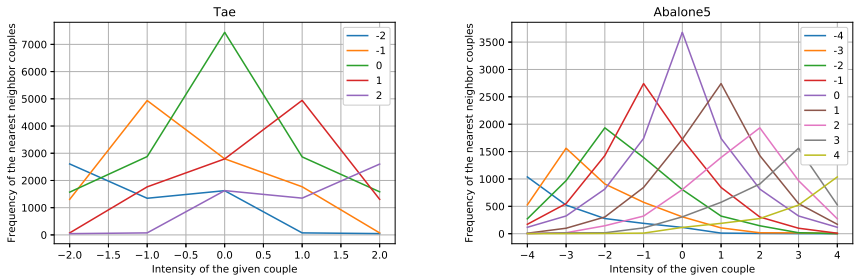
follows: $C_1 \prec C_2 \prec ... \prec C_r$.

We denote by $\mathcal{C} = \{(\mathbf{a}^1, \mathbf{b}^1), (\mathbf{a}^2, \mathbf{b}^2), ..., (\mathbf{a}^m, \mathbf{b}^m)\}$ the set of couples of $\mathcal{X}^2$ for which there is relative information available. For reasons that will become clear later on, it is assumed that if a couple $(\mathbf{a}, \mathbf{b})$ belongs to $\mathcal{C}$, then also the couple $(\mathbf{b}, \mathbf{a})$ belongs to $\mathcal{C}$. In addition, there is a pairwise order for each couple representing whether the first example in the couple is preferred to the second one or vice versa. Then, the second type of information (relative information) is collected in a set $\mathcal{R} = \{((\mathbf{a}^1, \mathbf{b}^1), R_1), ..., ((\mathbf{a}^m, \mathbf{b}^m), R_m)\}$, where an order relation $R_p = \prec$ indicates that $\mathbf{b}^p$ is preferred to $\mathbf{a}^p$ and an order relation $R_p = \succ$ indicates that $\mathbf{a}^p$ is preferred to $\mathbf{b}^p$, for any $p \in \{1, ..., m\}$. Note that here we do not consider the case in which both $\mathbf{a}^p$ and $\mathbf{b}^p$ are equally preferred. It is assumed that whenever $((\mathbf{a}, \mathbf{b}), R) \in \mathcal{R}$, it also holds that $((\mathbf{b}, \mathbf{a}), R^T) \in \mathcal{R}$, where R^T represents the transpose of R. The main characteristic of our problem is that the size of the absolute information is typically smaller than the size of the relative information, i.e., $n \ll m$.

4.2. Justification

In this subsection, we test how the intuition behind k-NN translates to the setting where datasets only include relative information. We construct this kind of dataset by comparing one-to-one all examples from a fully labeled dataset, thus generating a large amount of relative information. The intensity of a couple of examples is defined as the difference of the indices associated with the class labels of the examples of the couple. For instance, if the examples of the couple $(\mathbf{x}_i, \mathbf{x}_j)$ are labeled as $y_i = C_1$ and $y_j = C_3$, then the intensity of this couple is -2, and, similarly, the intensity of the couple $(\mathbf{x}_j, \mathbf{x}_i)$ is 2. If the dataset has N examples, we generate $N \cdot (N-1)$ couples. For each couple, we obtain the corresponding intensity.

In order to illustrate how these intensities are distributed, we subdivide all couples according to their intensity. Next, for each possible intensity, we



(a) (b)

Figure 4: Illustration of how the intensities of the nearest neighbor couples are distributed, the lines with different colors represent different intensities. For each intensity of the given couple, we compute the frequency of intensities of the nearest neighbor couples. (a) For the Tae dataset, the intensity of the couples ranges from -2 to 2. When the given couple and the nearest neighbor couples have the same intensity, the frequency is highest; (b) For the Abalone5 dataset, the intensity of the couples ranges from -4 to 4. When the given couple and the nearest neighbor couples have the same intensity, the frequency is highest.

iterate over the couples, respectively finding their ℓ nearest neighbor couples[3] and obtaining their corresponding intensities. We compute the frequency of each category of intensities for the associated neighbor couples. We select two datasets Tae and Abalone5 (more information on these datasets can be found in Table 1) and show how the intensities are distributed in Figure 4. For the Tae dataset, the intensity of the couples ranges from -2 to 2. For the cases in which the intensity of the couples is -2, the most frequent intensity among their nearest neighbor couples is -2. Similarly, for the cases in which the intensity of the couples is -1, the most frequent intensity among their nearest neighbor couples is -1. This phenomenon can also be observed for all other intensities. Note that the frequencies with which the intensities appear in the nearest neighbor couples decrease when moving away from the intensity of the selected couple. For the Abalone5 dataset, we randomly sample 100 examples from the original dataset and then generate all possible couples based on these examples. Repeating the process above, we observe a similar phenomenon. We conclude that nearest

[3]More details on how to compute nearest neighbor couples will be given in the upcoming subsection.

neighbor couples tend to have similar intensities as that of the couple of which they are neighbors. It thus seems natural to apply k-NN within this context.

4.3. Proposed method

Here, we propose a non-parametric method for ordinal classification that incorporates relative information and that can be understood and explained easily. Firstly, we find the k nearest neighbor examples $\mathcal{D}_k = \{\mathbf{x}_{i_j}\}_{j=1}^{k}$ of the test example $\mathbf{x}^*$. In order to exploit the relative information $\mathcal{R}$ in a simple manner, for every $j \in \{1, ..., k\}$, we see each couple $(\mathbf{x}^*, \mathbf{x}_{i_j})$ as a new object. Subsequently, we use the same idea behind k-NN to look for the ℓ nearest neighbor couples $\mathcal{R}_\ell^j = \{(\mathbf{a}_q^j, \mathbf{b}_q^j)\}_{q=1}^{\ell}$ of this new object $(\mathbf{x}^*, \mathbf{x}_{i_j})$. Note that we have an order relation $\prec$ or $\succ$ for each $(\mathbf{a}_q^j, \mathbf{b}_q^j)$.

For this process, we need to provide some technical details. When searching for the k nearest neighbor examples of $\mathbf{x}^*$, we use the Euclidean distance metric d (see Section 2). During the process of finding the ℓ nearest neighbor couples of $(\mathbf{x}^*, \mathbf{x}_{i_j})$, we compute the distance between couples according to the product metric (see [43], page 83, with $p = 1$), which is defined as

$$d_*((\mathbf{u}, \mathbf{v}), (\mathbf{w}, \mathbf{t})) = d(\mathbf{u}, \mathbf{w}) + d(\mathbf{v}, \mathbf{t}) \ . \tag{9}$$

In detail, the formula for computing the first nearest neighbor couple $(\mathbf{a}_1^j, \mathbf{b}_1^j)$ of $(\mathbf{x}^*, \mathbf{x}_{i_j})$ is as follows:

$$\underset{(\mathbf{a},\mathbf{b})\in\mathcal{C}}{\arg\min}\ d_*((\mathbf{x}^*, \mathbf{x}_{i_j}), (\mathbf{a}, \mathbf{b})) \ . \tag{10}$$

The second nearest neighbor couple is the one that minimizes $d_*((\mathbf{x}^*, \mathbf{x}_{i_j}), (\mathbf{a}, \mathbf{b}))$ aside of $(\mathbf{a}_1^j, \mathbf{b}_1^j)$, and so on. Note that both $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{b}, \mathbf{a})$ could be among the nearest neighbor couples and this is precisely why we made the technical assumption earlier that both always belong to $\mathcal{C}$. However, this is quite unlikely if $\mathbf{a}$ and $\mathbf{b}$ are distant from each other. By repeating this process ℓ times, we find the ℓ nearest neighbor couples $\mathcal{R}_\ell^j$ of $(\mathbf{x}^*, \mathbf{x}_{i_j})$.

