Applied Intelligence (2024) 54:4443–4466
https://doi.org/10.1007/s10489-024-05371-8

Metric learning for monotonic classification: turning the space up to the limits of monotonicity

Juan Luis Suárez[1] · Germán González-Almagro[1] · Salvador García[1] · Francisco Herrera[1]

Accepted: 4 March 2024 / Published online: 26 March 2024

Abstract
This paper presents, for the first time, a distance metric learning algorithm for monotonic classification. Monotonic datasets arise in many real-world applications, where there exist order relations in the input and output variables, and the outputs corresponding to ordered pairs of inputs are also expected to be ordered. Monotonic classification can be addressed through several distance-based classifiers that are able to respect the monotonicity constraints of the data. The performance of distance-based classifiers can be improved with the use of distance metric learning algorithms, which are able to find the distances that best represent the similarities among each pair of data samples. However, learning a distance for monotonic data has an additional drawback: the learned distance may negatively impact the monotonic constraints of the data. In our work, we propose a new model for learning distances that does not corrupt these constraints. This methodology will also be useful in identifying and discarding non-monotonic pairs of samples that may be present in the data due to noise. The experimental analysis conducted, supported by a Bayesian statistical testing, demonstrates that the distances obtained by the proposed method can enhance the performance of several distance-based classifiers in monotonic problems.

Keywords Distance metric learning · Monotonic classification · Nearest neighbors · Triplet loss · M-Matrix

1 Introduction

Monotonic constraints [19] are common in real-world prediction problems where the variables to be predicted are ordinal and their order depends on the input data. For example, when predicting house prices, it is expected that—all other things being equal—a bigger house in the same area will have a higher price. Similarly, in predicting students' final grades, students with consistently higher grades during a course are also expected to have a higher final grade. These problems are known as monotonic classification problems [6], and are relevant in fields such as credit risk modeling [8] and lecturer evaluation [5]. Monotonic problems are prevalent in many heavily-regulated industries, and incorporating reasonable expectations about consistent application of selection constraints into automated decision-making systems [29] is crucial [9, 20].

When dealing with these problems, accuracy is not the sole factor to consider. It is equally crucial that the predictions closely follow the monotonic constraints present in the data. Furthermore, the cost of an incorrect prediction should increase as the prediction deviates further away from its actual value. Consequently, there is a need for classifiers that can handle these constraints and factor them in while making predictions.

Ordinal regression methods [17] are commonly used in classification problems where the labels possess an inherent ordering. These methods, which continue to be widely popular today [10, 39, 40], can be particularly useful for monotonic data as the labels in such data also have an inherent ordering. However, ordinal regression methods are not designed to handle monotonic constraints unless they are tailored to that purpose. Despite the significance of monotonicity in several real-world applications, only a few ordinal regression methods specifically address this property. Therefore, further research is necessary to develop more effective and efficient methods for monotonic classification. In recent years, there has been a growing effort to

✉ Juan Luis Suárez
jlsuarezdiaz@decsai.ugr.es

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

develop new methods for monotonic classification by adapting prominent algorithms from nominal classification, such as decision trees [27] or random forest [42], while also striving to enhance the explainability of the models [23].

Similarity-based learning methods have been successful in monotonic classification problems [6]. This type of learning is inspired by the human ability to recognize objects by their resemblance to other previously seen objects. This idea can be extended to fulfill monotonicity constraints by restricting similar objects or instances to those that comply with these constraints. The well-known nearest neighbors rule for classification [11] has been extended following this idea, so that the nearest neighbors are filtered in order to meet the monotonic constraints [13]. Recently, a new proposal restated the previous idea using a fuzzy approach [16] aiming to gain robustness against possible noise in the monotonicity constraints.

All of the above algorithms require a distance metric to function, and standard distances such as the Euclidean distance have become the go-to choice. However, using a distance metric that is better suited to the data can improve classifier performance. Distance metric learning [33] accomplishes this task and has been successful in several advanced learning problems, such as multi-output learning [21] or multi-dimensional classification [22], as well as ordinal problems with no monotonic constraints [25, 32]. However, its application when monotonicity constraints are present adds a significant hurdle. Distance metrics have the ability to transform the space [14] and, while this can reduce the number of instances that may break the monotonicity of the dataset, it is difficult to ensure that no new false monotonic constraints are introduced in the process—which may worsen the quality of the data. Although preprocessing techniques such as feature selection methods [28, 44] are effective in monotonic classification problems, the same cannot be said for preprocessing techniques that have the potential to modify the interdependence among features. Consequently, the application of distance metric learning algorithms becomes challenging, making it hard to enhance distance-based classifiers.

Our research presents a novel distance metric learning algorithm for monotonic classification. This algorithm aims to transform the input space in such a way that no new monotonic constraints are introduced, thus resolving the earlier issue. We accomplish this objective through monotonic matrices and M-matrices [4], which possess unique characteristics for defining distances that are highly advantageous for monotonic datasets. As we proceed further into this paper, we will delve deeper into these properties.

This paper represents an extension of our previous work on distance metric learning for monotonic classification [34]. While our earlier paper focused on the development of the basic algorithm and its initial evaluation, this paper presents a comprehensive analysis of the method that includes an expanded description of the method, a further analysis of the background and a theoretical justification of our approach. Our work also provides an extensive experimental evaluation of the method. Specifically, we have conducted a Bayesian statistical analysis of the results and performed a hyperparameter analysis to explore the impact of different parameter settings on the performance of the algorithm. We consider the most relevant metrics in monotonic classification to measure classification performance and test constraint fulfillment after applying our proposed transformations.

The paper is organized as follows. Section 2 describes the current state of distance metric learning and monotonic classification from a similarity-based learning perspective. Section 3 outlines our proposal of distance metric learning for monotonic classification. Section 4 describes the experiments conducted to evaluate the performance of our algorithm, and the results obtained, including the Bayesian statistical analysis and the hyperparameter discussion. Finally, Section 5 ends with the concluding remarks.

2 Background

In this section we will discuss the main problems we have tackled in this paper: distance metric learning, monotonic classification and how similarity-based methods are employed to address monotonic classification nowadays.

2.1 Distance metric learning

Distance metric learning [33] arose with the purpose of improving similarity-based (or, equivalently, distance-based) learning methods such as the k-nearest neighbors classifier, or k-NN. For this purpose, distance metric learning aims at learning distances that facilitate the detection of hidden properties in the data that a standard distance, such as the Euclidean distance, would fail to discover. Here, we will define *distance* as any map $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is 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 $x, y, z \in \mathcal{X}$.

We will also consider as distances the so-called *pseudo-distances*, which are those maps that verify (2) and (3), and where $d(x, x) = 0$ instead of (1).

Linear distance metric learning is the most common approach to learning distances between numerical data. It consists in learning Mahalanobis distances, which are parameterized by positive semidefinite matrices. Given a positive semidefinite matrix $M \in \mathcal{M}_d(\mathbb{R})_0^+$, and $x, y \in \mathbb{R}^d$, the Mahalanobis distance between x and y defined by M is given as

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

Since every positive semidefinite matrix M can be decomposed as $M = L^T L$ with $L \in \mathcal{M}_d(\mathbb{R})$ it follows that

$$\begin{aligned} d_M(x, y)^2 &= (x-y)^T M(x-y) = (x-y)^T L^T L(x-y) \\ &= (L(x-y))^T (L(x-y)) = \|L(x-y)\|_2^2. \end{aligned}$$

Therefore, learning a Mahalanobis distance is equivalent to learning a linear map L and then measuring the Euclidean distance after applying that linear map. Thus, the linear distance metric learning approach comes down to learning a positive semidefinite matrix (also called metric matrix) M or a linear map matrix L. Both approaches are equivalent. Learning M usually facilitates convexity during the optimization, while learning L facilitates other tasks such as dimensionality reduction [12].

2.2 Monotonic classification

Monotonic classification [6] arises in certain types of problems of ordinal nature with two particularities: firstly, there are order relations in both the input data (samples) and the output data (labels); secondly, for any given pair of instances, their relative order is also expected to be present in the relative order of their class labels. This happens, for example, when the data represent different measures or evaluations on a particular topic and the label represents a global expert assessment. It is to be expected that, if the measures of one instance are better than the measures of another instance, the global assessment obtained should also be better.

We now formally define what a monotonic dataset is. Let $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$ be a numerical dataset. Let $y_1, \dots, y_N \in \{1, \dots, C\}$ be the corresponding labels. The labels can be ordered using the ordinal relation $\leq$ among the natural numbers, since they take values between 1 and C. For each pair of samples in X, we can also compare their features element-wise. We may not be interested in making all the features comparable, since the monotonic constraints affecting the data may not be present in all the attributes. Thus, let $d_1, \dots, d_m \in \{1, \dots, d\}$ be the indices of all the features that have monotonicity constraints. These constraints can be direct or inverse. Without loss of generality, we can assume all the constraints are direct, and otherwise we can just flip the sign of the affected attribute.

Given two pairs of samples $x_i, x_j \in X$, we define an order relation between them as the product order, considering only the features with monotonicity constraints, i. e.,

$$x_i \leq x_j \iff x_{il} \leq x_{jl}, \text{ for every } l \in \{d_1, \dots, d_m\}.$$

Observe that this order is a partial order, that is, there may be samples x_i, x_j such that $x_i \not\leq x_j$ and $x_i \not\geq x_j$ simultaneously. The dataset $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$ will be *monotonic* if, for every $x_i, x_j \in X$, then

$$x_i \leq x_j \iff y_i \leq y_j.$$

In other words, the dataset D is monotonic if, and only if, for every comparable pair of samples, it is simultaneously true that: (i) all the attributes with monotonic constraints of the first instance are lower or equal than the attributes from the second instance; and (ii) the label of the first instance is lower or equal than the label of the second instance.

It is important to remark that, in real scenarios, due to the subjective nature of the labeling process or to measurement errors, some datasets may not be fully monotonic and there may be several pairs of instances for which monotonicity is broken. In any case, the goal of *monotonic classification* is to provide algorithms that, when predicting new labels, are able to respect the monotonicity constraints of the datasets, and that are also robust against monotonicity clashes that may arise when the dataset is not fully monotonic.

2.3 Monotonic classification and similarity-based learning methods

Similarity-based learning can be seen as closely related to ordinal classification problems. Typically, it is to be expected that if two samples are close their labels will also be close, and the farther apart the samples are the more different their labels will be as well. The k-NN classifier can be easily adapted to this setup. A common approach to handle ordinal labels with this classifier is to modify the aggregation vote function for the nearest neighbors, using, for example, the median of the labels instead of the mode. This can also be extended to handle situations where additional information beyond the labeled data is available [35]. In general, similarity-based algorithms are beneficial in other problems related to ordinal data, including ranking [30].

When our data also have monotonic constraints, additional caution is necessary, since we want the values predicted by the classifier to satisfy these constraints as far as possible. An immediate extension of the nearest neighbors classifier to monotonic classification problems is the *monotonic k-nearest neighbors* classifier (Mon-k-NN), which takes into account only the nearest neighbors whose labels lie on an interval that does not violate the monotonicity constraints [13]. Given a sample $x_0 \in \mathbb{R}^d$ we can consider the interval

$[y_{\min}, y_{\max}]$, where

$$y_{\min} = \max\{y \in \{1, \dots, C\} \colon (x, y) \in D \text{ and } x \leq x_0\}$$
$$y_{\max} = \min\{y \in \{1, \dots, C\} \colon (x, y) \in D \text{ and } x_0 \leq x\}$$

Two variants of Mon-k-NN can be considered. The *in-range* (IR) variant considers the k-nearest neighbors to x_0 with labels in the interval $[y_{\min}, y_{\max}]$, while the *out-range* (OR) variant considers the k-nearest neighbors in D and then only those neighbors with labels in $[y_{\min}, y_{\max}]$ are factored in for the vote (if no neighbors have labels in this range, then a random label in the interval will be chosen). Observe that this algorithm will not work properly if the dataset is not fully monotonic. In such a case, $y_{\min}$ may be greater than $y_{\max}$. Therefore, it is necessary to apply a relabeling process that makes the dataset fully monotonic while disturbing it as little as possible. A relabeling method that is applied on the complement of the maximum independent set of the monotonicity violation graph is proposed in [13].

