



Fuzzy k -nearest neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise

Sergio González^{a,*}, Salvador García^a, Sheng-Tun Li^{b,c}, Robert John^d, Francisco Herrera^{a,e}

^a Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

^b Department of Industrial and Information Management and the Institute of Information Management, National Cheng Kung University, Tainan 701, Taiwan

^c Center for Innovative FinTech Business Models, National Cheng Kung University, Tainan 701, Taiwan

^d ASAP Research Group School of Computer Science, University of Nottingham, NG8 1BB Nottingham, UK

^e Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:

Received 20 July 2019

Revised 10 December 2019

Accepted 16 December 2019

Available online 26 January 2021

Keywords:

Fuzzy k -NN

Monotonic constraints

Ordinal classification

Ordinal regression

Class noise

ABSTRACT

This paper proposes a new model based on Fuzzy k -Nearest Neighbors for classification with monotonic constraints, Monotonic Fuzzy k -NN (MonFkNN). Real-life data-sets often do not comply with monotonic constraints due to class noise. MonFkNN incorporates a new calculation of fuzzy memberships, which increases robustness against monotonic noise without the need for relabeling. Our proposal has been designed to be adaptable to the different needs of the problem being tackled. In several experimental studies, we show significant improvements in accuracy while matching the best degree of monotonicity obtained by comparable methods. We also show that MonFkNN empirically achieves improved performance compared with Monotonic k -NN in the presence of large amounts of class noise.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

Monotonic constraints are prior-knowledge of some ordinal classification or regression problems about the order relationships between attributes and class labels [9]. Consider the example of house pricing. The following constraints are applied: A bigger house in the same neighborhood is constrained by higher prices as compared to smaller houses with the same features. That is, the classifier decisions should not decrease in the presence of better features while the rest remains the same. These prior constraints are required by many real-life evaluation problems, such as credit risk modeling [12] and lecturer evaluation [7]. These problems are known as Classification with Monotonic Constraints or Monotonic Classification [3].

These learning tasks have additional objectives besides accurate models, such as the monotonic consistency of predictions and minimization of the misclassification costs. The latter is also relevant since the errors between ordered classes do not hold the same importance. More metrics must be used during the learning and validation of the models. However, these other objectives may

impair accuracy [4]. Hence, a fair balance must be sought between the different needs of each problem.

Standard classifiers are discouraged for monotonic classification since they do not contemplate these constraints and their predictions violate the monotonicity required by certain applications. A classic example of these non-monotonic models is the standard decision tree [3]. Standard k -Nearest Neighbors algorithm also does not take these restrictions into account, which may lead to further harm as a result of their presence in preprocessing techniques [22].

In recent years, new algorithms have been designed to minimize the number of monotonic violations in their predictions [3,23,9], i.e. the number of pairs of instances that break monotonicity [3]. To do so, some approaches focus their entire learning mechanism just on monotonicity. This strategy usually achieves completely monotonic models, but it could lead to wrong generalizations being made that are different to the knowledge of the problem. Others infer monotonic relations from the training set while maximizing their accuracy. These models have been adapted from different families of classifiers [9], such as decision trees [3,32,34], support vector machines [12], fuzzy model based classifiers [1,29], neural networks [17,40] and ensemble learning [13,36,23].

Instance-based learning has proven to be a good approach for monotonic classification [2,15,31,18]. However, some of these

* Corresponding author.

E-mail addresses: sergiogvz@decsai.ugr.es (S. González), salvag@decsai.ugr.es (S. García), stli@mail.ncku.edu.tw (S.-T. Li), robert.john@nottingham.ac.uk (R. John), herrera@decsai.ugr.es (F. Herrera).

methods, such as Monotonic k -Nearest Neighbors [15] (MkNN), need to learn from a fully monotonic set to ensure monotonic predictions. This is rarely the case in real-life scenarios, where class noise and discrepancies are common. Therefore, data preprocessing [21,33,8,22] and relabeling strategies [35,16] must be used to remove non-monotonic samples or to change their class labels in order to force a monotonic set.

In standard classification, Fuzzy k -Nearest Neighbors [25] is a very solid method with high performance, thanks to its high robustness to class noise [14]. This class noise robustness mainly lies in the extraction of the class memberships for the crisp training samples by nearest neighbor rule. In this process, the class memberships of noisy instances are shared with surrounding classes and the incorrectly assigned class loses its influence. However, these mechanisms do not consider monotonic constraints and Fuzzy k -NN cannot deal with monotonic violations or monotonic noise in the training set.