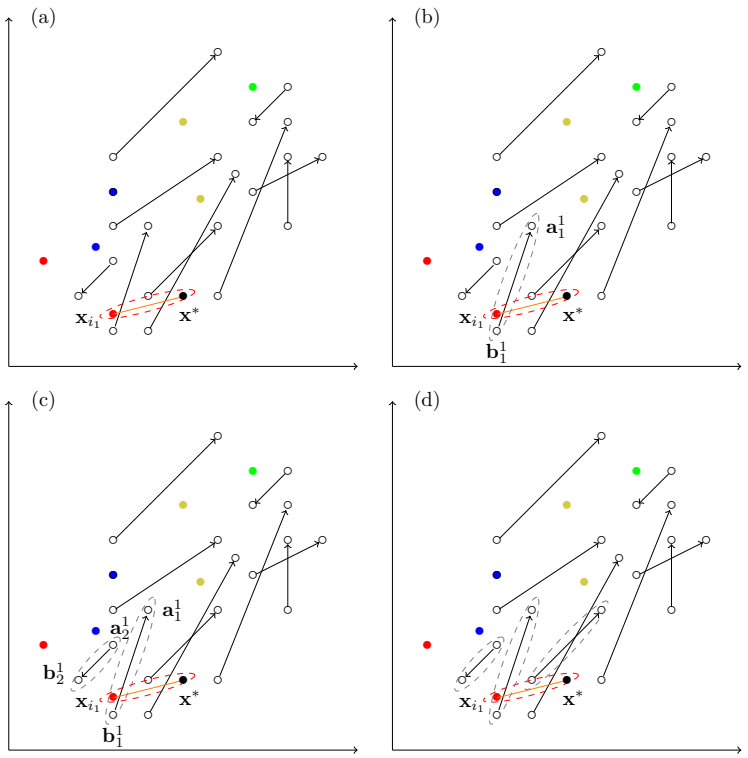


Figure 5: An example of the weighted-median-based method. (a) Find the nearest neighbor example $\mathbf{x}_{i_1}$ of $\mathbf{x}^*$; (b) Find the first nearest neighbor couple $(\mathbf{a}_1^1, \mathbf{b}_1^1)$ of $(\mathbf{x}^*, \mathbf{x}_{i_1})$; (c) Find the two nearest neighbor couples $(\mathbf{a}_1^1, \mathbf{b}_1^1)$ and $(\mathbf{a}_2^1, \mathbf{b}_2^1)$ of $(\mathbf{x}^*, \mathbf{x}_{i_1})$; (d) Find the three nearest neighbor couples of $(\mathbf{x}^*, \mathbf{x}_{i_1})$.

Secondly, similar relations are assumed for a couple and its nearest neighbor couples, which is justified by the result of Subsection 4.2. We thus get ℓ relations between $\mathbf{x}^*$ and $\mathbf{x}_{i_j}$ based on the ℓ nearest neighbor couples of $(\mathbf{x}^*, \mathbf{x}_{i_j})$. More specifically, if the given relation for the nearest neighbor couple $(\mathbf{a}_q^j, \mathbf{b}_q^j)$ is $\mathbf{a}_q^j \prec \mathbf{b}_q^j$, then the relation for the corresponding couple $(\mathbf{x}^*, \mathbf{x}_{i_j})$ is expected to be $\mathbf{x}^* \prec \mathbf{x}_{i_j}$. Similarly, if the given relation for the nearest neighbor couple $(\mathbf{a}_q^j, \mathbf{b}_q^j)$ is $\mathbf{a}_q^j \succ \mathbf{b}_q^j$, then the relation for the corresponding couple $(\mathbf{x}^*, \mathbf{x}_{i_j})$ is expected to be $\mathbf{x}^* \succ \mathbf{x}_{i_j}$. For example, in Figure 5(b), we obtain that the nearest neighbor couple of $(\mathbf{x}^*, \mathbf{x}_{i_1})$ is $(\mathbf{a}_1^1, \mathbf{b}_1^1)$. Because the given relation for $(\mathbf{a}_1^1, \mathbf{b}_1^1)$ is $\mathbf{a}_1^1 \succ \mathbf{b}_1^1$, the relation for $(\mathbf{x}^*, \mathbf{x}_{i_1})$ is expected to be $\mathbf{x}^* \succ \mathbf{x}_{i_1}$.

For each relation among these ℓ relations, we get an interval I_{jq} of potential class labels for $\mathbf{x}^*$. We define the interval I_{jq}, with $j \in \{1, ..., k\}$ and $q \in \{1, ..., \ell\}$, to assign the possible values of y^* for the q-th nearest neighbor couple of $(\mathbf{x}^*, \mathbf{x}_{i_j})$. If the given class label of $\mathbf{x}_{i_j}$ is C_c and we get that the relation for the couple $(\mathbf{x}^*, \mathbf{x}_{i_j})$ is $\mathbf{x}^* \prec \mathbf{x}_{i_j}$ according to its q-th nearest neighbor couple, the possible values of y^* would be $I_{jq} = [C_1, C_c]$. Note that the possible values of y^* also include C_c because the fact that an example is preferred (resp. not preferred) to another example does not imply that it should be classified with a greater (resp. smaller) class label. Obviously, it does imply that it should be classified with a greater (resp. smaller) or equal class label. Similarly, if the relation is $\mathbf{x}^* \succ \mathbf{x}_{i_j}$, then the possible values of y^* would be $I_{jq} = [C_c, C_r]$. We finally gather all the intervals $\mathbf{I} = \{I_{jq}\}_{j \in \{1,...,k\}, q \in \{1,...,\ell\}}$ according to all the nearest neighbor examples and all the corresponding nearest couples.

Thirdly, we consider the penalty-based function associated with the median for intervals discussed, for instance, by Beliakov et al. [44]:

$$P(\mathbf{I}, y) = \sum_{j=1}^{k} \sum_{q=1}^{\ell} (|l_{I_{jq}} - y| + |r_{I_{jq}} - y|) \ , \tag{11}$$

where $I_{jq} = [l_{I_{jq}}, r_{I_{jq}}]$. We introduce a distance-based weight for different cou-

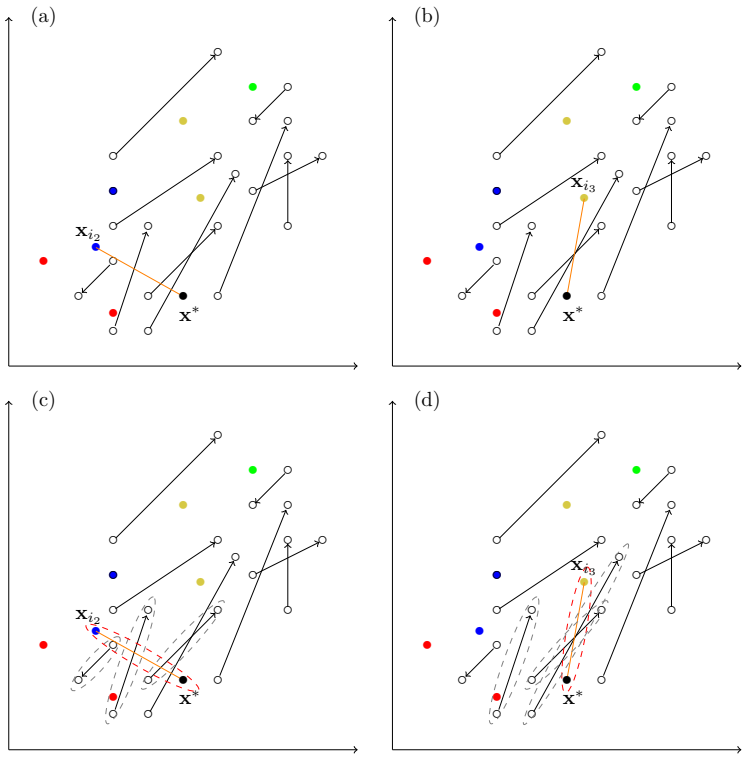


Figure 6: An example of the weighted-median-based method. (a) Find the second nearest neighbor example $\mathbf{x}_{i_2}$ of $\mathbf{x}^*$; (b) Find the third nearest neighbor example $\mathbf{x}_{i_3}$ of $\mathbf{x}^*$; (c) Find the three nearest neighbor couples of $(\mathbf{x}^*, \mathbf{x}_{i_2})$; (d) Find the three nearest neighbor couples of $(\mathbf{x}^*, \mathbf{x}_{i_3})$.