A more recent proposal [16] relies on the fuzzy k-NN [18] in an effort to gain robustness against monotonicity constraints. The *monotonic fuzzy k-nearest neighbors* classifier (Mon-F-k-NN) first uses the in-range monotonic k-nearest neighbors to compute class membership probabilities for each sample in the training set. For each $x_i \in \mathcal{X}$ and $c \in \{1, \dots, C\}$, the probability $u(x_i, c)$ that the class of x_i will be c is defined as

$$u(x_i, c) = \begin{cases} RCr + (nn_c/k)(1 - RCr), & \text{if } y_i = c \\ (nn_c/k)(1 - RCr), \end{cases}$$

where nn_c is the number of nearest neighbors of the class c and RCr is a *real class relevance* estimation, between 0 and 1 (typically established as 0.5). From these memberships, each sample is reassigned to a class whose probability is a median value within the list of membership probabilities for the sample. This class reassignment enhances the monotonicity of the dataset. Finally, at the prediction stage, given the sample x, its k monotonic nearest neighbors $x_{i_1}, \dots, x_{i_k}$ are found and used to compute the membership probabilities of x as

$$u(x, l) = \frac{\sum_{j=1}^{k} u(x_{i_j}, l) \frac{pOR_j}{\|x - x_{i_j}\|^{m-1}}}{\sum_{j=1}^{k} \frac{pOR_j}{\|x - x_{i_j}\|^{m-1}}}.$$

Again, both in-range and out-range variants are available at the prediction stage. The out-range variant considers all the neighbors, even if their labels are not in $[y_{\min}, y_{\max}]$. If this is the case, then pOR_j is set to a previously fixed *out-range penalty* that decides how much weight these neighbors will have in the computation of the membership. In any other case, $pOR_j = 1$. The parameter m determines the influence of the distances of the neighbors. Lastly, the final class of x is taken again using the class associated with the median membership probability in $u(x, \cdot)$.

2.4 Monotonic classification and distance metric learning

Learning a Mahalanobis distance for a monotonic classification problem has several difficulties to overcome. If we try to learn the distance using a metric matrix, the distance is modified while the dataset is not, thus its monotonicity remains unchanged. This is not entirely positive, since the potential of distance metric learning gets squandered and, therefore, also the possibility of reducing the non-monotonicity of the dataset if it exists. However, if we learn the distance using a linear transformation, there is no guarantee that new false monotonic constraints are added. This may happen if we pick a distance defined by a generic $L \in \mathcal{M}_d(\mathbb{R})$. Consider, for example, the extreme case of a matrix L defining a 90-degree rotation in $\mathbb{R}^2$. If such a matrix transforms the dataset, all the monotonic constraints of the original dataset are lost and, furthermore, all those pairs of instances that were not comparable become false monotonic constraints with this rotation.

These drawbacks have so far prevented the development of distance metric learning algorithms for monotonic classification. To the best of our knowledge, there are currently no proposals in this area.

3 Algorithm description

In this section we will describe our distance metric learning proposal for monotonic classification. First, we will introduce the concepts needed to apply the algorithm. Then, we will describe the algorithm and, finally, we will show its optimization procedure. We named this approach *Large Margin Monotonic Metric Learning* (LM^3L).

3.1 Preliminary definitions

We will focus on the case where all the features in the dataset are subject to direct monotonicity constraints, so that the order relationship in the dataset coincides with the product order in $\mathbb{R}^d$. It's important to note that if there are inverse monotonicity constraints present, we can simply invert the sign of the corresponding features and apply the algorithm to the resultant dataset. Additionally, we will discuss the situation involving non-monotonic features at the end of the section.

As mentioned above, one of the problems of learning a distance by means of a linear transformation is that this transformation disturbs the monotonic constraints and, therefore, some new constraints that are not necessarily true could be added. However, this can be avoided by restricting ourselves to the appropriate subset of matrices, such as the one defined below.

Definition 1 A linear transformation or square matrix $L \in \mathcal{M}_d(\mathbb{R})$ is said to be *monotone* [4] if for any real vector $x \in \mathbb{R}^d$, we have that

$$Lx \geq 0 \implies x \geq 0,$$

where $0 \in \mathbb{R}^d$ is the vector with zeros in all its entries and $\geq$ is the product order in $\mathbb{R}^d$.

Observe that, if L is monotone, if we have two samples $x_i, x_j \in \mathcal{X}$ so that $Lx_i \geq Lx_j$, then $L(x_i - x_j) \geq 0$ and therefore $x_i \geq x_j$. This means that any pair of samples that meets a monotonicity constraint after applying L was already meeting the constraint before applying the transformation. So, when L is monotone, no new monotonic constraints can be added after the dataset is transformed. However, this property is not reciprocal: if $x_i \geq x_j$, it does not necessarily follow that $Lx_i \geq Lx_j$. Consequently, some monotonic constraints may be lost in this transformation. This will allow the algorithms that use this type of matrices to select the constraints that may be more relevant in the dataset without ever adding new incorrect monotonicity constraints after applying the transformation.

Monotone matrices are tough to use in optimization settings, since they cannot be adequately parameterized for this purpose. When L is invertible and monotonic, L is the inverse of a positive matrix (that is, a matrix with all its entries greater than or equal to zero). This may facilitate its parameterization, but the computation of the inverse matrix would make the optimization procedure very expensive. However, there is a subset of monotone matrices with much more suitable properties for use in differential optimization. We describe them below.

Definition 2 A linear transformation or square matrix $L \in \mathcal{M}_d(\mathbb{R})$ is an *M-Matrix* [4] if it can be expressed as $L = sI - B$, where I is the identity matrix of dimension d, $B \in \mathcal{M}_d(\mathbb{R})$ is a positive matrix, and $s \in \mathbb{R}$ verifies that $s \geq \rho(B)$, where $\rho(B)$ is the spectral radius of the matrix B.

M-matrices are monotone [4] and, since they depend on the real value s and the positive matrix B, they can be easily and efficiently used to optimize a differentiable objective function.

3.2 Objective function and optimization

After establishing the linear applications that enable us to regulate the monotonicity of the dataset, the next step is to define the function to be optimized. Since the linear application already controls the monotonicity implicitly, the focus of the objective function will be on assessing a goodness-of-classification metric. This metric should consider the ordinal nature of the dataset, in that the prediction penalty should increase as the actual label moves farther away from the predicted label.

Drawing inspiration from the large margin proposals for distance metric learning in other classification tasks [25, 43], we present a triplet-based objective function. For each anchor sample x_i in the dataset, we consider a positive sample x_j and a negative sample x_l such that $y_i \leq y_j < y_l$ or $y_i \geq y_j > y_l$. The aim is to minimize the distance from x_i to x_j while simultaneously maximizing the distance from x_i to x_l. The objective function and the associated constrained optimization problem are defined as follows:

$$\begin{aligned}\min_{L \in \mathcal{M}_d(\mathbb{R})} f(L) = & \sum_{x_i \in \mathcal{X}} \sum_{\substack{x_j, x_l \in \mathcal{U}(x_i) \\ y_i \leq y_j < y_l \\ \text{or} \\ y_i \geq y_j > y_l}} \left[\|L(x_i - x_j)\|^2 - \|L(x_i - x_l)\|^2 + \lambda\right]_+ \\ \text{s. t. :} & L = sI - B \\ & B_{ij} \geq 0, (i, j = 1, \dots, d) \\ & s \geq \rho(B).\end{aligned} \tag{1}$$

In the aforementioned optimization problem, the notation $[z]_+ = \max\{z, 0\}$ is used, where λ denotes a margin constant. The aim is to ensure that the distance from the negative sample to the anchor sample is not smaller than the distance from the positive sample to the anchor sample plus the margin constant. Moreover, for each $x_i \in \mathcal{X}$, $\mathcal{U}(x_i)$ represents a neighborhood that includes the K nearest neighbors to x_i for the Euclidean distance. This neighborhood is computed prior to the optimization process and serves to filter the instances that are initially farther away, giving a local character to the method and reducing the computational cost. This draws inspiration from the metric learning method for ordinal regression proposed in [25]. The parameter K represents a hyperparameter that can be adjusted to enhance algorithm performance. It is suggested to set K to a sufficiently large value to ensure representative neighborhoods. This is further discussed in Section 4.5. The choice of the Euclidean distance stems from its suitability as an a priori distance measure before the algorithm learns from the data [24, 36, 41, 43]. However, alternative precomputed distance measures can also be considered.

The constraints specified in the optimization problem of (1) guarantee that no additional monotonic constraints are introduced when the dataset is transformed. On the other hand, the objective function aims to bring data from nearby classes closer while pushing data from distant classes farther apart. By minimizing (1), the transformed dataset that we obtain has optimal ordinality and monotonicity properties, which can then be learned by a similarity-based classifier.

To optimize (1), we propose a stochastic projected gradient descent method. Since L is fully parameterized by s and B, the optimization problem can be rewritten as

$$\min_{\substack{s\in\mathbb{R}, B\in\mathcal{M}_d(\mathbb{R})\\ s\geq\rho(B)\\ B_{ij}\geq 0\ \forall i,j}} f(L) = \sum_{x_i\in\mathcal{X}} \sum_{\substack{x_j,x_l\in\mathcal{U}(x_i)\\ y_i\leq y_j<y_l\\ \text{or}\\ y_i\geq y_j>y_l}} \Big[\|(sI-B)(x_i-x_j)\|^2 - \|(sI-B)(x_i-x_l)\|^2+\lambda\Big]_+ . \tag{2}$$

At each gradient step we can update the pair (s, B) using the partial derivatives. We know that [26]

$$\frac{\partial f}{\partial s}(s,B) = \sum_{x_i\in\mathcal{X}} \sum_{x_j,x_l\in\mathcal{A}_L(x_i)} d_{ij}^T[2sI-(B+B^T)]d_{ij} - d_{il}^T[2sI-(B+B^T)]d_{il}, \tag{3}$$

$$\frac{\partial f}{\partial B}(s,B) = \sum_{x_i\in\mathcal{X}} \sum_{x_j,x_l\in\mathcal{A}_L(x_i)} 2(B-sI)(O_{ij}-O_{il}), \tag{4}$$

where $d_{ij} = x_i - x_j$, $O_{ij} = d_{ij}d_{ij}^T$, and $\mathcal{A}_L(x_i)$ is the set of active (positive, negative) 2-tuples associated with the anchor sample x_i and $L = sI - B$, that is:

$$\mathcal{A}_L(x_i) = \{(x_j,x_l)\colon x_j,x_l\in\mathcal{U}(x_i), [(y_i\leq y_j<y_l) \text{ or } (y_i\geq y_j>y_l)] \text{ and } \|L(x_i-x_j)\|^2 - \|L(x_i-x_l)\|^2+\lambda>0\}.$$

From this, in the stochastic gradient descent process, we choose at each step a random sample $x_i \in \mathcal{X}$ and update s and B with the following rules:

$$s_{new} = s_{old} - \eta \sum_{x_j,x_l\in\mathcal{A}(x_i)} d_{ij}^T[2sI-(B+B^T)]d_{ij} - d_{il}^T[2sI-(B+B^T)]d_{il}, \tag{5}$$

$$B_{new} = B_{old} - \eta \sum_{x_j,x_l\in\mathcal{A}(x_i)} 2(B-sI)(O_{ij}-O_{il}), \tag{6}$$

where η is a pre-established learning rate. Since the above update rules do not ensure that s and B meet the constraints to which they are subject, it is necessary to project them into the constrained set. Therefore, after applying the update rules, we convert the negative entries of B to zero and, if s is smaller than $\rho(B)$, we make it equal to $\rho(B)$:

$$\pi(B) = (\tilde{B}_{ij}), \text{ where } \tilde{B}_{ij} = \max\{B_{ij}, 0\}, \text{ for each } i,j = 1,\ldots,d. \tag{7}$$

$$\pi(s) = \max\{s, \rho(B)\}. \tag{8}$$

This concludes the optimization process of LM^3L. In short, at each epoch the samples $x_i \in \mathcal{X}$ are chosen randomly. With each of the samples, s and B are updated using the rules from (5) and (6) and then projected into valid values with (7) and (8). The process is repeated until a maximum of epochs is reached or the algorithm converges. With the final values of s and B, the obtained distance is retrieved by means of the linear transformation $L = sI - B$.

3.3 Benefits of the method

Our distance metric learning algorithm for monotonic classification offers several advantages from a theoretical perspective. Firstly, it can find new transformed variables in the latent space that may better capture the monotonicity of the dataset. By learning a distance metric that is specifically tailored to the problem of monotonic classification, without introducing any new fake monotonic constraints, the algorithm can identify new features that are better suited for capturing the underlying monotonic structure of the data. This can lead to better classification performance and a deeper understanding of the relationships between the variables.

Secondly, since no new monotonic constraints can be added but some of them may be removed, our algorithm can help us to filter the dataset and to discover different ways of interaction in the latent space. By removing constraints that are not relevant, the algorithm can provide insight into the structure of the data and help us to discover new relationships among the variables. This can be particularly useful in cases where the data is high-dimensional or complex, and where traditional methods may struggle to identify meaningful patterns [38].

Finally, the new variables in the latent space can contribute further information on how the variables are related and their impact on the monotonicity of the dataset, which can assist in making interpretable and explainable decisions about the data. By providing a more complete picture of the underlying structure of the data, the algorithm can help us to identify important features and relationships that may not be immediately apparent from the raw data. This can be particularly useful in cases where the data is being used to make critical decisions, such as in finance or healthcare, where interpretability and transparency are essential.