In this paper, a new model designed on the basis of Fuzzy k -NN with notions of MkNN is proposed to take monotonic constraints into account, and is called Monotonic Fuzzy k -Nearest Neighbors (MonFkNN). MonFkNN has been designed with three desired features:

- (i) Robustness against monotonic violations.
- (ii) Monotonic predictions without a pure monotonic training set.
- (iii) Flexibility in its configurations covering different needs of performance.

With these objectives in mind, MonFkNN has been designed with new mechanisms to manage monotonicity constraints and the monotonic violations in the training set. The main contributions of the MonFkNN design are:

- (i) The initial robustness of Fuzzy k -NN has been redesigned to mitigate the influence of monotonic violations. Firstly, the violations due to sample replicas with different classes are joined to form one class membership. Then, our approach incorporates a strictly monotonic nearest neighbor rule to the calculation of the memberships of the training examples.
- (ii) These monotonically constrained memberships and their medians are used in the prediction phase. The class memberships aggregation of MonFkNN is also monotonically constrained by the nearest neighbor extraction or a penalty to the contribution of non-monotonic instances.
- (iii) MonFkNN was built as a flexible classifier that covers different necessities of monotonicity and accuracy by tuning its parameters. It can be configured with a rigidly monotonic or standard k -NN rule if monotonicity or precision is preferred in the predictions, respectively.

All these mechanisms reinforce the robustness of our proposal against monotonic noise without the need for relabeling. We understand monotonic noise as being the actual noise that can alter the class labels and, as a result, change the monotonic constraints among the samples in the data. Their parameters make our proposal adaptable to the different objectives of monotonic classification. We distinguish two different parameter configurations: a pure monotonic version in which monotonicity is prioritized, and an approximate configuration that focuses more on the prediction accuracy.

We have performed several empirical studies to verify the desired features of MonFkNN. First, different behaviors of its two configurations are empirically analyzed and compared to the original FkNN. Then, our proposal is compared with 7 methods from the state-of-the-art, exhibiting substantial improvements in accu-

racy and maintaining the best degree of monotonicity. Finally, the robustness of our method against monotonic noise, i.e. monotonic violations, is shown in contrast to MkNN. In this last experiment, MonFkNN performs considerably better than Monotonic k -NN in scenarios with large amounts of class noise. The experimental framework used consists of 12 data-sets commonly used in monotonic classification, 7 monotonic classifiers and 3 metrics covering different aspects of performance: Accuracy, Mean Absolute Error and Non-Monotonic Index. All results are additionally validated with the non-parametric statistical Wilcoxon and Friedman rank [20,19] and Bayesian Sign tests [5].

The paper is organized as follows. In Section 2, we present the problem of classification with monotonic constraints and the methods related to our proposal: MkNN and Fuzzy k -NN. Section 3 is dedicated to explaining our model MonFkNN in detail and its algorithmic differences as compared to FkNN. The experimental framework used in the different empirical studies is presented in Section 4. In Section 5, the previously mentioned empirical studies are carried out and analyzed. Finally, the main conclusions of this study are stated in Section 6.

2. Preliminaries

In this section, we introduce the preliminaries needed: Classification with monotonicity constraints, Monotonic k -Nearest Neighbors and the original Fuzzy k -Nearest Neighbors.

2.1. Monotonic classification

Monotonic classification [9] is an ordinal regression problem with monotonic constraints relating to the order of the variables and the class labels. Ordinal regression and/or classification can be seen as a nonstandard classification problem [11], which attempts to minimize the difference between the predicted labels and the real labels. Classification with monotonic constraints is also considered to be a nonstandard supervised learning problem [11].

Formally, monotonic classification aims to predict the class label y from input vector x with Q number of features, where $y \in \mathcal{Y} = \{l_1, l_2, \dots, l_c\}$ and x represents an individual of our classification problem. The categories \mathcal{Y} are arranged in an order relation \prec as $l_1 \prec l_2 \prec \dots \prec l_c$. And, as the main property of monotonic classification, the attributes and class predictions are monotonically constrained by the problem prior-knowledge, i.e. $x \succeq x' \rightarrow f(x) \geq f(x')$ [26], where $x \succeq x'$ implies $\forall_{j=1, \dots, Q}, x_j \geq x'_j$, that is, x dominates x' . Therefore, the main objective is to build classifiers that do not violate these constraints, otherwise known as monotonic classifiers.

Two different types of monotonic classifiers can be distinguished: approximate monotonic models, which minimizes the number of monotonic violations in their decisions and pure monotonic classifiers, whose predictions are always monotonic concerning the training and future examples. The latter is hard to achieve, particularly in real-life applications where the training data-sets are rarely purely monotonic. To be considered monotonic, all of the pairs of instances in a data-set must be monotonic [2]: $x_i \succeq x_j \rightarrow y_i \geq y_j, \forall_{i,j}$.