ples similarly to Eq. (4):

$$s_{jq} = \begin{cases} \frac{d_*((\mathbf{x}^*,\mathbf{x}_{i_j}),(\mathbf{a}_q^j,\mathbf{b}_q^j)) - d_*((\mathbf{x}^*,\mathbf{x}_{i_j}),(\mathbf{a}_1^j,\mathbf{b}_1^j))}{d_*((\mathbf{x}^*,\mathbf{x}_{i_j}),(\mathbf{a}_\ell^j,\mathbf{b}_\ell^j)) - d_*((\mathbf{x}^*,\mathbf{x}_{i_j}),(\mathbf{a}_1^j,\mathbf{b}_1^j))} & , \text{ if } d_*((\mathbf{x}^*,\mathbf{x}_{i_j}),(\mathbf{a}_\ell^j,\mathbf{b}_\ell^j)) \\ & \quad \neq d_*((\mathbf{x}^*,\mathbf{x}_{i_j}),(\mathbf{a}_1^j,\mathbf{b}_1^j)) \ , \\ 1 & , \text{ otherwise .} \end{cases} \tag{12}$$

The class label y^* of $\mathbf{x}^*$ is then determined using an appropriate aggregation function:

$$y^* = f(\mathbf{y}^*) = \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^{k} w_j \sum_{q=1}^{\ell} s_{jq}(|l_{I_{jq}} - y| + |r_{I_{jq}} - y|) \ , \tag{13}$$

where the weights w_j are defined as in Eq. (4). Here, if we get more than one minimizer, we do not get a unique class label for the test sample. It seems natural to consider the frequency of class labels from the absolute information as a possible tie-breaker and, thus, assign the most frequent class label (among those that are minimizers of Eq. (13)) to the test example. However, such case is not likely to occur in our setting.

To summarize, our proposed method can be described in the following algorithm.

Algorithm 1 The proposed algorithm

Input: The absolute information $\mathcal{A}$, the relative information $\mathcal{R}$, the parameters k, ℓ and the test example $\mathbf{x}^*$
  for $j = 1 : k$ **do**
    Find the nearest neighbor example $\mathbf{x}_{i_j}$ of $\mathbf{x}^*$ and compute the weight w_j;
    for $q = 1 : \ell$ **do**
      Find the nearest neighbor couple $(\mathbf{a}_q^j, \mathbf{b}_q^j)$ of $(\mathbf{x}^*, \mathbf{x}_{i_j})$;
      Compute the interval I_{jq} and the weight s_{jq};
    end for
  end for
  Compute the class label y^*
Output: The class label y^*

Example 3. *Consider the problem of ordinal classification with absolute and relative information in Figure 3. We set both k and ℓ equal to three. Firstly, in*

Figure 5, for the test example $\mathbf{x}^*$, we find the first nearest neighbor example $\mathbf{x}_{i_1}$ and compute the weight w_1. For the couple $(\mathbf{x}^*, \mathbf{x}_{i_1})$, we find the corresponding three nearest neighbor couples $(\mathbf{a}_1^1, \mathbf{b}_1^1)$, $(\mathbf{a}_2^1, \mathbf{b}_2^1)$, $(\mathbf{a}_3^1, \mathbf{b}_3^1)$, and compute the corresponding weights s_{11}, s_{12}, s_{13}. From the example in Figure 5(b), we get that the order relation for the couple $(\mathbf{x}^*, \mathbf{x}_{i_1})$ is expected to be $\mathbf{x}^* \succ \mathbf{x}_{i_1}$. From this figure, we can see that the class label of $\mathbf{x}_{i_1}$ is C_1 (here C_1 is red, C_2 is blue, C_3 is yellow and C_4 is green), and we get the interval $I_{11} = [C_1, C_4]$ for y^*. Subsequently, we repeat the process to get I_{12} and I_{13} for the other two nearest neighbor couples $(\mathbf{a}_2^1, \mathbf{b}_2^1)$ and $(\mathbf{a}_3^1, \mathbf{b}_3^1)$ of $(\mathbf{x}^*, \mathbf{x}_{i_1})$. Specifically, the order relation for the second neighbor couple $(\mathbf{a}_2^1, \mathbf{b}_2^1)$ is $\mathbf{a}_2^1 \prec \mathbf{b}_2^1$, thus the order relation for the couple $(\mathbf{x}^*, \mathbf{x}_{i_1})$ is expected to be $\mathbf{x}^* \prec \mathbf{x}_{i_1}$. We get the interval $I_{12} = \{C_1\}$. Similarly, we get the interval $I_{13} = [C_1, C_4]$. Secondly, in Figure 6, for the second and third neighbors $\mathbf{x}_{i_j}$, we repeat the same process to find all the corresponding nearest neighbor couples $(\mathbf{a}_q^j, \mathbf{b}_q^j)$ and compute all the corresponding weights. If the given class label of $\mathbf{x}_{i_j}$ is C_c and we get the order relation $\mathbf{x}^* \prec \mathbf{x}_{i_j}$, then we assign $I_{jq} = [C_1, C_c]$. Otherwise if we get the order relation $\mathbf{x}^* \succ \mathbf{x}_{i_j}$, then we assign $I_{jq} = [C_c, C_r]$. Finally, by computing Eq. (13) and assuming all weights are set to one, we obtain the class label $y^* = C_1$.

5. Experiments

In this section, we describe the datasets, introduce the performance measures and analyze the performance.

5.1. Datasets

We perform our experiments on some datasets from real ordinal classification problems and other datasets from discretized regression problems. The first kind of datasets are from some open repositories, i.e. the UCI machine learning repository [45] and mldata.org [46]. The second kind of datasets are collected by Chu [47]. In real-life ordinal classification problems, data usually needs to be collected by experts, so here the size of the datasets of the first type is small.

Table 1: Description of the benchmark datasets.

Dataset	#Examples	#Features	#Classes
Real ordinal classification datasets			
Tae (*TA*)	151	54	3
Automobile (*AU*)	205	26	6
Balance-scale (*BS*)	625	4	3
Eucalyptus (*EU*)	736	91	5
Red-wine (*RW*)	1599	12	6
Car (*CA*)	1728	21	4
Discretized regression datasets			
*Housing*5 (*HO*5)	506	14	5
*Abalone*5 (*AB*5)	4177	11	5
*Bank*1-5 (*BA*1-5)	8192	8	5
*Bank*2-5 (*BA*2-5)	8192	32	5
*Computer*1-5 (*CO*1-5)	8192	12	5
*Computer*2-5 (*CO*2-5)	8192	21	5
*Housing*10 (*HO*10)	506	14	10
*Abalone*10 (*AB*10)	4177	11	10
*Bank*1-10 (*BA*1-10)	8192	8	10
*Bank*2-10 (*BA*2-10)	8192	32	10
*Computer*1-10 (*CO*1-10)	8192	12	10
*Computer*2-10 (*CO*2-10)	8192	21	10

The datasets from discretized regression problems are larger, they are collected by discretizing the examples into ordinal classes with equal frequency. Table 1 describes the characteristics of these datasets, including the number of examples, features and classes. All the features have been properly standardized to avoid the impact of the scale of features. We use ten-fold cross-validation to obtain the performance of our methods.