In summary, our distance metric learning algorithm offers several major benefits from a theoretical perspective, including the ability to identify new transformed variables in the latent space, the ability to filter and discover new ways of interaction in the data, and the ability to facilitate interpretable and explainable decisions about the data. These benefits make it a powerful tool for researchers and practitioners working in a variety of fields and applications where monotonicity is a key consideration.

3.4 LM^3L and non-monotonic features

In the previous sections we have assumed that all the features in the dataset are subject to monotonic constraints. However, this assumption may not hold in real-world scenarios. In the context of distance-based classification, both the majority and median-vote k-NN classifiers do not consider the monotonic constraints in any sense. On the other hand, the monotonic and monotonic-fuzzy variants assume the monotonicity across all the features [13, 16]. Since our algorithm is designed to learn a distance that respects the dataset's monotonic constraints, it is essential to address how to handle non-monotonic features and how the later classification stage will be affected by them.

LM^3L can be adapted or combined with other algorithms in order to handle non-monotonic features. One approach is to apply LM^3L locally to the monotonic attributes and then employ another distance metric learning algorithm for standard classification [14, 43] locally to the non-monotonic features. Concatenating the obtained maps, represented as a matrix containing the two locally learned distance matrices as blocks, will yield a global distance metric for use in the classification stage. This method treats the two types of features separately, thus not capturing interactions between monotonic and non-monotonic features.

An alternative approach that does capture the interactions between monotonic and non-monotonic features is to introduce an unconstrained matrix L_0 to the optimization problem of (1) for the non-monotonic attributes. This introduces the constraint $L = sI - B + L_0$, where L_0 only contains non-zero rows in the positions corresponding to the non-monotonic features. Consequently, the monotonic constraints remain effective for monotonic features, while L_0 removes limitations on exploring the search space for non-monotonic features.

In applying subsequent distance-based classification methods, since monotonic nearest neighbors approaches assume all features are monotonic, it's advisable to rely on standard majority-vote or median-vote classifiers. These classifiers do not assume monotonicity in the data, as those assumptions are already considered during the distance learning stage.

4 Experiments

In this section we describe the experiments we have developed with our algorithm and the results we have obtained.

4.1 Experimental framework

We have assessed the distance metric learned by LM^3L through various distance-based classifiers. All of them are variations of the nearest neighbors classifier, which include: the original k-NN (majority-vote), the median-vote k-NN, the monotonic k-NN (both the in-range and out-range versions), and the monotonic fuzzy k-NN (both the in-range and out-range versions). The standard k-NN is commonly applied in non-ordinal classification problems, while the median-vote k-NN is the natural adaptation for k-NN in ordinal regression, without taking into consideration any monotonic constraints. The remaining k-NN versions refer to the monotonic nearest neighbors approaches discussed in Section 2.3.

The goal of these experiments is to evaluate whether the distance metric learned by LM^3L can improve the performance of k-NN in two ways: (1) classification accuracy when dealing with new data and (2) adherence to the monotonic constraints of the dataset. To achieve this goal, we will compare various versions of k-NN using both the Euclidean distance and the distance learned by LM^3L.

The experiments were conducted using a fixed number of neighbors $k = 9$ for all the k-NN classifiers. The distances computed using each classifier were evaluated through a stratified 5-fold cross validation, which preserves the original class proportions in each fold. Ten different numerical datasets with monotonic constraints from various sources were used in the experiments [1, 17, 37]. To prepare the data for the experiments, any features with inverse monotonic constraints were sign-switched, and a *min-max* normalization to the interval [0, 1] was applied. It is worth noting that some datasets were not completely monotonic, and contained pairs of samples that violated the monotonicity constraints. The datasets selected for the experiments, along with their dimensions and monotonicity properties, are presented in Table 1.

4.2 Metrics and results

To evaluate the classification performance of the distances with each classifier, we have used two metrics: the *mean absolute error* (MAE) [17], which penalizes the classification error according to the distances between the labels, and the *concordance index* (C-INDEX) [15], which measures the ratio between the number of ordered pairs in both true labels and predictions and the number of all comparable pairs.

For the execution of these experiments, the parameters suggested for LM^3L are as follows: a fixed neighborhood size of 50 for the anchor samples, a maximum of 300 optimization epochs, a neighborhood margin λ of 0.1, and an adaptive learning rate η. The adaptive learning rate starts at 10^{-6} and, at each epoch, it is either increased by 1 % if the objective function improves, or halved if it does not, following the adaptive approach in [43]. These parameters were chosen based on the guidelines of the algorithms that inspired this method, as well as a preliminary hyperparameter analysis presented in Section 4.5. The code of LM^3L used for these experiments is available in `pyDML` [31], which is

Table 1 Datasets used in the experiments

Dataset	# Samples	# Features	# Classes	Attribute directions	Non-Monotonic pairs / Comparable pairs (%)
autoMPG8	392	7	5	(-,-,-,-,+,+,+)	0.044 / 36.14
car	1728	6	4	All direct (+)	0.246 / 39.67
ERA	1000	4	9	All direct (+)	3.349 / 16.77
ESL	482	4	7	All direct (+)	0.585 / 59.81
LEV	1000	4	5	All direct (+)	1.330 / 24.08
machineCPU	209	6	5	(-,+,+,+,+,+)	1.196 / 46.24
pima	768	8	2	All direct (+)	0.151 / 6.576
SWD	1000	10	4	All direct (+)	0.949 / 12.62
balance	625	4	3	(-,-,+,+)	0.000 / 25.64
boston-housing	506	13	5	(-,+,-,+,-,+,-,+,-,-,-,+,-)	0.299 / 14.60

a Python library containing various distance metric learning algorithms.

Table 2 shows the results of the classification performance. This table also includes, for each combination of distance and classifier, its average ranking over all the combinations of distance and classifiers (AVG RANK [ALL]) and its average ranking within the distances that use the same classifier (AVG RANK [IN]). The combination of distance and classifier with the highest value or C-INDEX and MAE for each dataset is highlighted in bold.

To evaluate the fulfillment of the monotonic constraints we rely on the *non monotonicity index* (NMI). This metric is a normalized measure of how many samples do not fulfill a monotonic constraint. This can be used to evaluate both the monotonicity of the transformed training dataset after applying LM^3L and the monotonicity of the predicted samples with respect the training dataset. For a training set $\mathcal{X}$ and a labeled point $(x, y) \in \mathbb{R}^d \times \{1, \dots, C\}$ we define

$$NClash_{\mathcal{X}}(x) = |\{x_i \in \mathcal{X} : (x_i < x \text{ and } y_i > y) \text{ or } (x_i > x \text{ and } y_i < y)\}|.$$

Then, the NMI of the labeled dataset $\mathcal{X}$ with respect to the labeled dataset $\mathcal{Y}$ is defined as

$$NMI(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}||\mathcal{Y}| - |\mathcal{X} \cap \mathcal{Y}|} \sum_{x \in \mathcal{Y}} NClash_{\mathcal{X}}(x).$$

We can use the NMI in different ways. If we want to measure the monotonicity of the original training set $\mathcal{X}$, we can use $NMI(\mathcal{X}, \mathcal{X})$. If we want to measure the monotonicity of the training set after being transformed by a linear map $L \in \mathcal{M}_d(\mathbb{R})$, we can use $NMI(L\mathcal{X}, L\mathcal{X})$. Finally, if we want to measure the monotonicity of a set of test samples and their predictions, $\mathcal{X}_t$, with respect to the training set, we can use $NMI(L\mathcal{X}, \mathcal{X}_t)$. Table 3 shows the results regarding the fulfillment of the monotonic constraints. In this table, the metric NMI-TRAIN represents the NMI for the training sets, for the Euclidean distance (that is, with no transformations applied) and for the transformed dataset using the distance learned by LM^3L. The NMI-TEST metric represents each of the NMIs of the training sets for each distance, with respect to the sets of predicted values by each of the classifiers for the test set. We also show the total of comparable pairs in the training set (CP-TRAIN), and between the training and test sets (CP-TEST), for each of the distances. The lowest values of NMI and CP for each dataset are highlighted in bold.

Finally, we also include in Table 5 the time required for LM^3L to learn the distance within each dataset. It is important to note that these times are independent of the classifier employed, as the distance is learned independently of the classification stage. In addition, there is no comparison with the Euclidean distance, opposed as it was done in Table 2. One might assume that the Euclidean distance requires zero time, but in reality, no distance learning process occurs in that scenario. The provided timings illustrate that the algorithm scales effectively in response to the growing number of samples in the datasets. This can be primarily attributed to the local nature provided by the neighborhood filter during triplet generation.

4.3 Analysis of results

Based on the results presented in Tables 2 and 3, we can make the following observations. Firstly, we can conclude that the performance of the Med-k-NN classifier improves significantly when combined with LM^3L in terms of both MAE and C-INDEX. In fact, the combination of LM^3L and Med-k-NN is the most successful one in the experiments. In contrast, the combination of LM^3L with monotonic classifiers yielded less competitive results compared to non-monotonic classifiers, often performing worse than the Euclidean distance in

Table 2 MAE and C-INDEX of the distance and classifiers on each dataset

Dataset	k-NN		Med-k-NN		Mon-k-NN (IR)		Mon-k-NN (OR)		Mon-F-k-NN (IR)		Mon-F-k-NN (OR)	
	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
C-INDEX												
autoMPG8	0.918457	0.920312	0.920898	**0.925040**	0.920753	**0.925040**	0.919799	**0.925040**	0.919230	0.911326	0.918868	0.912257
car	0.963044	0.972001	0.962430	**0.974398**	0.971015	**0.974398**	0.970817	**0.974398**	0.964425	0.736149	0.942335	0.718687
ERA	0.692304	0.683062	0.696339	0.692771	0.675817	0.662343	0.675817	0.662343	**0.702484**	0.698284	0.700877	0.701683
ESL	0.911469	0.914401	0.917153	0.918642	0.912686	0.901638	0.911558	0.902055	0.917847	0.915939	**0.918671**	0.916644
LEV	0.803299	0.816627	0.804052	0.819050	0.768287	0.736397	0.768287	0.736397	0.818089	0.815131	**0.822355**	0.821664
machineCPU	0.857039	0.854280	0.860909	0.867868	**0.880012**	0.877510	0.873755	0.874879	0.863247	0.869615	0.855029	0.870145
pima	0.705144	0.712881	0.705144	0.712881	0.706292	0.706473	0.709292	0.706473	**0.719280**	0.667439	0.685984	0.665552
SWD	0.742335	0.743421	0.749917	0.752555	0.685411	0.674434	0.689080	0.674434	0.759628	0.757493	**0.761008**	0.754048
balance	0.903139	0.898828	0.933961	0.940843	0.946904	0.948984	0.946882	0.948657	0.948366	**0.951353**	0.944890	0.944437
boston-housing	0.780491	0.848367	0.795450	**0.853051**	0.803741	**0.853051**	0.799271	**0.853051**	0.819694	0.810555	0.773578	0.810751
AVG SCORE	0.827672	0.836418	0.834625	**0.845710**	0.827092	0.826027	0.826456	0.825773	0.843229	0.813328	0.832360	0.811587
AVG RANK [ALL]	8.863636	6.409091	6.954545	**3.772727**	6.818182	6.454545	7.363636	6.636364	4.090909	7.090909	6.363636	7.181818
AVG RANK [IN]	1.666667	**1.333333**	1.833333	**1.166667**	**1.416667**	1.583333	**1.416667**	1.583333	**1.250000**	1.750000	**1.333333**	1.666667
MAE												
autoMPG8	0.342740	0.322085	0.332608	**0.306889**	0.334720	**0.306889**	0.337281	**0.306889**	0.329528	0.359831	0.331758	0.360091
car	0.052658	0.050339	0.054392	0.049193	**0.030085**	0.049193	0.031826	0.049193	0.056113	0.256990	0.090264	0.270272
ERA	1.411089	1.465980	1.300229	1.310030	1.527210	1.556053	1.527210	1.556053	**1.291049**	1.300199	1.293127	1.292118
ESL	0.331710	0.327648	0.321226	**0.317207**	0.352483	0.396309	0.356799	0.396309	0.342489	0.346530	0.335982	0.338046
LEV	0.438115	0.415123	0.421073	0.405062	0.496170	0.544144	0.496170	0.544144	0.398021	0.407012	0.384050	0.389021
machineCPU	0.613086	0.612891	0.574255	0.560191	**0.489040**	0.513241	0.508696	0.527438	0.549397	0.527777	0.572187	0.543366
pima	0.247441	0.247432	0.247441	0.247432	0.244826	0.251328	**0.240905**	0.251328	0.244835	0.259138	0.247449	0.260453
SWD	0.479069	0.482130	0.448037	0.450157	0.503911	0.536098	0.499966	0.536098	0.442002	0.447038	**0.440997**	0.450057
balance	0.148918	0.158031	0.147267	0.134311	**0.083351**	0.087984	**0.083351**	0.091236	0.092991	0.092927	0.129795	0.129666
boston-housing	0.588659	0.422635	0.549184	**0.406773**	0.499192	**0.406773**	0.517014	**0.406773**	0.672612	0.473747	0.570193	0.477609
AVG SCORE	0.465349	0.450429	0.439571	**0.418725**	0.456099	0.464801	0.459922	0.466546	0.441904	0.447119	0.439580	0.451070
AVG RANK [ALL]	8.590909	6.590909	6.409091	**4.318182**	6.000000	7.227273	6.454545	7.590909	5.363636	6.636364	5.818182	7.000000
AVG RANK [IN]	1.750000	**1.250000**	1.833333	**1.166667**	**1.166667**	1.833333	**1.166667**	1.833333	**1.250000**	1.750000	**1.333333**	1.666667