2.2. Monotonic k -nearest neighbors

MkNN [15] modifies the standard nearest neighbor rule of the well-known lazy learning method to avoid monotonic violations in its predictions. To do so, MkNN computes for each new example x_i the range $r_i = [y_{min}, y_{max}]$ of valid class labels, which satisfies the monotonic constraints. The lower-bound y_{min} of r_i is computed as

the highest class label of all instances in the training set \mathcal{D} below the example x_i . Analogously the upper-bound y_{max} is the minimum class label of the instances in \mathcal{D} that are higher than x_i (see Eq. (1)).

$$r_i = \begin{cases} y_{min} = \max\{y \mid (x, y) \in \mathcal{D} \wedge x_i \succeq x\} \\ y_{max} = \min\{y \mid (x, y) \in \mathcal{D} \wedge x \succeq x_i\} \end{cases} \quad (1)$$

Two different MkNN variants can be distinguished depending on how the neighbors are extracted for a new instance x_i . The *InRange* variant considers the k nearest examples x_j with their class labels y_j in the range $[y_{min}, y_{max}]$. The *OutOfRange* version extracts first the k nearest neighbors x_j and then, those neighbors outside of the range r_i are filtered out from the decision. If all of them are removed, a random label in r_i is chosen. As in the standard k -NN method, the majority class among the k neighbors is used as the predicted label.

MkNN is one of the methods that require monotonic data-sets to work properly [15]. Since, with monotonicity violations, the range r_i could not be correctly computed, a relabeling technique should be used to transform the non-monotonic training data into monotonic data. These techniques intend to identify and remove the monotonicity violations by making the fewest possible changes with minimum class difference. [15,35,16].

2.3. Fuzzy k -Nearest Neighbors

Fuzzy Sets [39] express the uncertainty of the example memberships to each class label. The memberships of the example x_i are represented as a degree of each class belonging $u_i = (u_{i1}, u_{i2}, \dots, u_{ic})$, where $u_{il} \in [0, 1]$ and $\sum_{l=1}^c u_{il} = 1$. Nowadays, development in fuzzy sets and classifiers is still an ongoing process [37].

Fuzzy k -Nearest Neighbors algorithms [14] incorporate fuzzy concepts into the classical k -NN decision rule to learn from fuzzy sets and produce fuzzy classification rules. Recently, different approaches have been proposed based on distinct fuzzy set extensions. However, the original Fuzzy k -NN [25] (FkNN) is still one of the best approaches [14]. Recent approaches provide for the optimization of parameters in FkNN [6].

For a given new instance x_i , Fuzzy k -NN [25] extracts its K nearest neighbors in the same manner as the standard k -NN. Then, its memberships for each class l are computed with the following expression:

$$u(x, l) = \frac{\sum_{j=1}^K u(x_j, l) * \frac{1}{\|x-x_j\|^{(m-1)}}}{\sum_{j=1}^K \frac{1}{\|x-x_j\|^{(m-1)}}} \quad (2)$$

As shown in Eq. (2), the membership $u(x_i, l) = u_{il}$ of sample x_i to class l is assigned with the product of the class membership $u(x_j, l)$ of the neighbors x_j and the inverse of their distances to x_i . The latter serves as a weight that biases towards the memberships of nearer samples. The parameter m determines the degree of influence of the neighbor distances. The recommended value $m = 2$ [25] makes the contributions of the neighboring samples reciprocal to their distances. A crisp class label for the example x_i can be decided as being the label l with the greatest membership degree u_{il} .

Facing a labeled training set, Fuzzy k -NN [25] brings it into a fuzzy set with sample memberships using the nearest neighbor rule. For each training sample x_i , k nearest neighbors are extracted using the leave-one-out scheme. Then memberships $u(x_i, l)$ for each class l are computed according to Eq. (3) with the number of neighbors nn_l found for each class l . This transformation has proven useful against noisy samples as the memberships lose influence as they are spread to the surrounding classes (not the assigned class).

$$u(x_i, l) = \begin{cases} 0.51 + 0.49 * (nn_l/k), & \text{if } y_i = l \\ 0.49 * (nn_l/k), & \text{otherwise} \end{cases} \quad (3)$$

3. Monotonic fuzzy k -Nearest Neighbors

In this section, we explain our approach in detail – MonFkNN and all its mechanisms that consider monotonicity constraints. In SubSection 3.1, we explain how MonFkNN gives a final class from class memberships in a more proper manner according to monotonicity. SubSection 3.2 is dedicated to the extraction of the class memberships from the training set and redesigned to reduce the impact of monotonic noise without the need for monotonic relabeling. In SubSection 3.3, the class membership aggregation built-in MonFkNN is explained and related to the robustness and flexibility of the classifier using its parameters. Finally, we discuss the algorithmic differences between our proposal and the original FkNN in SubSection 3.4.