Note that these datasets do not contain relative information. In the experiments, we generate relative information from absolute information. The generation process is described in Figure 7. Specifically, for each dataset, firstly, we subdivide it into ten folds. One of the folds (the red part in the figure) is used for testing. The remaining folds (the yellow and blue parts) are used for collecting absolute and relative information. Secondly, we use a parameter to adjust the percentage of absolute information. We randomly select a part (the yellow part) corresponding to this fixed percentage and keep it unchanged as absolute information. We use the other part (the blue part) for generating relative information (the green part) by transforming the labels into order relations

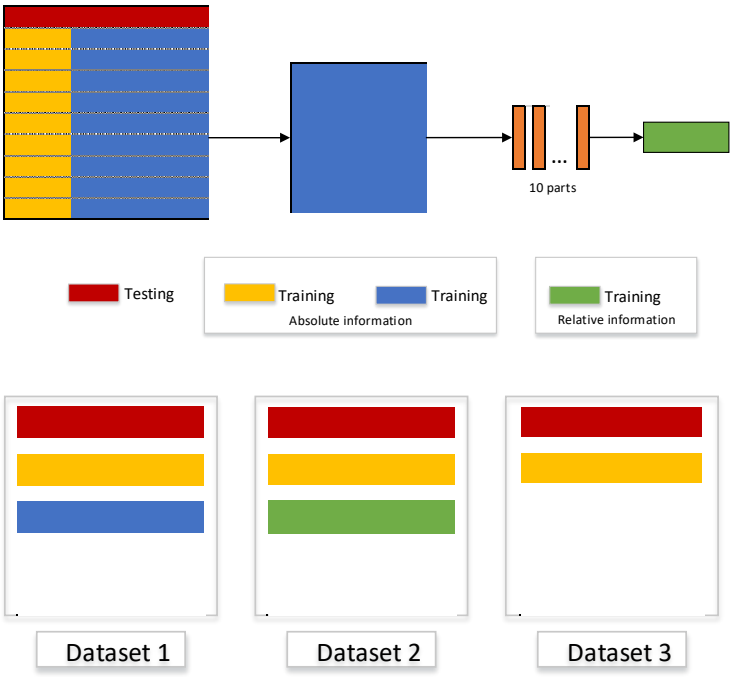


Figure 7: Process of generating relative information from absolute information. The top part of the figure shows the process of generating relative information from the original data. We divide the original dataset into ten folds (marked as ten different rows). One of these folds is used for testing (colored in red) and the other nine folds are used for training. Next, we divide each of the folds into two parts (one part colored in yellow and another one in blue). The part colored in yellow is kept as absolute information whereas the part colored in blue is transformed into relative information. This is done by dividing the part colored in blue into ten parts of equal size and comparing every two examples row by row. The bottom part of the figure describes three new datasets used for validating our method. The red part used for testing and the yellow part used for training remain the same in all three datasets. Additional information for training is considered in Dataset 1 in the form of absolute information and in Dataset 2 in the form of relative information.

between examples. More specifically, we randomly divide the blue part into ten parts of equal length (the ten orange parts in the top part of the figure). For every row, we compare the class labels of every two examples of the ten parts to obtain the corresponding order relations. For instance, if there are two training examples and their class labels are $y_1 = C_1$ and $y_2 = C_2$ with the order relation $C_1 \prec C_2$, then we will generate two couples $(\mathbf{x}_1^1, \mathbf{x}_2^1)$ with $\mathbf{x}_1^1 \prec \mathbf{x}_2^1$ and $(\mathbf{x}_2^1, \mathbf{x}_1^1)$ with $\mathbf{x}_2^1 \succ \mathbf{x}_1^1$. Note that if two examples have the same class label, no couples are generated. Note that every example is involved in at most nine order relations. Due to this way of generating relative information, many of the generated couples are chained, meaning that they share one of their examples. Note that, this approach reduces the complexity of generating relative information – something that is especially important when dealing with large datasets.

Ultimately, in order to validate our method, for each original dataset, we construct three different datasets, as shown in Figure 7. The test data (the red part) is the same for the three datasets. The remaining 90% is used for creating different training datasets. *Dataset 1* keeps all the absolute information (the yellow and blue parts), whereas *Dataset 3* only keeps a small part of the absolute information (the yellow part). *Dataset 2* not only includes the absolute information of *Dataset 3* (the yellow part), but also contains the relative information (the green part). By comparing the performance on *Dataset 1* and *Dataset 3*, we test the difference of using different amounts of absolute information. By comparing the performance on *Dataset 2* and *Dataset 3*, we test the impact of incorporating relative information. We expect the performance on *Dataset 1* to be the best and the performance on *Dataset 3* to be the worst, whereas the performance on *Dataset 2* is expected to be placed in between the performances on *Dataset 1* and *Dataset 3*.

5.2. Performance measures

There are many performance measures used for evaluating ordinal classification models [48, 49]. Here, we choose the three most common performance measures, the Mean Zero-one Error (MZE), the Mean Absolute Error (MAE)

and the C-index. The MZE describes the error rate of the classifier computed as:

$$\text{MZE} = \frac{1}{T}\sum_{i=1}^{T}\delta(y_i^* \neq y_i) = 1 - \text{Acc} \ , \tag{14}$$

where T is the number of test examples, y_i is the observed class label and y_i^* is the predicted class label. Acc is the accuracy of the classifier. The value of MZE ranges from 0 to 1. It describes the global performance, but it neglects the relations among the class labels.

The MAE is the average absolute error between the observed class label and the predicted class label. If the class labels are represented by numbers, the MAE is computed as:

$$\text{MAE} = \frac{1}{T}\sum_{i=1}^{T}|y_i - y_i^*| \ . \tag{15}$$

The value of MAE ranges from 0 to $r-1$ (maximum absolute error between classes). Because the real distances among the class labels are unknown, the numerical representation of the class labels has a strong impact on the MAE performance.

In order to avoid the above-mentioned impact, a more suitable approach is to consider the relation between the observed class label and the predicted class label. Here we use the concordance index or C-index to represent these relations. The C-index is computed as the proportion of the number of concordant pairs to the number of comparable pairs (see [50], page 50):

$$\text{C-index} = \frac{1}{\sum_{C_p \prec C_q} T_{C_p} T_{C_q}} \sum_{y_i \prec y_j} \left(\delta(y_i^* \prec y_j^*) + \frac{1}{2}\delta(y_i^* = y_j^*)\right) \ , \tag{16}$$

where T_{C_p} and T_{C_q} are the numbers of test examples with the class label C_p and C_q, respectively, y_i and y_i^* are the observed and predicted class label of $\mathbf{x}_i$ and y_j and y_j^* are the observed and predicted class label of $\mathbf{x}_j$. When there are only two different class labels, the C-index amounts to the area under the Receiver Operating Characteristic (ROC) curve [51] and ranges from 0 to 1.

A lower MZE or MAE, or a higher C-index indicates a better performance.

Table 2: Performances of the weighted-mode-based (Majority) method and the weighted-median-based (Median) method on the benchmark datasets. The best results are highlighted in boldface.