Table 3 Results of the monotonicity analysis on each dataset

Metric	NMI-TRAIN		CP-TRAIN		NMI-TEST					
Classifier	(C.I.)		(C.I.)		k-NN		Med-k-NN		Mon-k-NN (IR)	
Distance	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
autoMPG8	0.002558	**0.000000**	**0.401125**	0.000539	0.001534	**0.000000**	0.001390	**0.000000**	0.000834	**0.000000**
car	0.000058	**0.000000**	**0.143672**	0.003914	0.000126	**0.000000**	0.000116	**0.000000**	0.000028	**0.000000**
ERA	0.033560	**0.021917**	**0.167776**	0.075397	0.026396	0.018346	0.025813	0.017834	0.027845	0.020012
ESL	0.009530	**0.003307**	**0.703749**	0.348972	0.006029	0.002217	0.005926	0.002212	0.007266	0.002425
LEV	0.013335	**0.005879**	**0.240869**	0.048001	0.008462	0.003953	0.008401	0.003911	0.010639	0.005150
machineCPU	0.013873	**0.003957**	**0.497229**	0.289424	0.011322	0.002963	0.011395	0.002906	0.006639	0.001817
pima	0.001640	**0.000045**	**0.073187**	0.001316	0.000896	0.000015	0.000896	0.000015	0.000396	0.000002
SWD	0.009505	**0.004307**	**0.126258**	0.008387	0.005609	0.003078	0.005274	0.003100	0.007336	0.003595
balance	**0.000000**	**0.000000**	**0.256326**	0.209876	0.000051	0.000041	0.000038	0.000016	**0.000000**	**0.000000**
boston-housing	0.002775	**0.000000**	**0.149025**	0.000000	0.001934	**0.000000**	0.001731	**0.000000**	0.001443	**0.000000**
AVG SCORE	0.008683	**0.003941**	**0.275922**	0.098583	0.006236	0.003061	0.006098	0.002999	0.006243	0.003300
AVG RANK	1.954545	**1.045455**	**1.000000**	2.000000	10.681818	4.636364	9.772727	4.090909	9.500000	4.318182
Metric	NMI-TEST						CP-TEST			
Classifier	Mon-k-NN (OR)		Mon-F-k-NN (IR)		Mon-F-k-NN (OR)		(C.I.)			
Distance	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L		
autoMPG8	0.000762	**0.000000**	0.000747	**0.000000**	0.001657	**0.000000**	**0.400190**	0.000507		
car	0.000028	**0.000000**	0.000085	0.000005	0.000222	4.18e-07	**0.143342**	0.003887		
ERA	0.027845	0.020012	0.024477	0.017358	0.024352	**0.017335**	**0.167337**	0.075719		
ESL	0.007138	0.002404	0.006103	0.001991	0.006096	**0.001914**	**0.703130**	0.351790		
LEV	0.010639	0.005150	0.007547	0.003831	0.007555	**0.003815**	**0.240689**	0.047908		
machineCPU	0.006346	0.001614	0.008060	**0.001277**	0.011200	0.002779	**0.489219**	0.286028		
pima	0.000411	0.000002	0.000231	**0.000000**	0.000744	0.000012	**0.073161**	0.001432		
SWD	0.006880	0.003595	0.005230	0.002951	0.005212	**0.002945**	**0.125906**	0.008363		
balance	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0.000003	**0.000000**	**0.256840**	0.211008		
boston-housing	0.001438	**0.000000**	0.000441	**0.000000**	0.001719	**0.000000**	**0.147432**	0.000000		
AVG SCORE	0.006149	0.003278	0.005292	**0.002741**	0.005876	0.002880	**0.274725**	0.098664		
AVG RANK	8.954545	4.045455	7.545455	**2.545455**	9.181818	2.727273	**1.000000**	2.000000		

The (C.I.) column title states that the metrics do not depend on the classifier used, only on the distance

Table 4 Non-monotonicity index over the comparable pairs in both train and test datasets

Metric	NMI on CP [TRAIN]		NMI on CP [TEST]											
Classifier	(C.I.)		k-NN		Med-k-NN		Mon-k-NN (IR)		Mon-k-NN (OR)		Mon-F-k-NN (IR)		Mon-F-k-NN (OR)	
Distance	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
autoMPG8	0.006381	**0.000000**	0.003834	**0.000000**	0.003473	**0.000000**	0.002084	**0.000000**	0.001904	**0.000000**	0.001870	**0.000000**	0.004162	**0.000000**
car	0.000402	**0.000000**	0.000881	**0.000000**	0.000812	**0.000000**	0.000190	**0.000000**	0.000190	**0.000000**	0.000589	0.001088	0.001530	0.000135
ERA	**0.200070**	0.291700	0.158011	0.243392	0.154546	0.236284	0.166521	0.264883	0.166521	0.264883	0.146549	0.229999	**0.145805**	0.229653
ESL	0.013542	**0.009365**	0.008573	0.006386	0.008426	0.006376	0.010339	0.006775	0.010157	0.006734	0.008683	0.005616	0.008672	**0.005460**
LEV	**0.055372**	0.122519	0.035194	0.082691	0.034919	0.081782	0.044228	0.107630	0.044228	0.107630	**0.031383**	0.080121	0.031412	0.079804
machineCPU	0.027944	**0.013879**	0.023643	0.011360	0.023905	0.011158	0.013765	0.007022	0.013158	0.006294	0.016674	**0.004882**	0.023268	0.010723
pima	**0.022528**	0.044904	0.012296	0.036612	0.012296	0.036612	0.005440	0.000465	0.005628	0.000465	0.003171	**0.000000**	0.010213	0.036147
SWD	0.075348	**0.051356**	0.044585	0.036811	0.041939	0.037076	0.058386	0.042992	0.054865	0.042992	0.041572	0.035292	0.041459	**0.035218**
balance	**0.000000**	**0.000000**	0.000200	0.000257	0.000149	0.000083	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0.000012	**0.000000**
boston-housing	**0.018065**	-	**0.013812**	-	**0.012271**	-	**0.010734**	-	**0.010695**	-	**0.003323**	-	**0.012661**	-
AVG SCORE	**0.038232**	0.053438	0.027446	0.041861	0.026692	0.041047	0.028361	0.042976	0.027966	0.042900	**0.023074**	0.035700	0.025394	0.039774
AVG RANK	1.650000	**1.350000**	8.083333	7.636363	7.166666	7.000000	6.916666	6.136363	6.500000	5.863636	4.625000	**4.545454**	6.500000	5.318181

Table 5 Time (in seconds) required for LM^3L to learn the distance within each dataset

	TIME
autoMPG8	3053.281696
car	9943.930562
ERA	8204.803218
ESL	2991.658279
LEV	7656.171229
machineCPU	1502.185281
pima	5180.773168
SWD	7346.903710
balance	1818.846962
boston-housing	4512.102694

those particular cases. Thus, we can say that the distance learned by LM^3L is capable of achieving superior classification performance compared to the Euclidean distance, but only when used in combination with the more traditional k-NN and Med-k-NN classifiers. This could be attributed to the fact that the monotonic classifiers already heavily focus on optimizing the constraint aspect, which may render their combination with LM^3L counterproductive. In any case, combining LM^3L with a non-monotonic classifier is not a drawback, since monotonic constraints are already taken into account in the distance learning process itself.

Finally, by looking at the monotonicity results, it becomes evident that the transformation learned by our algorithm significantly reduces the number of non-monotonic pairs of samples after transforming the training set. The observed monotonicity of the predicted samples with respect to the training set confirms that LM^3L is successful in decreasing the number of predictions that violate a monotonic constraint, for all classifiers. This highlights the capability of our method to avoid introducing new incorrect monotonic constraints when transforming the dataset, owing to the utilization of M-matrices in the optimization process. However, it is worth noting that the reduction in NMI comes at the expense of diminishing the number of comparable instances in the dataset, as is apparent in Table 3; the number of comparable pairs is consistently higher in the untransformed dataset. Nevertheless, this reduction can assist in identifying instances that are inaccurately linked in monotonic constraints due to noise or lack of accuracy. The results presented in Table 4 demonstrate the performance of the NMI metric when considering only the comparable pairs in both the training and test sets. The lowest relative NMI values are again highlighted in bold. The average NMI values for Euclidean and LM^3L distances are similar, but LM^3L outperforms Euclidean distance in terms of ranking. These findings suggest that, despite the reduction in the number of comparable pairs, the NMI metric normalized by the number of comparable pairs remains competitive when LM^3L is employed.

4.4 Bayesian non-parametric statistical analysis

In order to assess the extent to which the best models obtained outperform the other models, and to compare the distances learned on the same classifier, we have performed a series of Bayesian statistical tests. We have prepared several pairwise Bayesian sign tests [2] to perform these comparisons. The tests take into account the differences between the C-Index and MAE scores obtained by each pair of compared algorithms, assuming that their prior distribution is a Dirichlet process [3], defined by a prior strength $s = 1$ and a prior pseudo-observation $z_0 = 0$. After perceiving the score obtained for each dataset, the tests produce a posterior distribution that gives us the probabilities that either one of the compared algorithms outperforms the other, or that they are practically equivalent. The region of practical equivalence has been established as the region where the score differences are in the interval $[-0.01, 0.01]$. In summary, from the posterior distribution we obtain three probabilities: the probability that the first algorithm outperforms the second, the probability that the second algorithm outperforms the first, and the probability that the two algorithms are practically equivalent. The distribution can be plotted as a ternary simplex plot for a sample of the posterior distribution, where a greater skew of the points towards on of the regions represent a higher probability.

To carry out the Bayesian sign tests we have used the R package `rNPBST` [7]. In Figs. 1 and 2 we show all the pairwise comparisons among every combination of distance and classifier. This comparison is displayed as a heatmap, with the lower half showing the posterior probability for the algorithm with the highest likelihood of outperformance against its competitor. The color of the heatmap in this half indicates which algorithm is the winner via an increase in color intensity with higher probability of outperformance. The upper half shows the posterior probabilities that the compared pairs of algorithms are practically equivalent (the rope region probability). Again, the intensity of the color refers to a higher probability, while the two colors indicate how high the rope probability is: whether the algorithms are more likely to perform equivalently or the better algorithm in the lower half clearly wins.

In the comparisons of Figs. 1 and 2, we can confirm that Med-k-NN with the distance learned by LM^3L is the algorithm that stands out the most, since, when compared to the other algorithms, its probability of winning always exceeds the probability of the other algorithm winning. Looking at the C-Index, we observe that the rope probabilities are high in general, which indicates that it is also likely that Med-k-NN with LM^3L has an equivalent performance, in terms of

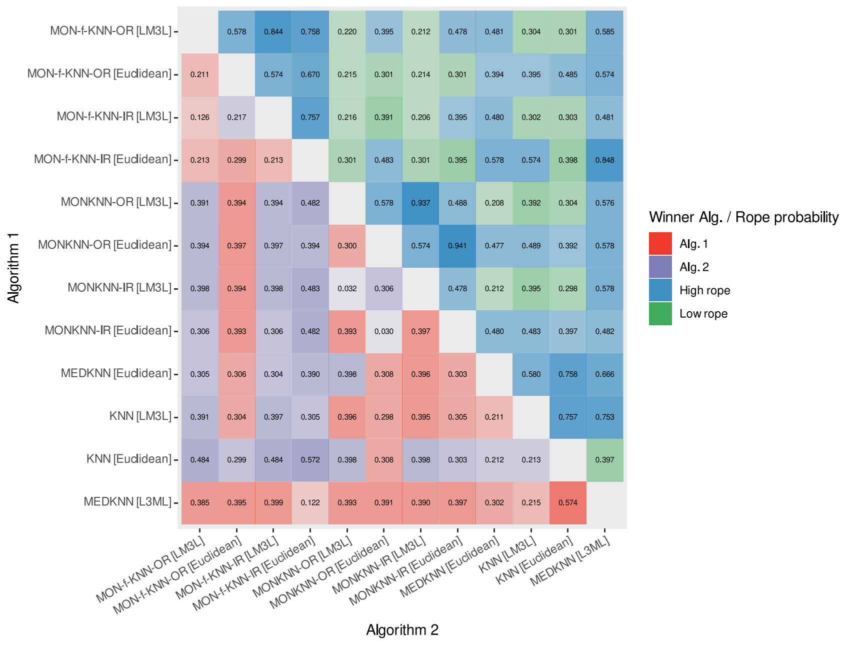

Fig. 1 Pairwise Bayesian comparisons of the C-Index scores obtained by the different algorithms

the C-Index, to the other algorithms. In any case, the probability that this algorithm will be significantly outperformed by any of the compared algorithms is always lower. As for MAE, we see that the rope probabilities are no longer as high. Therefore, the probability that $Med{-}\mathrm{k}{-}NN$ with LM^3L significantly outperforms any of the compared algorithms, with respect to MAE, is clearly dominant.

The above heatmaps offer a general overview of the Bayesian test results. To have a more specific view we focus now on two main comparisons: one for the best algorithm obtained against the rest of the algorithms, and another one for the Euclidean distances against the distances learned by LM^3L within the same classifier, for each of the classifiers analyzed in this study. For this purpose, we have obtained the ternary simplex plots and the posterior distribution barplots for each of the pairwise comparisons, which are available in Appendix A.

The first comparison with Bayesian tests puts the classification model with Med-*k*-NN and the distance learned by LM^3L, which is the best performer according to the tables, against the rest of the classifiers and distances. Figures 11-14 show the relevant Bayesian plots.

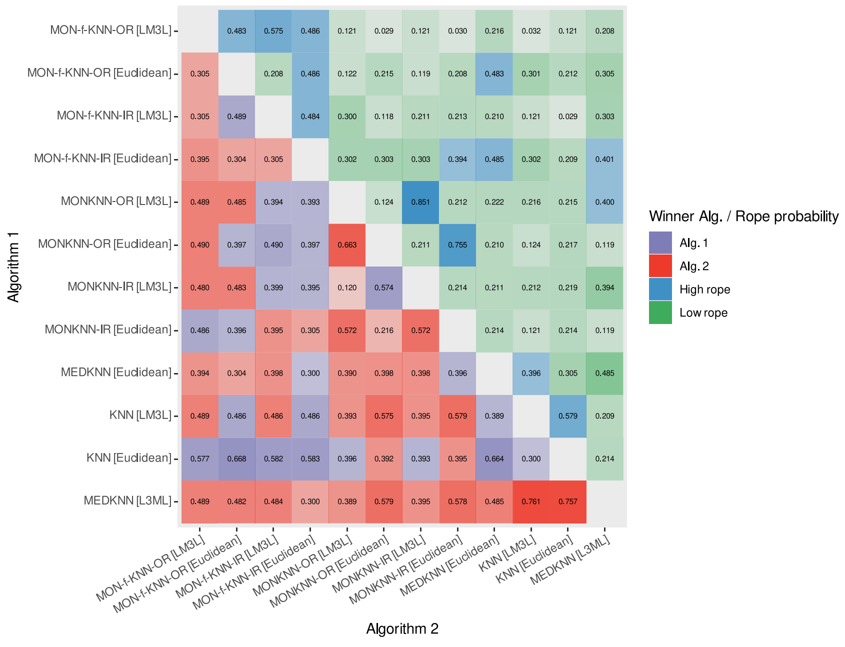

Fig. 2 Pairwise Bayesian comparisons of the MAE scores obtained by the different algorithms

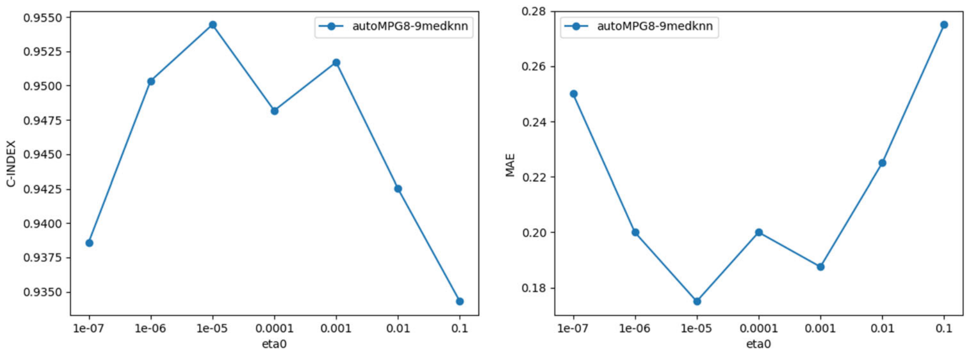

Fig. 3 Effect of η_0 on C-Index and MAE in autoMPG8

This comparison confirms what we had already observed in the heatmaps: in all the algorithms there is a clear trend towards the regions associated with Med-k-NN with LM^3L and the rope. In the case of C-Index there is a greater bias towards the rope, while for the MAE it becomes strongly apparent that the distributions are concentrated in the region of Med-k-NN with LM^3L, thus showing that this algorithm is most likely significantly outperforming the rest under this metric.

The second analysis with Bayesian tests compares, within the same classifier, the Euclidean distance and the distance learned by LM^3L. Figures 15-16 show the Bayesian plots obtained for this analysis. We can confirm, as already seen in the Tables, that LM^3L is able to outperform the Euclidean distance when using the non-monotonic majority-vote and median-vote nearest neighbor classifiers, although it is not significantly better than the Euclidean distance when comparing within each of the monotonic classifiers. According to the metrics, the MAE shows more bias towards the winner algorithm region, for each case, and the C-Index is more dominated by the rope. In any case, these diagrams show how dominant Med-k-NN with LM^3L is with respect to the Euclidean Med-k-NN. Together with the above comparison, LM^3L is still validated as the best alternative when used in conjunction with the median-vote nearest neighbors.

4.5 Analysis of hyperparameters

In this section, we analyze the hyperparameters of the proposed algorithm on some of the datasets used in the experiments above. The main parameters of LM^3L are:

- The initial learning rate for the gradient optimization η_0.
- The large margin λ in the objective function.
- The neighborhood size K.
- The maximum number of iterations of the gradient optimization.

We evaluate these hyperparameters in the datasets `autoMPG8` and `boston-housing`, which are both inspired by real world monotonic problems. We use LM^3L with the same Med-9-NN classifier used in the experiments.

Influence of the initial learning rate In what follows, we analyze the impact of the initial learning rate on the above-mentioned datasets. With the rest of the parameters fixed as

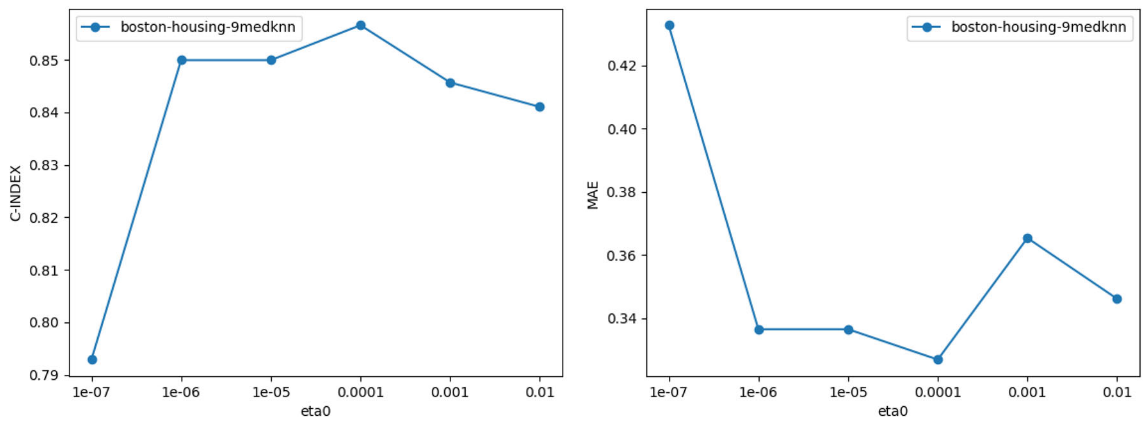

Fig. 4 Effect of η_0 on C-Index and MAE in boston-housing

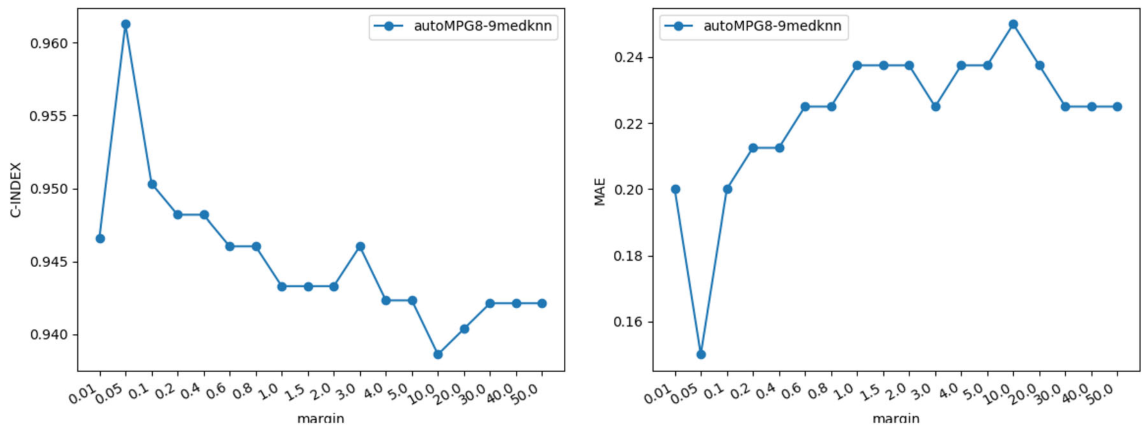

Fig. 5 Effect of λ on C-Index and MAE in autoMPG8

in the initial experimentation, we vary η_0 according to the set of values $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The results are shown in Figs. 3 and 4.

In the graphics we can see that, when η_0 ranges from 10^{-6} to 10^{-3}, the adaptive update of the learning rate is enough to lead to competitive results. In contrast, when eta_0 is too low or too high, it negatively influences the optimization process and the final metrics are suboptimal, despite the adaptability of η during the gradient optimization.

Influence of the margin The margin λ determines the degree to which the most distant class is kept away in the triplets that are used during the optimization process. When the margin is low, the three ordered elements in the triplet are closer than when the margin is high. We analyze the impact of λ for the set of values

$$\{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, \\ 3.0, 4.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0\}$$

with the other parameters fixed, in Figs. 5 and 6.

Here, it is readily visible that the highest levels of performance are achieved when the margins are below 1, which tells us that it is interesting to keep the elements of the triplets close together (as long as they are correctly ordered). The amplitude of the optimal margin range seems to be small as well, so it is crucial to specify this margin adequately to achieve optimal performance.

Influence of neighborhood size The neighborhood size K determines how many neighbors are considered to compute the triplets around each anchor sample in the dataset. This parameter gives a local character to the algorithm, as only nearby samples will be considered for each element. If K is low, only a few nearest neighbors will be used to compute the triplets. If K is high, the triplets will take into account most of the dataset. A lower value of K also translates into higher efficiency.

We analyze the impact of the neighborhood size used with the set of values in the range from 10 to 100 with a step of 5. Figures 7 and 8 show the effect of the neighborhood size on the C-Index and MAE.