Id	MZE		MAE		$1 - $ C-index	
	Majority	Median	Majority	Median	Majority	Median
TA	0.4268	**0.4189**	0.5368	**0.5210**	0.2817	**0.2807**
AU	**0.2847**	0.2876	0.4123	**0.4113**	0.1421	**0.1393**
BS	**0.1593**	0.1849	**0.1967**	0.2111	**0.0808**	0.0964
EU	0.4842	**0.4820**	0.6385	**0.6273**	0.1823	**0.1814**
RW	0.3472	**0.3430**	0.4074	**0.3993**	**0.2291**	0.2629
CA	0.1608	**0.1552**	0.2168	**0.1906**	**0.0882**	0.1485
HO5	0.3755	**0.3624**	0.4442	**0.4201**	0.1144	**0.1091**
AB5	0.5823	**0.5761**	0.8514	**0.8113**	0.2404	**0.2298**
BA1-5	**0.6279**	0.6276	0.8972	**0.8549**	0.2493	**0.2408**
BA2-5	**0.7582**	0.7580	1.4072	**1.3430**	0.4378	**0.4337**
CO1-5	0.3577	**0.3519**	0.4029	**0.3919**	0.1010	**0.0981**
CO2-5	0.3146	**0.3074**	0.3443	**0.3331**	0.0849	**0.0823**
HO10	0.6209	**0.5925**	0.9706	**0.9089**	0.1212	**0.1145**
AB10	0.7693	**0.7611**	1.8192	**1.7043**	0.2529	**0.2391**
BA1-10	**0.7961**	0.7987	1.887	**1.7678**	0.2625	**0.2482**
BA2-10	**0.8744**	0.8757	2.9375	**2.7712**	0.4469	**0.4394**
CO1-10	0.5683	**0.5563**	0.8605	**0.8148**	0.1063	**0.1002**
CO2-10	0.5293	**0.5139**	0.7440	**0.6978**	0.0896	**0.0835**
p-value	0.01762		5.35672e-4		0.05264	

For ease of presentation, we replace C-index by $(1 - \text{C-index})$. In this way, a lower MZE, MAE or $1 - $ C-index represents a better performance.

5.3. Performance analysis

In this subsection, we analyze the performance of the different methods discussed in this paper. All the experimental results are obtained by applying ten-fold cross validation.

5.3.1. Comparing the weighted-mode-based and the weighted-median-based methods

We set the number k of nearest neighbor examples as 5 and carry out experiments on all datasets from real ordinal classification problems and discretized regression problems. Table 2 shows the performances of the *weighted-mode-based k-NN* (see Eq. (5)) and the *weighted-median-based k-NN* (see Eq. (8)). It is clear

that the *weighted-median-based k-NN* performs better on almost all datasets. In order to test whether there is a significant difference in performance between the two methods, we perform the Wilcoxon signed-rank test [52] at a significance level of $\alpha = 0.05$. If the p-value is smaller than the fixed significance level of α, then it means that there exists a statistically significant difference between these two methods. From Table 2, we can see that the p-values for MZE and MAE are smaller than α. The results show that the *weighted-median-based k-NN* outperforms the *weighted-mode-based k-NN* for these two performance measures. Note that the p-value for $1 - $ C-index is almost equal to α.

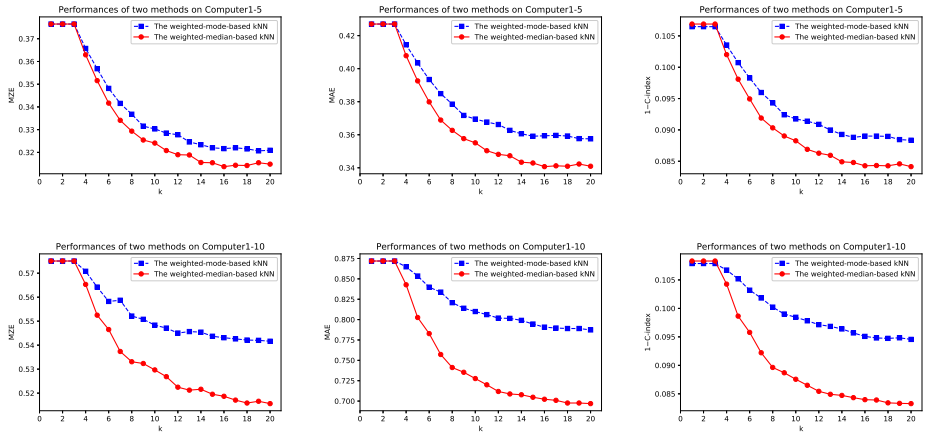


Figure 8: MZE, MAE and $1 - $ C-index of the weighted-mode-based and weighted-median-based methods with different numbers of nearest neighbor examples k on the *Computer1-5* and *Computer1-10* datasets.

In addition, we compare the performance of the *weighted-median-based k-NN* and that of the *weighted-mode-based k-NN* when the number k of nearest neighbor examples increases. Figure 8 shows the MZE, MAE and $1 - $ C-index performances of these two methods with a varying number of nearest neighbor examples on the *Computer1-5* and *Computer1-10* datasets. When the number k of nearest neighbor examples increases, the gap between the performances increases as well. Besides, when the number of class labels is larger, which means that there is a richer ordinal scale, the gap between the performances of these two methods also becomes larger. This is due to the fact that the *weighted-*

median-based k-NN takes advantage of the information within an ordinal scale, whereas the *weighted-mode-based k-NN* ignores this useful information.

5.3.2. Influence of using absolute and relative information

In order to demonstrate the importance of absolute information and relative information and show the impact of using different percentages of this information, for each dataset, we create three different new datasets (as shown in Figure 7 and described in Subsection 5.1) with different amounts of absolute information and relative information to test our proposed method. We fix the number k of nearest neighbor examples at 5 and the number ℓ of nearest neighbor couples at 5. We initially set the percentage of absolute information as 5%.

Table 3 shows the experimental results on all datasets. For each original dataset, we test our method on the three different new datasets: *Dataset 1*, *Dataset 2* and *Dataset 3*. It can be seen that for *Dataset 1* we have the best performance and for *Dataset 3* we have the worst performance, which shows that using a large amount of absolute information leads to a better performance than using a small amount of absolute information. Importantly, from the table, we get an order of the performances on these three datasets: *Dataset 1* > *Dataset 2* > *Dataset 3*. For the three different performance measures, the performances on almost all the datasets respect this ordering, which corresponds to our initial intuition. This validates the importance of incorporating relative information.

In order to test whether there is a significant difference in performance on the three different datasets, we perform the Friedman test [53] and the Wilcoxon signed-rank test at a significance level of $\alpha = 0.05$, shown in Table 4. The results show that all p-values are smaller than α, which means that there exists a statistically significant difference between the performance on the three constructed datasets obtained from all original datasets.

Besides, we change the percentage of absolute information and range it from 5% to 50%. The performances on the *Bank1-5* and *Bank1-10* datasets are used for illustrative purposes and shown in Figure 9. For lower percentages,

Table 3: Performances on three new constructed datasets for each original dataset. The results with an inconsistent order are highlighted in boldface.