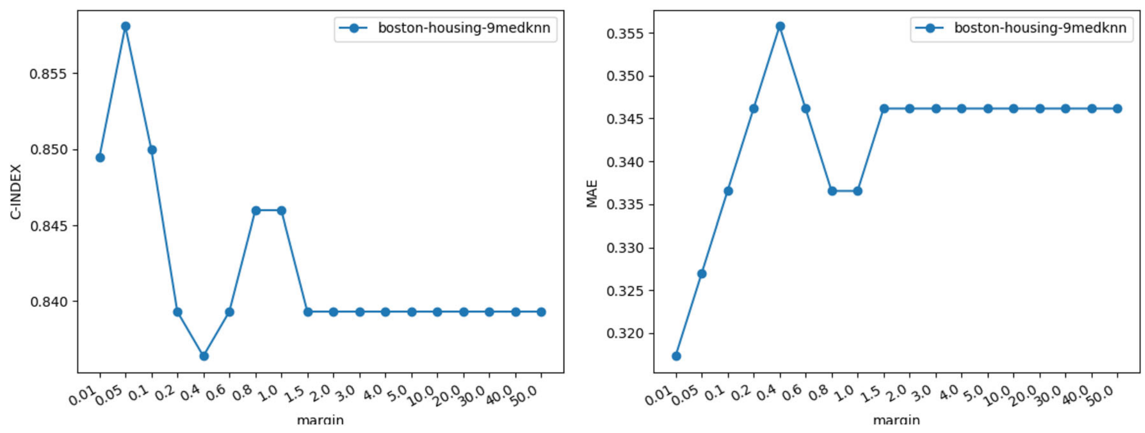

Fig. 6 Effect of λ on C-Index and MAE in boston-housing

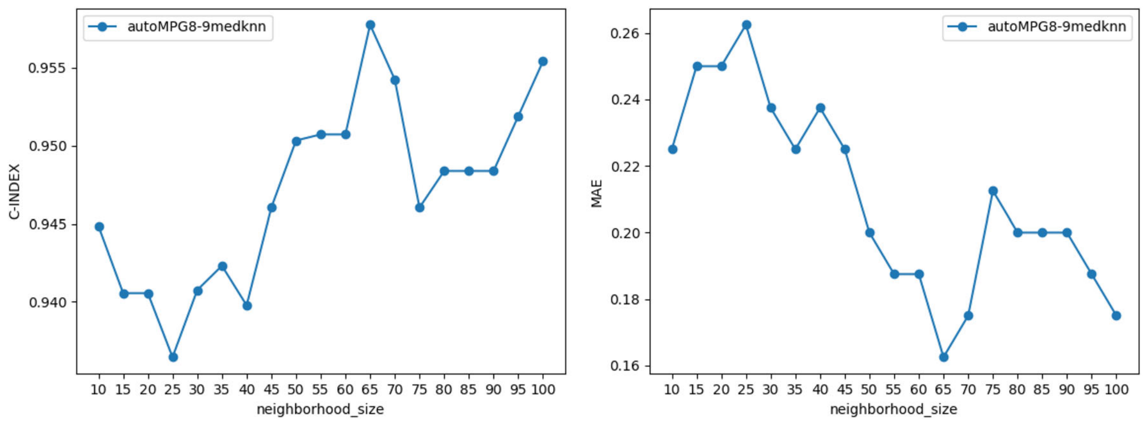

Fig. 7 Effect of K on C-Index and MAE in autoMPG8

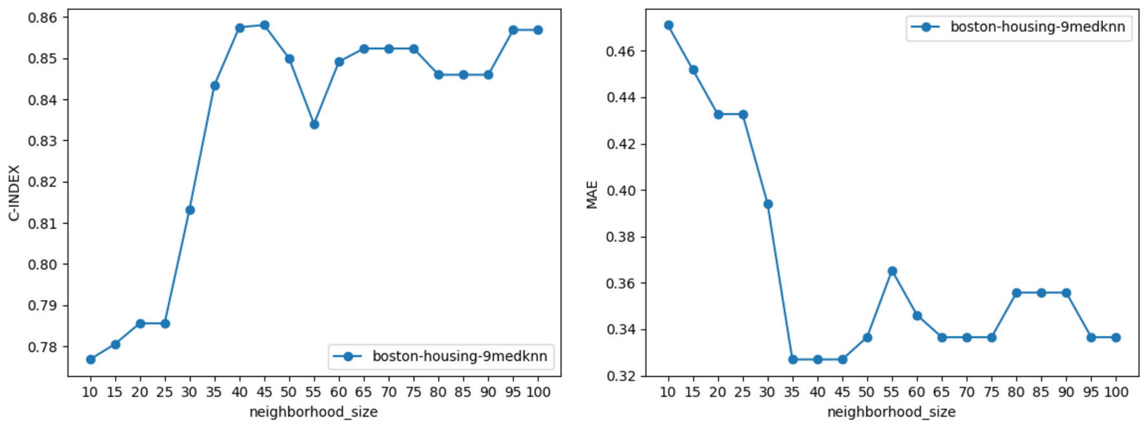

Fig. 8 Effect of K on C-Index and MAE in boston-housing

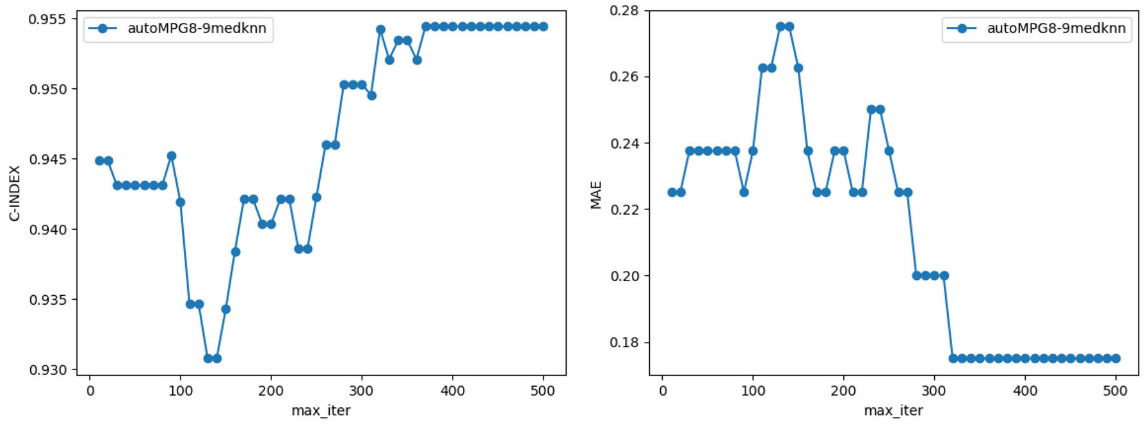

Fig. 9 Effect of the number of iterations on C-Index and MAE in autoMPG8

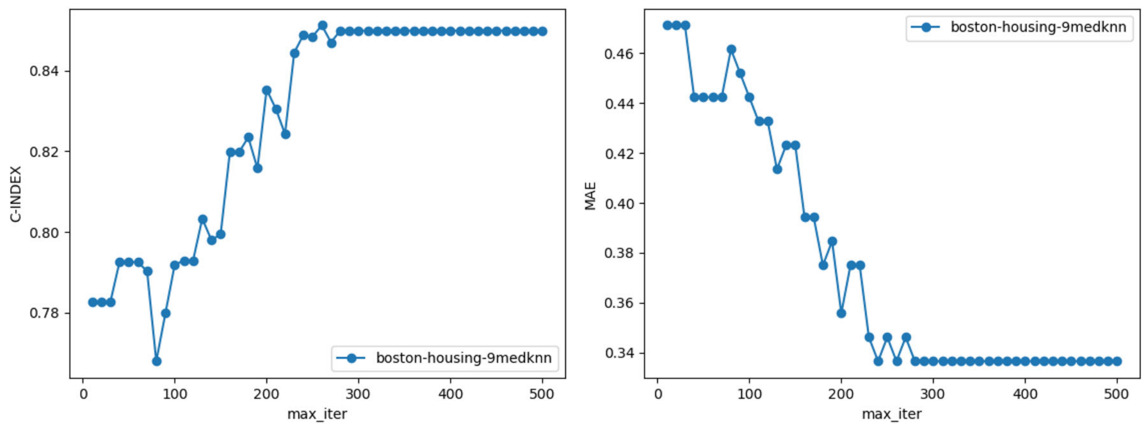

Fig. 10 Effect of number of iterations on C-Index and MAE in boston-housing

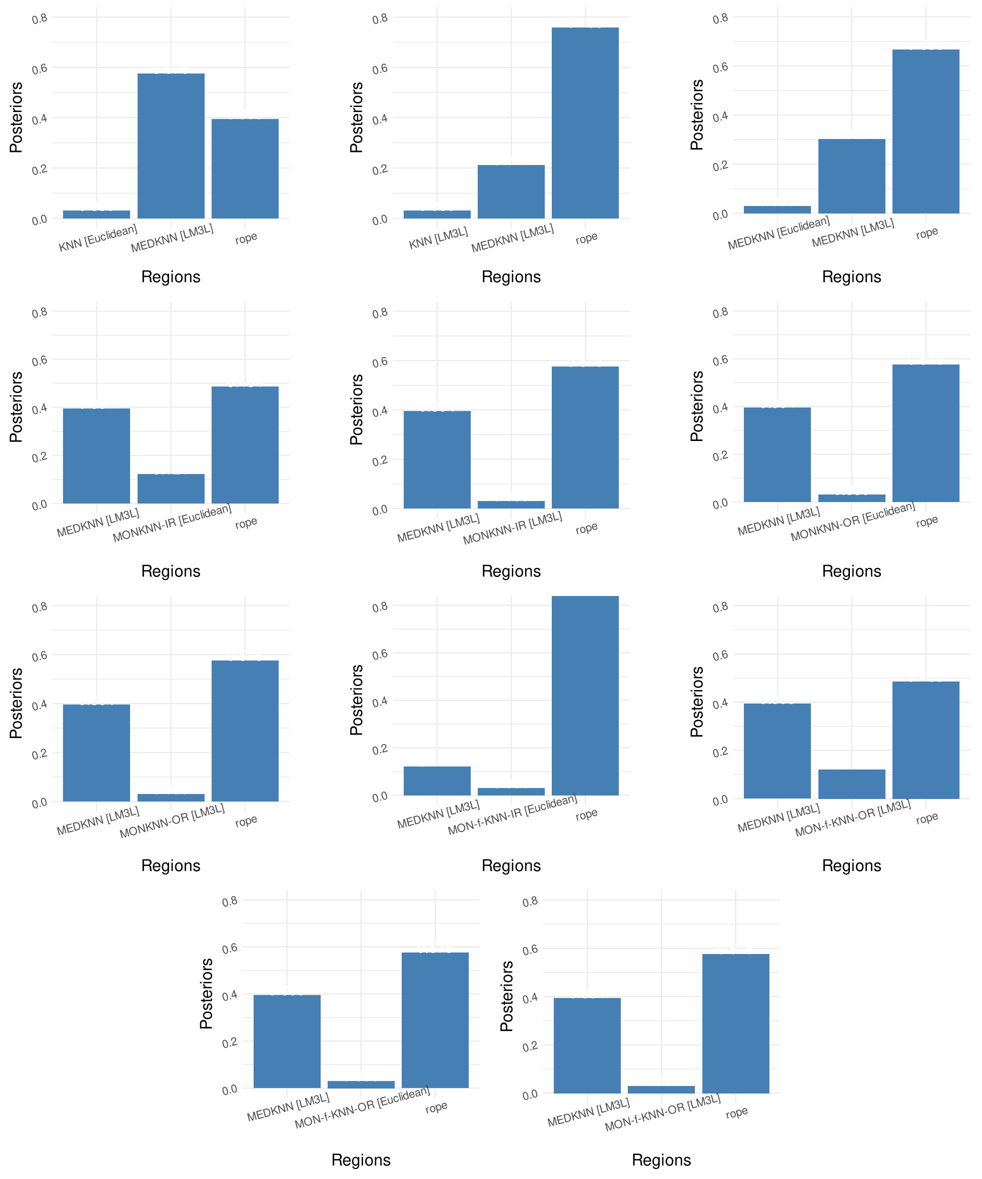

Fig. 11 Posterior distributions using C-Index for the Bayesian comparison between the best model and the rest of the classifiers and distances

In the figures we can observe that a good performance of the algorithm is usually achieved when the neighborhood size is 50 or higher. The optimal value may vary but, in general, a higher quality is obtained when the neighborhood size is in this range.

Influence of the number of iterations Lastly, we study how the number of iterations of the gradient optimization affects the convergence of the algorithm. Figures 9 and 10 show the effect of the number of iterations on the C-Index and MAE, in a range from 10 to 500 with a step of 10.

From the graphics we can conclude that the algorithm seems to converge with a number of iterations around 300, and there does not seem to be overfitting, as a higher number of iterations does not imply a worsening in the values of the metrics in this case.

5 Conclusion

In this paper, we have presented a new distance metric learning algorithm developed specifically for monotonic classification that, for the first time, exploits the potential of linear transformations to reduce the non-monotonicity of the dataset, thanks to the use of M-matrices. In addition, the distances learned allow us to improve the classification performance of the classifiers analyzed.