Dataset		MZE	MAE	1 − C-index	Dataset		MZE	MAE	1 − C-index	Dataset		MZE	MAE	1 − C-index
TA	Dataset 1	0.4033	0.4959	0.2751	AU	Dataset 1	0.2714	0.3519	0.1207	BS	Dataset 1	0.1871	0.2094	0.0953
	Dataset 2	0.5092	0.5822	0.3332		Dataset 2	0.5395	0.6919	0.2104		Dataset 2	0.1954	0.2771	0.1382
	Dataset 3	0.5959	0.7664	0.4549		Dataset 3	0.6047	0.9410	0.3804		Dataset 3	0.2366	0.3549	0.1817
EU	Dataset 1	0.4825	0.6388	0.1861	RW	Dataset 1	0.3527	0.4070	0.2348	CA	Dataset 1	0.1556	0.1904	**0.1489**
	Dataset 2	0.5747	0.8488	0.2557		Dataset 2	0.4641	0.5378	0.2987		Dataset 2	0.1759	0.2112	**0.1488**
	Dataset 3	0.6383	1.0267	0.3275		Dataset 3	0.4997	0.5757	0.3338		Dataset 3	0.2269	0.2778	**0.2440**
HO5	Dataset 1	0.3757	0.4392	0.1118	AB5	Dataset 1	**0.5767**	0.8149	0.2314	BA1−5	Dataset 1	0.6304	0.8605	0.2422
	Dataset 2	0.4551	0.5817	0.1609		Dataset 2	**0.5947**	0.8541	0.2459		Dataset 2	0.6660	0.9957	0.2974
	Dataset 3	0.5496	0.7456	0.2085		Dataset 3	**0.5937**	0.8547	0.2515		Dataset 3	0.7187	1.1602	0.3622
BA2−5	Dataset 1	**0.7616**	1.3434	0.4328	CO1−5	Dataset 1	0.3516	0.3926	0.0981	CO2−5	Dataset 1	0.3093	0.3347	0.0826
	Dataset 2	**0.7553**	1.3594	0.4390		Dataset 2	0.3885	0.4592	0.1192		Dataset 2	0.3524	0.4068	0.1057
	Dataset 3	**0.7761**	1.4203	0.4615		Dataset 3	0.4159	0.4856	0.1243		Dataset 3	0.3727	0.4217	0.1064
HO10	Dataset 1	0.5960	0.9332	0.1179	AB10	Dataset 1	0.7629	1.7013	0.2396	BA1−10	Dataset 1	0.7986	1.7653	0.2476
	Dataset 2	0.7164	1.3757	0.1932		Dataset 2	0.7685	1.7592	0.2484		Dataset 2	0.8363	2.1106	0.3128
	Dataset 3	0.7921	1.6814	0.2367		Dataset 3	0.7750	1.7603	0.2531		Dataset 3	0.8699	2.4688	0.3796
BA2−10	Dataset 1	0.8716	2.7594	0.4384	CO1−10	Dataset 1	0.5525	0.8026	0.0986	CO2−-10	Dataset 1	0.5145	0.7029	**0.0843**
	Dataset 2	0.8749	2.7751	0.4390		Dataset 2	0.6038	0.9838	0.1260		Dataset 2	0.5642	0.8658	**0.1101**
	Dataset 3	0.8850	2.8525	0.4535		Dataset 3	0.6141	1.0158	0.1294		Dataset 3	0.5853	0.8784	**0.1080**

Table 4: p-values according to the Friedman test and the Wilcoxon test over all datasets based on the three different performance measures.

Test	Method	MZE	MAE	$1-$ C-index
Friedman	Dataset 1, Dataset 2 and Dataset 3	3.9277e-08	5.6028e-08	3.9277e-08
Wilcoxon	Dataset 1 and Dataset 2	2.5022e-04	1.3183e-04	1.5510e-04
	Dataset 2 and Dataset 3	1.5510e-04	1.3183e-04	1.1383e-04
	Dataset 1 and Dataset 3	1.3183e-04	1.3183e-04	1.8218e-04

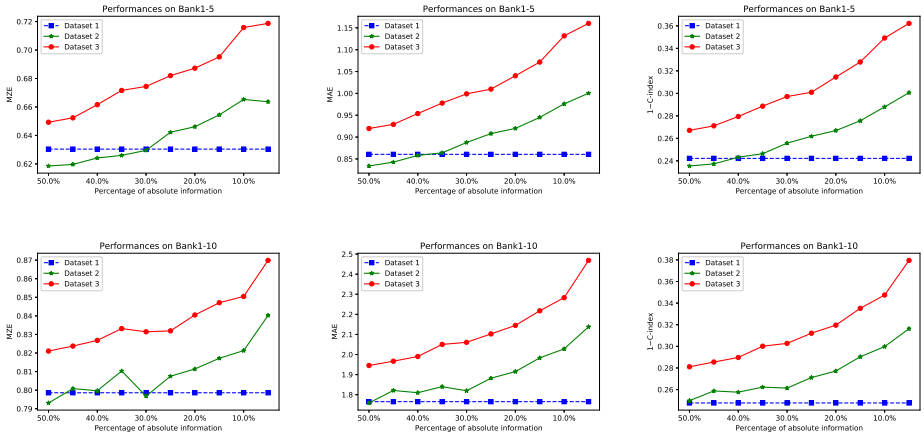


Figure 9: MZE, MAE and $1-$C-index of three constructed datasets with different percentages of absolute information on the *Bank1-5* and *Bank1-10* datasets.

the MZE, MAE and $1-$C-index performances are worse. Obviously, this is because the amount of absolute information is reduced. The performance on *Dataset 2* is always better than the performance on *Dataset 3*, which means that using relative information is meaningful. Besides, there is also an interesting finding that when the percentage of absolute information is high (close to 50%), the performance on *Dataset 2* is better than the performance on *Dataset 1*, which means that using a small amount of absolute information and relative information together outperforms only using absolute information. This shows that relative information could replace absolute information to some extent.

5.3.3. Sensitivity to noisy relative information

In the above experiments, we have generated relative information from absolute information assuming that relative information is as reliable as absolute

information. However, in real life, relative information is usually less reliable, because it is provided by novices who are not as proficient as experts. Here, in order to evaluate the sensitivity to noisy relative information, we artificially introduce different types of noise (for a given noise percentage) to the relative information and analyze the impact on the performance of our method. For this purpose, based on the process for generating *Dataset 2* in Subsection 5.1, we generate four datasets.

The first dataset *D1* is the same as *Dataset 2*. The second dataset *D2* includes the same relative information as *Dataset 2* and additional noisy couples generated from examples with identical class labels, which had been ignored for generating *Dataset 2*. For instance, for a noise percentage of 10%, 10% of the couples that would have been generated from examples with identical class labels are assigned a random order. The third dataset *D3* includes the same generated relative information as *Dataset 2*, however, the order relations of some couples are changed when these couples had been generated from examples with neighboring class labels. For instance, for a noise percentage of 10%, 10% of the couples that had been generated from examples with neighboring class labels are randomly chosen to change their order relations. The fourth dataset *D4* includes the same relative information as dataset *D3* with the addition of the noisy couples generated for dataset *D2*.

We perform the same experiment on the four datasets as in the previous subsection by fixing the percentage of absolute information at 50%. We set the percentage of noise for the relative information as 10%, 20%, 30%, 40% or 50%. The corresponding noisy couples for *D2*, *D3* and *D4* are generated by incrementally adding more noisy couples according to the aforementioned percentages of noise. We repeat the experiments 10 times. The average performances on the four newly constructed datasets for *Tae*, for example, are shown in Figure 10. The performance on *D1* obviously does not change while varying the percentage of noise. The performances on *D2*, *D3* and *D4* are worse than the performance on *D1*, and continue to get worse when increasing the percentage of noise. Obviously, this is because the relative information becomes less

reliable as the noise percentage increases. The performance on *D2* is better and worsens less quickly than the performances on *D3* and *D4*. This shows that our method is less sensitive to the first type of noise (introduced for examples with identical class labels) and more sensitive to the second type of noise (introduced for examples with neighboring class labels), which perfectly makes sense.

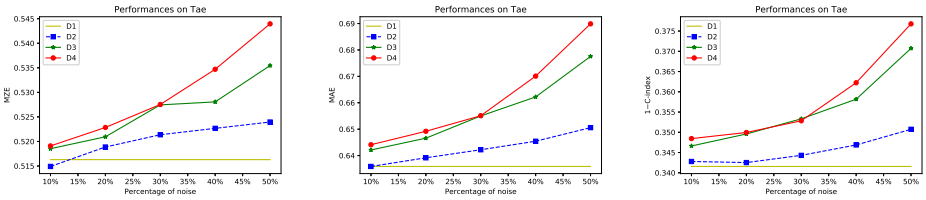


Figure 10: MZE, MAE and $1-$ C-index of the four constructed datasets with different types of noisy relative information for different percentages of noise for *Tae*.