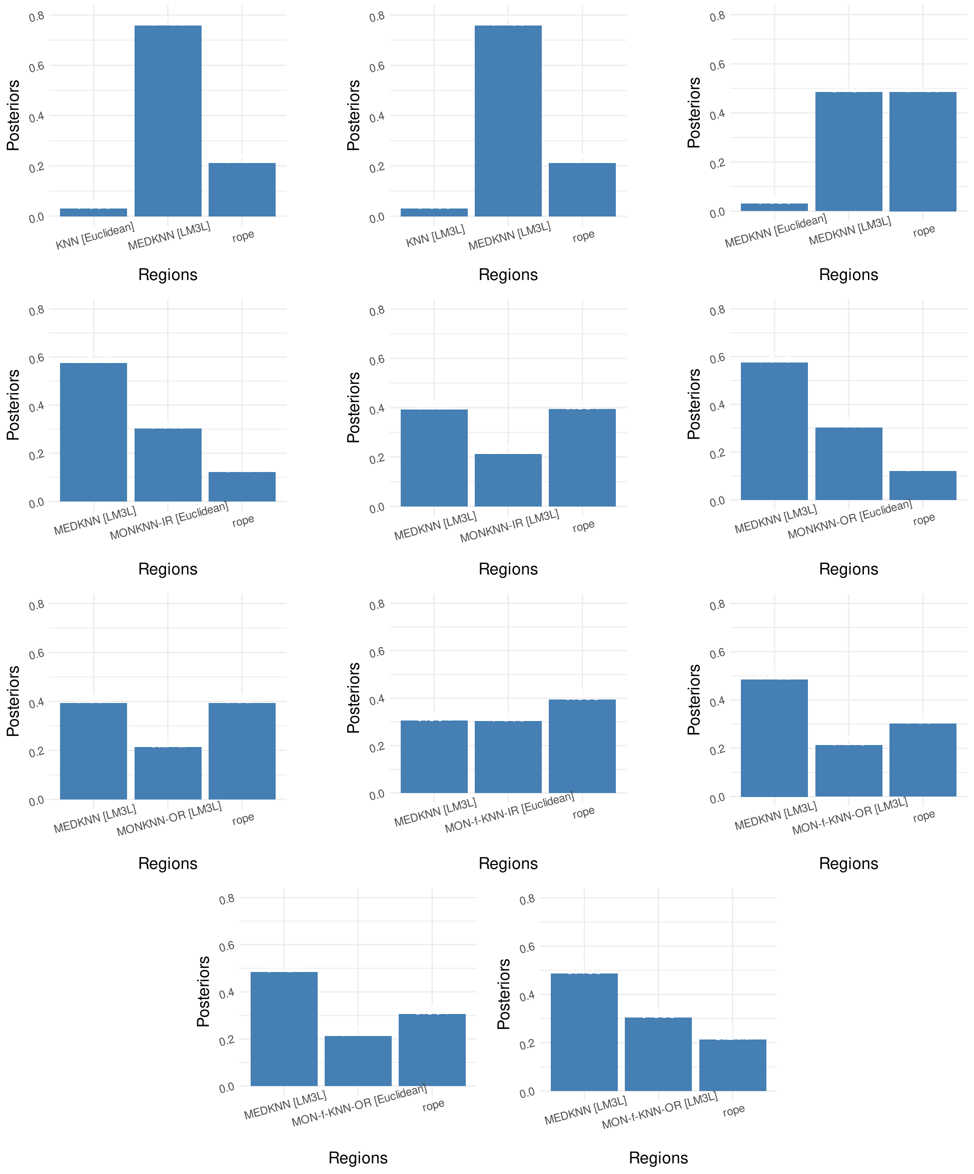

Fig. 12 Posterior distributions using MAE for the Bayesian comparison between the best model and the rest of the classifiers and distances

The results, supported by a Bayesian analysis, have shown that LM^3L combined with the median vote nearest neighbors classifier can outperform even the monotonic distance-based classifiers. In addition, the transformation of the space performed by LM^3L allows the number of non-monotonic pairs in the dataset to be reduced without introducing any false new monotonic constraints. LM^3L is thus presented as an alternative to consider in monotonic classification problems based on distances or similarities.

A Bayesian test simplex plots and posterior distributions

This Appendix shows the pairwise Bayesian diagrams for the two comparisons described in Section 4.4. The simplex plots are ternary plots displaying a sample of the posterior distribution. There are three regions that correspond to either algorithm having a performance advantage and to the rope (practical equivalence); each region is centered at one different vertex of the simplex, so that a greater tendency of the points towards a region represents a greater probability for that option. The posterior distributions are also shown as barplots where the bars represent the probabilities of each of the algorithms being better than the other, or the probability that they are practically equivalent. These plots are shown for the two metrics considered in the experiments: MAE and C-Index.

A.1 Comparison of the best model obtained with the rest of the algorithms

This section shows the results of the Bayesian tests that compare Med-k-NN with each of the other algorithms. The results are displayed in Figs. 11, 12, 13, and 14. The simplex diagrams and posterior distribution barplots are shown for both MAE and C-Index metrics.

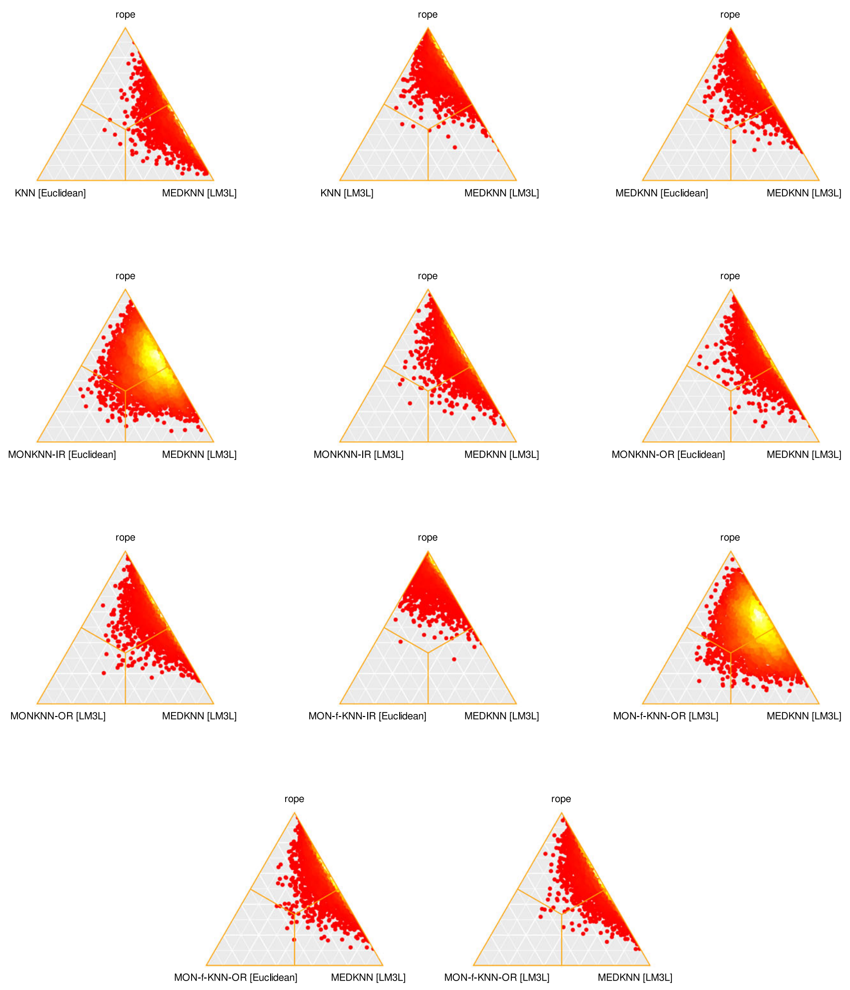

Fig. 13 Simplex plots using C-Index for the Bayesian comparison between the best model and the rest of the classifiers and distances

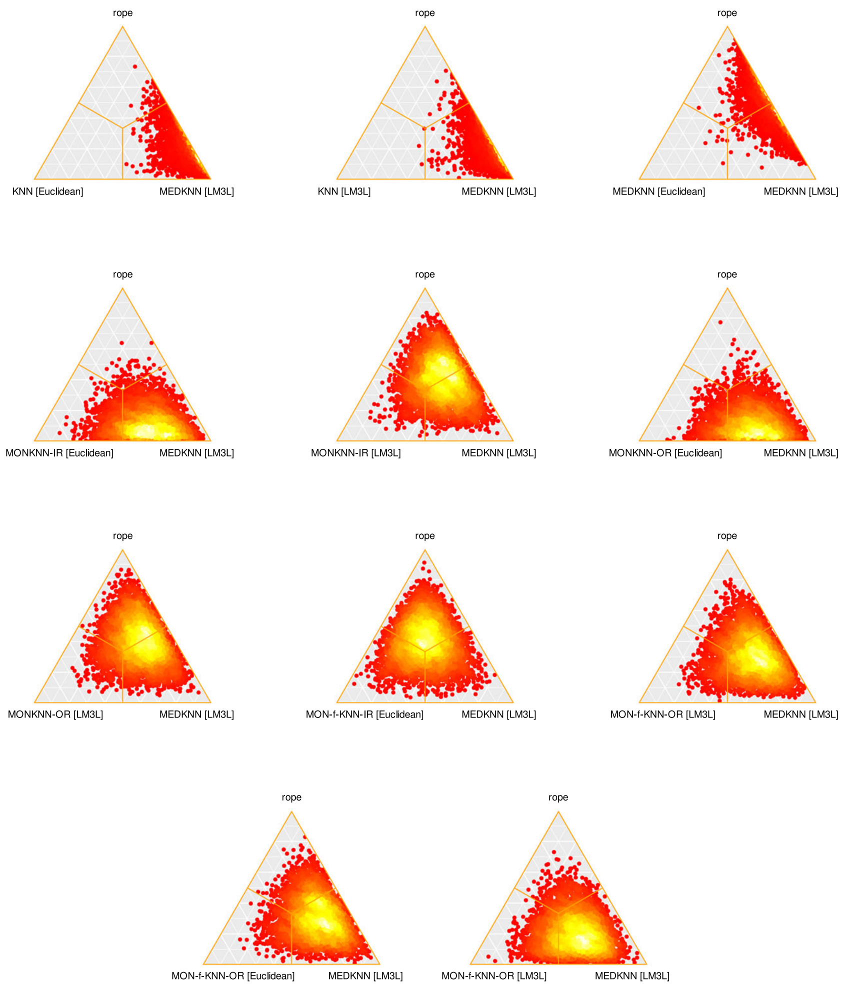

Fig. 14 Simplex plots using MAE for the Bayesian comparison between the best model and the rest of the classifiers and distances

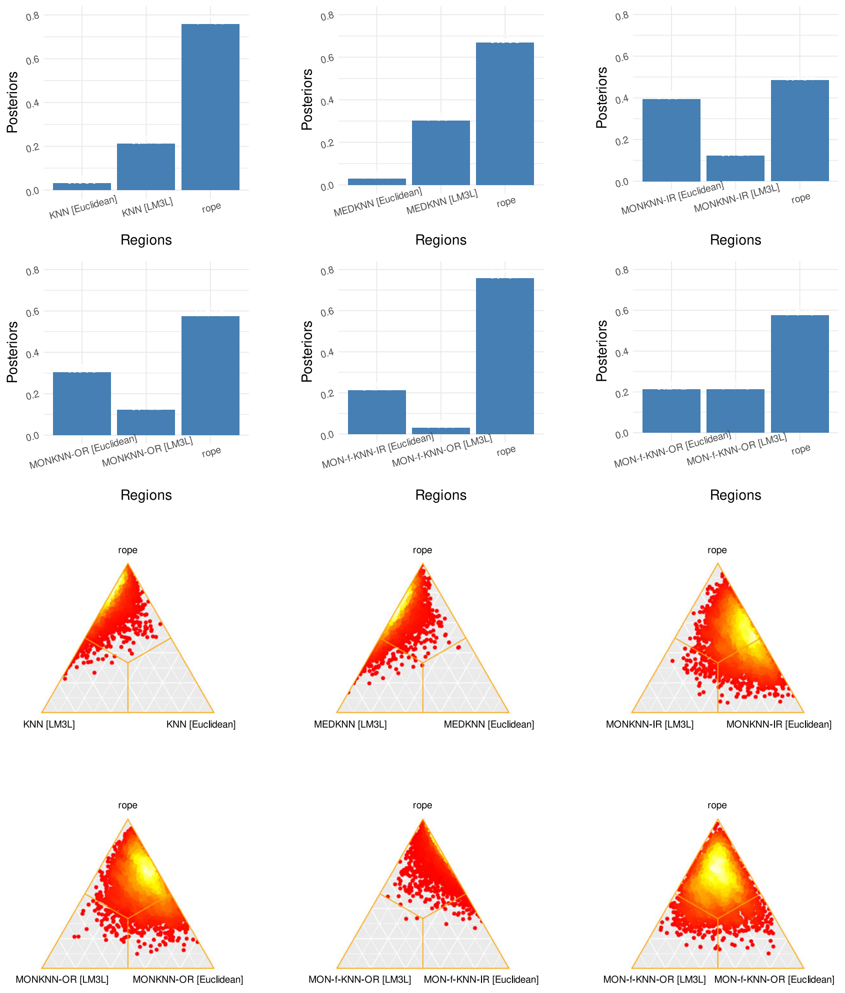

Fig. 15 Simplex plots and posterior distributions using C-Index for the Bayesian comparison between LM^3L and the Euclidean distance for each classifier

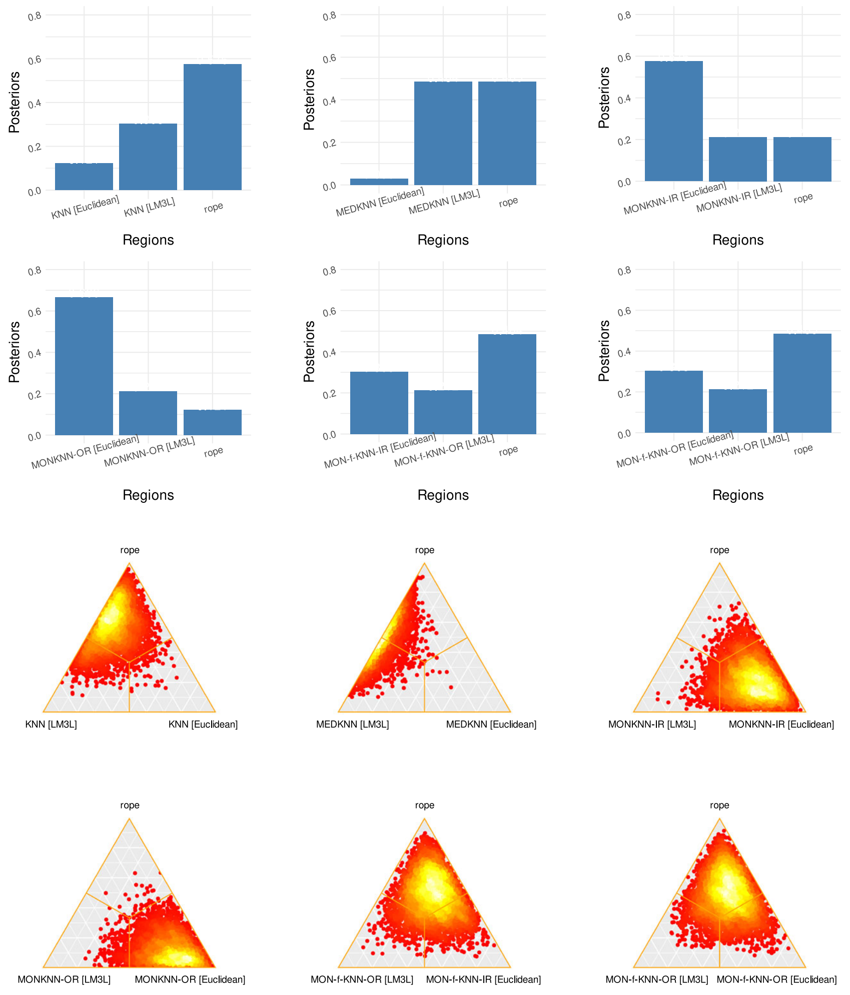

Fig. 16 Simplex plots and posterior distributions using MAE for the Bayesian comparison between LM^3L and the Euclidean distance for each classifier

A.2 Comparison of distances within the same algorithm

This section shows the results of the Bayesian tests that compare both Euclidean distance and the distance learned by LM^3L for each of the classifiers used in the experiments. The results are shown in Figs. 15-16. The simplex diagrams and posterior distribution barplots are shown for both MAE and C-Index metrics.

Author Contributions Juan Luis Suárez: Conceptualization, Methodology, Software, Investigation, Writing - review & editing, Validation. Germán González-Almagro: Conceptualization, Methodology, Writing - review & editing. Salvador García: Conceptualization, Methodology, Writing - review & editing, Investigation, Funding acquisition, Supervision. Francisco Herrera: Funding acquisition, Project administration, Supervision.

Funding Funding for open access publishing: Universidad de Granada/CBUA. Funding for open access charge: Universidad de Granada / CBUA. Our work has been supported by the research projects PID2020-119478GB-I00 and A-TIC-434-UGR20, and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

Data availability and access The datasets used in this work are publicly available in the KEEL [37] repository (https://sci2s.ugr.es/keel/datasets.php) and in the ORCA [17] repository (https://github.com/ayrna/orca), as a download link (http://www.uco.es/grupos/ayrna/ucobigfiles/datasets-orreview.zip). Any of the specific folds used in the experimentation can be requested to the corresponding author on request.

Declarations

Competing interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Compliance with ethical standards The study was conducted in compliance with ethical standards, including obtaining informed consent from all participants and ensuring the confidentiality and anonymity of their data.

Ethical and informed consent for data used The data used in this study were publicly available datasets. No ethical approval was required for the use of these datasets. We adhered to best practices for data management, including ensuring the quality and reliability of the data and properly citing the source of the data.

References

1. Ben-David A (1992) Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. Decis Sci 23(6):1357–1372
2. Benavoli A, Corani G, Demšar J, Zaffalon M (2017) Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. J Mach Learn Res 18(1):2653–2688
3. 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
4. Berman A, Plemmons RJ (1994) Nonnegative matrices in the mathematical sciences. SIAM
5. Cano JR, Aljohani NR, Abbasi RA, Alowidbi JS, Garcia S (2017) Prototype selection to improve monotonic nearest neighbor. Eng Appl Artif Intell 60:128–135
6. Cano JR, Gutiérrez PA, Krawczyk B, Woźniak M, García S (2019) Monotonic classification: an overview on algorithms, performance measures and data sets. Neurocomputing 341:168–182
7. Carrasco J, García S, Rueda M, Das S, Herrera F (2020) Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: practical guidelines and a critical review. Swarm Evol Comput 54:100665
8. Chen CC, Li ST (2014) Credit rating with a monotonicity-constrained support vector machine model. Expert Syst Appl 41(16):7235–7247
9. Chen D, Ye W (2022) Monotonic neural additive models: pursuing regulated machine learning models for credit scoring. In: Proceedings of the third ACM international conference on AI in finance, pp 70–78
10. Chen H, Jia Y, Ge J, Gu B (2022) Incremental learning algorithm for large-scale semi-supervised ordinal regression. Neural Netw 149:124–136
11. Cover TM, Hart PE et al (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27
12. Cunningham JP, Ghahramani Z (2015) Linear dimensionality reduction: survey, insights, and generalizations. J Mach Learn Res 16(1):2859–2900
13. Duivesteijn W, Feelders A (2008) Nearest neighbour classification with monotonicity constraints. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 301–316
14. Goldberger J, Hinton GE, Roweis ST, Salakhutdinov RR (2005) Neighbourhood components analysis. In: Advances in neural information processing systems, pp 513–520
15. Gönen M, Heller G (2005) Concordance probability and discriminatory power in proportional hazards regression. Biometrika 92(4):965–970
16. González S, García S, Li ST, John R, Herrera F (2021) Fuzzy k-nearest neighbors with monotonicity constraints: moving towards the robustness of monotonic noise. Neurocomputing 439:106–121
17. Gutiérrez PA, Perez-Ortiz M, Sanchez-Monedero J, Fernandez-Navarro F, Hervas-Martinez C (2015) Ordinal regression methods: survey and experimental study. IEEE Trans Knowl Data Eng 28(1):127–146
18. Keller JM, Gray MR, Givens JA (1985) A fuzzy k-nearest neighbor algorithm. IEEE Trans Syst Man Cybern (4):580–585
19. Kotłowski W, Słowiński R (2008) Statistical approach to ordinal classification with monotonicity constraints. In: Preference learning ECML/PKDD 2008 workshop
20. Leslie D (2019) Understanding artificial intelligence ethics and safety. arXiv preprint arXiv:1906.05684
21. Liu W, Xu D, Tsang IW, Zhang W (2018) Metric learning for multi-output tasks. IEEE Trans Pattern Anal Mach Intell 41(2):408–422
22. Ma Z, Chen S (2018) Multi-dimensional classification via a metric approach. Neurocomputing 275:1121–1131
23. Marques-Silva J, Gerspacher T, Cooper MC, Ignatiev A, Narodytska N (2021) Explanations for monotonic classifiers. In: International conference on machine learning. PMLR, pp 7469–7479
24. Nguyen B, Morell C, De Baets B (2017) Supervised distance metric learning through maximization of the jeffrey divergence. Pattern Recogn 64:215–225
25. Nguyen B, Morell C, De Baets B (2018) Distance metric learning for ordinal classification based on triplet constraints. Knowl-Based Syst 142:17–28
26. Petersen KB, Pedersen MS et al (2008) The matrix cookbook. Technical University of Denmark 7(15):510
27. Qian Y, Xu H, Liang J, Liu B, Wang J (2015) Fusing monotonic decision trees. IEEE Trans Knowl Data Eng 27(10):2717–2728
28. Sang B, Chen H, Yang L, Wan J, Li T, Xu W (2022) Feature selection considering multiple correlations based on soft fuzzy dominance rough sets for monotonic classification. IEEE Trans Fuzzy Syst 30(12):5181–5195
29. Schoeffer J, Kuehl N, Machowski Y (2022) "there is not enough information": on the effects of explanations on perceptions of informational fairness and trustworthiness in automated decision-making. In: 2022 ACM Conference on fairness, accountability, and transparency, pp 1616–1628
30. Schoeffer J, Kuehl N, Valera I (2021) A ranking approach to fair classification. In: ACM SIGCAS conference on computing and sustainable societies, pp 115–125
31. Suárez JL, García S, Herrera F (2020) pydml: a python library for distance metric learning. J Mach Learn Res 21(96):1–7
32. Suárez JL, García S, Herrera F (2021) Ordinal regression with explainable distance metric learning based on ordered sequences. Mach Learn 110(10):2729–2762
33. Suárez JL, García S, Herrera F (2021) A tutorial on distance metric learning: mathematical foundations, algorithms, experimental analysis, prospects and challenges. Neurocomputing 425:300–322
34. Suárez JL, González-Almagro G, García S, Herrera F (2022) A preliminary approach for using metric learning in monotonic classification. In: Advances and trends in artificial intelligence. Theory and practices in artificial intelligence: 35th International conference on industrial, engineering and other applications of applied intelligent systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings. Springer, pp 773–784

35. Tang M, Pérez-Fernández R, De Baets B (2020) Fusing absolute and relative information for augmenting the method of nearest neighbors for ordinal classification. Inf Fusion 56:128–140
36. Torresani L, Lee Kc (2007) Large margin component analysis. Adv Neural Inf Process Syst 19:1385
37. Triguero I, González S, Moyano JM, García López S, Alcalá Fernández J, Luengo Martín J, Fernández Hilario A, Díaz J, Sánchez L, Herrera F et al (2017) Keel 3.0: an open source software for multi-stage analysis in data mining. In: International journal of computational intelligence systems. Atlantis Press, pp 1238–1249
38. Truong T, Duong H, Le B, Fournier-Viger P, Yun U (2022) Frequent high minimum average utility sequence mining with constraints in dynamic databases using efficient pruning strategies. Appl Intell 52(6):6106–6128
39. Vargas VM, Gutiérrez PA, Barbero-Gómez J, Hervás-Martínez C (2023) Soft labelling based on triangular distributions for ordinal classification. Inf Fusion
40. Vargas VM, Gutiérrez PA, Hervás-Martínez C (2022) Unimodal regularisation based on beta distribution for deep ordinal regression. Pattern Recogn 122:108310
41. Wang F, Zhang C (2007) Feature extraction by maximizing the average neighborhood margin. In: 2007 IEEE conference on computer vision and pattern recognition. IEEE, pp 1–8
42. Wang J, Qian Y, Li F, Liang J, Ding W (2019) Fusing fuzzy monotonic decision trees. IEEE Trans Fuzzy Syst 28(5):887–900
43. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10(Feb):207–244
44. Zhang X, Jiang Z, Xu W (2022) Feature selection using a weighted method in interval-valued decision information systems. Appl Intell, pp 1–20

Juan Luis Suárez received the B.Sc. degrees in Mathematics and Computer Science, and the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2018, 2019 and 2024, respectively. He is currently a post-doctoral researcher in the Department of Computer Science and Artificial Intelligence at the University of Granada, and member of the Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI). His research interests include data mining, data preprocessing, no standard classification, distance metric learning and AI safety.

Germán González-Almagro received the Ph.D. and the M.Sc. degrees in data science from the University of Granada in 2019 and 2023, respectively. He is currently a research assistant at the Department of Computer Science and Artificial Intelligence and member of the Andalusian Data Science and Computational Intelligence Institute. His research revolves around data mining, data preprocessing, non-standard problems and semi-supervised learning.

Salvador García received the B.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently a Full Professor in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. Dr. García has published more than 125 papers in international journals (more than 90 in Q1), h-index 64. As edited activities, he is an Editor in Chief of "Information Fusion" (Elsevier), and an associate editor of "Swarm and Evolutionary Computation" (Elsevier), "AI Communications" (IOS Press) and "Machine Learning" (Springer) journals. He is a co-author of the books entitled "Data Preprocessing in Data Mining", "Learning from Imbalanced Data Sets" and "Big Data Preprocessing: Enabling Smart Data" published by Springer. His research interests include data science, data preprocessing, Big Data, evolutionary learning, deep learning and data-centric AI. He belonged to the list of the Highly Cited Researchers in the area of Computer Sciences (2014-2020): http://highlycited.com/ (Clarivate Analytics).

Francisco Herrera (Senior Member, IEEE) received the M.Sc. Degree in mathematics in 1988, and the Ph.D. degree in mathematics in 1991, both from the University of Granada, Spain. He is a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and Director of the Andalusian Research Institute of Data Science and Computational Intelligence (DaSCI). He's an Academician in the Royal Academy of Engineering (Spain). He has been the Supervisor over 60 Ph.D. students. He has published more than 600 journal papers, receiving more than 147 000 citations (Scholar Google, H-index 182). He has been nominated as a Highly Cited Researcher (in the fields of computer science and engineering, respectively, 2014 to present, Clarivate Analytics). He acts as Editorial Member of a dozen of journals. His current research interests include among others, computational intelligence, information fusion and decision making, trustworthy artificial intelligence and data science (including data preprocessing, prediction and big data).