5.3.4. Influence of the values k and ℓ

In order to analyze the influence of the number k of nearest neighbor examples and the number ℓ of nearest neighbor couples, we perform experiments on datasets with varying k and ℓ. We set the percentage of absolute information as 5%.

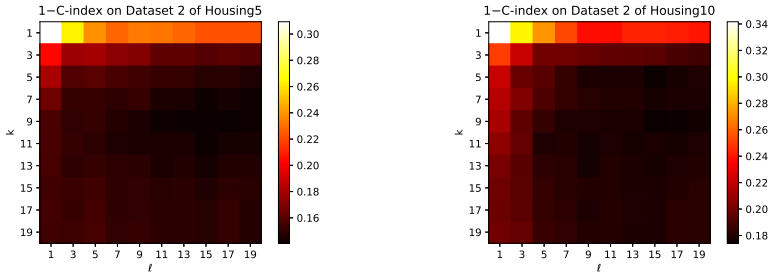


Figure 11: $1-$ C-index of Dataset 2 for different values of k and ℓ on the *Housing5* and *Housing10* datasets.

First, we vary k and ℓ, ranging from 1 to 20. In Figure 11, we plot the heatmap of performances on the *Dataset 2* of the *Housing5* and *Housing10* datasets. It can be seen that, while increasing the number k of nearest neigh-

bor examples and the number ℓ of nearest neighbor couples, the performance becomes better because of considering more neighborhood information.

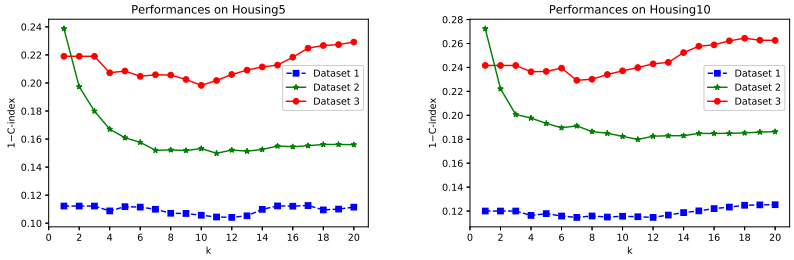


Figure 12: $1 -$ C-index of three constructed datasets with a different number k of nearest neighbor examples on the *Housing5* and *Housing10* datasets.

Second, we fix the number ℓ of nearest neighbor couples at 5 and vary the number k of nearest neighbor examples, ranging from 1 to 20. The performances on the *Housing5* and *Housing10* datasets are shown in Figure 12. It can be seen that, while increasing k, the performance remains relatively stable to some extent for *Dataset 1*, probably because there is a large amount of absolute information that provides enough class labels in the neighborhood. Moreover, the performance decreases as k increases for *Dataset 3*, probably because there is only a small amount of absolute information and the number k of nearest neighbor examples has a big impact for this dataset. Increasing k for *Dataset 2* initially results in an increasing performance. Once the number k of nearest neighbor examples is sufficiently large, the performance becomes stable, which confirms that considering a local neighborhood allows to obtain a good performance.

Third, we fix the number k of nearest neighbor examples at 5 and vary the number ℓ of nearest neighbor couples, ranging from 1 to 20. The performances on the *Housing5* and *Housing10* datasets are shown in Figure 13. While varying the number ℓ of nearest neighbor couples, it can be seen that the performances for *Dataset 1* and *Dataset 3* remain constant. This is obvious since we use the classical k-NN on these two datasets without taking relative information into account. For *Dataset 2*, increasing the number of nearest neighbor couples

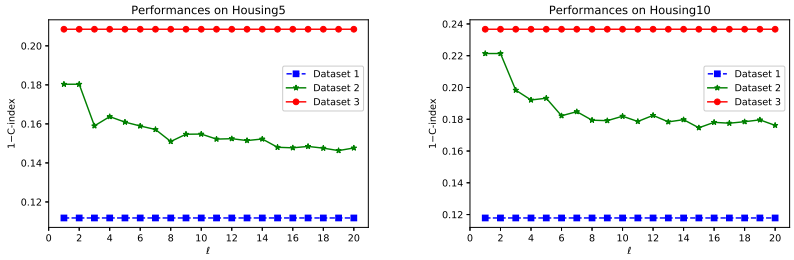


Figure 13: $1-$ C-index of three constructed datasets with a different number ℓ of nearest neighbor couples on the *Housing5* and *Housing10* datasets.

initially leads to an increasing performance. Once the number ℓ of nearest neighbor couples is sufficiently large, the performance becomes relatively stable, which means that a small number of nearest neighbor couples allows to get a good performance.

5.4. Discussion on the computational complexity

We analyze the computational complexity of our proposed method (in Algorithm 1). The search for nearest neighbor examples can be performed in $O(kn^2)$ using linear nearest neighbor search. Similarly, the search for nearest neighbor couples can be performed in $O(\ell m^2)$. Thus, the whole computational complexity is $O(kn^2\ell m^2)$. One should note that, although this complexity might seem too high, our problem setting requires that the size of absolute information is very small and $n \ll m$. Loosely speaking, this implies that the computational complexity of this method is $O(m^2)$, approximately. In order to speed up the search for nearest neighbor couples, k-d-trees and other tree structures [54] can be used.

6. Conclusions and future work

We have proposed a new method for ordinal classification with absolute and relative information. Specifically, we have combined absolute information and relative information and improved the classical machine learning method of

k-NN for ordinal classification. To test our method, we have performed experiments on some datasets from real ordinal classification problems and discretized regression problems. The experimental results show that the performance improves when more relative information is used, which demonstrates the effectiveness of incorporating relative information.

Our proposed method is a first attempt to extend an instance-based method to the framework in which we consider absolute and relative information. In future work, we will incorporate some ideas from the field of distance metric learning and try to learn a distance metric that is more appropriate than the (product) Euclidean distance metric. Moreover, we will explore how to incorporate both absolute and relative information in model-based machine learning methods.

References

[1] J. Chen, X. Liu, S. Lyu, Boosting with side information, in: Proceedings of the 2012 Asian Conference on Computer Vision, Springer, 2012, pp. 563–577.

[2] T. De Bie, M. Momma, N. Cristianini, Efficiently learning the metric with side-information, in: International Conference on Algorithmic Learning Theory, Springer, 2003, pp. 175–189.

[3] R. Jonschkowski, S. Höfer, O. Brock, Patterns for learning with side information, arXiv preprint arXiv:1511.06429.

[4] M. Sader, R. Pérez-Fernández, L. Kuuliala, F. Devlieghere, B. De Baets, The constrained median: a method to incorporate side information for improving the quality of assigned scores in shelf life tests, submitted.

[5] Q. Ouyang, J. Zhao, Q. Chen, Instrumental intelligent test of food sensory quality as mimic of human panel test combining multiple cross-perception sensors and data fusion, Analytica Chimica Acta 841 (2014) 68–76.

[6] C. H. Wan, L. H. Lee, R. Rajkumar, D. Isa, A hybrid text classification approach with low dependency on parameter by rating K-nearest neighbor and support vector machine, Expert Systems with Applications 39 (15) (2012) 11880–11888.

[7] Q.-B. Liu, S. Deng, C.-H. Lu, B. Wang, Y.-F. Zhou, Relative density based k-nearest neighbors clustering algorithm, in: Proceedings of IEEE International Conference on Machine Learning and Cybernetics, Vol. 1, 2003, pp. 133–17.

[8] O. Kramer, Dimensionality reduction by unsupervised k-nearest neighbor regression, in: Proceedings of the 10th IEEE International Conference on Machine Learning and Applications and Workshops (ICMLA), Vol. 1, 2011, pp. 275–278.

[9] Q. Yu, A. Sorjamaa, Y. Miche, A. Lendasse, E. Séverin, A. Guillén, F. Mateo, Optimal pruned K-nearest neighbors: OP-KNN application to financial modeling, in: Proceedings of the 8th IEEE International Conference on Hybrid Intelligent Systems (HIS08), 2008, pp. 764–769.

[10] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, R. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, in: Proceedings of the Annual Symposium on Computer Application in Medical Care, American Medical Informatics Association, 1988, p. 261.

[11] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.

[12] S. D. Bay, Nearest neighbor classification from multiple feature subsets, Intelligent Data Analysis 3 (3) (1999) 191–209.

[13] E. Fix, J. L. Hodges Jr, Discriminatory analysis-nonparametric discrimination: consistency properties, Tech. rep., California Univ Berkeley (1951).

[14] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.

[15] B. Nguyen, C. Morell, B. De Baets, Distance metric learning for ordinal classification based on triplet constraints, Knowledge-Based Systems 142 (2018) 17–28.

[16] B. Nguyen, C. Morell, B. De Baets, Large-scale distance metric learning for k-nearest neighbors regression, Neurocomputing 214 (2016) 805–814.

[17] S. Datta, D. Misra, S. Das, A feature weighted penalty based dissimilarity measure for k-nearest neighbor classification with missing features, Pattern Recognition Letters 80 (2016) 231–237.

[18] P. A. Gutiérrez, S. García, Current prospects on ordinal and monotonic classification, Progress in Artificial Intelligence 5 (3) (2016) 171–179.

[19] M. J. Mathieson, Ordinal models for neural networks, in: Proceedings of the 3rd Interenational Conference on Neural Networks in Capital Markets, 1996, pp. 523–536.

[20] A. S. Fullerton, J. Xu, The proportional odds with partial proportionality constraints model for ordinal response variables, Social Science Research 41 (1) (2012) 182–198.

[21] Q. Tian, S. Chen, X. Tan, Comparative study among three strategies of incorporating spatial structures to ordinal image regression, Neurocomputing 136 (2014) 152–161.

[22] P. A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, C. Hervás-Martínez, Ordinal regression methods: survey and experimental study, IEEE Transactions on Knowledge and Data Engineering 28 (1) (2016) 127–146.

[23] J. A. Aledo, J. A. Gámez, D. Molina, Tackling the supervised label ranking problem by bagging weak learners, Information Fusion 35 (2017) 38–50.

[24] J. Fürnkranz, E. Hüllermeier, Preference learning, Springer, 2010.

[25] S. Ovadia, Ratings and rankings: Reconsidering the structure of values and their measurement, International Journal of Social Research Methodology 7 (5) (2004) 403–414.

[26] H. van Herk, M. van de Velden, Insight into the relative merits of rating and ranking in a cross-national context using three-way correspondence analysis, Food Quality and Preference 18 (8) (2007) 1096–1105.

[27] M. Sader, J. Verwaeren, R. Pérez-Fernández, B. De Baets, Integrating expert and novice evaluations for augmenting ordinal regression models, Information Fusion 51 (2019) 1–9.

[28] M. Goldstein, k_n-nearest neighbor classification, IEEE Transactions on Information Theory 18 (5) (1972) 627–630.

[29] T. M. Mitchell, Machine learning and data mining, Communications of the ACM 42 (11) (1999) 30–36.

[30] Y. Song, J. Huang, D. Zhou, H. Zha, C. L. Giles, Iknn: Informative k-nearest neighbor pattern classification, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2007, pp. 248–264.

[31] S. A. Dudani, The distance-weighted k-nearest-neighbor rule, IEEE Transactions on Systems, Man, and Cybernetics (4) (1976) 325–327.

[32] K. Hechenbichler, K. Schliep, Weighted k-nearest-neighbor techniques and ordinal classification (2004).
URL `http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-1769-9`

[33] D. Dubois, H. Prade, On the use of aggregation operations in information fusion processes, Fuzzy Sets and Systems 142 (1) (2004) 143–161.

[34] C. Alsina, B. Schweizer, M. J. Frank, Associative functions: triangular norms and copulas, World Scientific, 2006.

[35] G. Beliakov, A. Pradera, T. Calvo, Aggregation functions: A guide for practitioners, Vol. 221, Springer, 2007.

[36] V. Torra, Y. Narukawa, Modeling decisions: information fusion and aggregation operators, Springer Science & Business Media, 2007.

[37] T. Calvo, A. Kolesárová, M. Komorníková, R. Mesiar, Aggregation operators: properties, classes and construction methods, in: Aggregation Operators, Springer, 2002, pp. 3–104.

[38] J. L. García-Lapresta, R. G. del Pozo, D. Pérez-Román, Metrizable ordinal proximity measures and their aggregation, Information Sciences 448 (2018) 149–163.

[39] J. L. García-Lapresta, D. Pérez-Román, Aggregating opinions in non-uniform ordered qualitative scales, Applied Soft Computing 67 (2018) 652–657.

[40] T. Calvo, G. Beliakov, Aggregation functions based on penalties, Fuzzy sets and Systems 161 (10) (2010) 1420–1436.

[41] H. Bustince, G. Beliakov, G. P. Dimuro, B. Bedregal, R. Mesiar, On the definition of penalty functions in data aggregation, Fuzzy Sets and Systems 323 (2017) 1–18.

[42] R. Pérez-Fernández, B. De Baets, On the role of monometrics in penalty-based data aggregation, IEEE Transactions on Fuzzy Systems, in press, DOI: 10.1109/TFUZZ.2018.2880716.

[43] M. M. Deza, E. Deza, Encyclopedia of distances, in: Encyclopedia of Distances, Springer, 2009, pp. 1–583.

[44] G. Beliakov, H. Bustince, S. James, T. Calvo, J. Fernández, Aggregation for Atanassov's intuitionistic and interval valued fuzzy sets: The median operator, IEEE Transactions on Fuzzy Systems 20 (3) (2012) 487–498.

[45] A. Asuncion, D. J. Newman, UCI machine learning repository (2007). URL http://www.ics.uci.edu/~mlearn/MLRepository.html

[46] PASCAL, (Pattern Analysis, Statistical Modelling and Computational Learning) machine learning benchmarks repository (2011). URL http://mldata.org/

[47] W. Chu, Z. Ghahramani, Gaussian processes for ordinal regression, Journal of Machine Learning Research 6 (7) (2005) 1019–1041.

[48] M. Cruz-Ramírez, C. Hervás-Martínez, J. Sánchez-Monedero, P. A. Gutiérrez, Metrics to guide a multi-objective evolutionary algorithm for ordinal classification, Neurocomputing 135 (2014) 21–31.

[49] S. Baccianella, A. Esuli, F. Sebastiani, Evaluation measures for ordinal regression, in: Proceedings of the 9-th IEEE International Conference on Intelligent Systems Design and Applications, 2009, pp. 283–287.

[50] W. Waegeman, Learning to rank: a ROC-based graph-theoretic approach, Ph.D. thesis, Springer (2009).

[51] C. Cortes, M. Mohri, AUC optimization vs. error rate minimization, in: Advances in Neural Information Processing Systems, 2004, pp. 313–320.

[52] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (1) (2006) 1–30.

[53] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[54] A. M. Kibriya, E. Frank, An empirical comparison of exact nearest neighbour algorithms, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2007, pp. 140–151.