3.1. From class memberships to the final class label

Since FkNN works with class memberships, a mechanism that respects monotonicity is needed to get a final class from a vector whose elements sum up to the value of one. The class with the greatest membership is the most common decision in multiple classifiers. The original Fuzzy k -NN gives their crisp predictions as the class label with the highest membership.

However, this might not be appropriate for scenarios with monotonic constraints. For example, let $x_i \leq x_j$ and their class memberships $u_i = (0.2, 0.2, 0.4, 0.2, 0.0)$ and $u_j = (0.0, 0.4, 0.3, 0.2, 0.1)$, then their final classes chosen with the highest membership break the monotonicity: $argmax(u_i) = l_3 > l_2 = argmax(u_j)$. Even though, the instance x_j has more weight values assigned to the higher labels than instance x_i . In fact, u_j weakly dominates u_i according to the *first degree stochastic dominance relation* (FSD) [28] since the x_i cumulative distribution function $U_i = (0.2, 0.4, 0.8, 1.0, 1.0)$ is greater, element by element, than $U_j = (0.0, 0.4, 0.7, 0.9, 1.0)$, that is, $u_i \preceq_{FSD} u_j \iff (\forall l \in \mathcal{Y})(U_i(l) \geq U_j(l))$. To make FSD applicable, class membership vectors are normalized to sum up to the value of one and treated as probability mass functions. Therefore, a cumulative distribution function U can be computed for given normalized class memberships, where FSD is defined. This transformation can be done thanks to the order relation between classes in monotonic classification. FSD is useful for defining monotonicity constraints in probabilistic classifications [31,30], with the expression $x_i \leq x_j \Rightarrow u_i \preceq_{FSD} u_j$.

Therefore, the function that transfers a membership vector to a class label must satisfy $u_i \preceq_{FSD} u_j \Rightarrow y_i \leq y_j$. Centrality measures, such as mean and median, have proven to be good solutions [28,31]. Particularly, the median is applicable to ordinal problems. Following the traditional definition of median as the 50th percentile, the median is computed as the range $[l_m, l_M]$:

$$\begin{aligned} l_m &= \min\{l \in \mathcal{Y} \mid U\{X \leq l\} \geq 1/2\} \\ l_M &= \max\{l \in \mathcal{Y} \mid U\{X \geq l\} \geq 1/2\} \end{aligned} \quad (4)$$

where l is a class label of possible labels \mathcal{Y} , $U\{X \leq l\}$ is the cumulative membership/probability of belonging to a class smaller or equal to l and $U\{X \geq l\}$ is the analogous definition for a class greater or equal to l .

Going back to the previous example, the classes for x_i and x_j chosen by the median does not break monotonicity: $med(u_i) = med(u_j) = 3$. For $l_m \neq l_M$, any class label l which $l_m < l < l_M$ must have a membership $u(l) = 0$ and $U(l_m) = U(l_M) = 1/2$. For example, instance x_i with class member-

ships $u_t = (0.2, 0.3, 0, 0.3, 0.2)$ could be assigned to the classes $med(u_t) = [2, 4] = 3$.

3.2. Class memberships robust to monotonic noise

In this subsection, the class membership calculation redesigned to monotonic classification is explained. The objective of this first stage is to fix or reduce the influence of non-monotonic examples in the classification. Our method uses the robustness of the traditional Fuzzy k -NN within the knowledge of the monotonic relations between the neighbors. Algorithm 1 summarizes the procedure of obtaining robust noise class memberships for the training set.

Algorithm 1. Training class memberships extraction

First, we have to deal with the simplest monotonic violations, that is, instances with the same input values and different classes (Lines 2–13 of Algorithm 1). These mislabels frequently appear in traditional data-sets [2] of classification with monotonic constraints as these sets are rankings or evaluations made by different experts.

Therefore, MonFkNN first substitutes the replicas of any example x with one feature vector x and its memberships $u(x)$. The membership $u(x, l)$ of the instance x to the class l is computed with the frequency of duplicated examples x_j in the training set \mathcal{D} belonging to class l ($y_j = l$), as shown in the following expression:

$$u(x, l) = \frac{|\{x_j \in \mathcal{D} | x_j = x \wedge y_j = l\}|}{|\{x_j \in \mathcal{D} | x_j = x\}|} \quad (5)$$

The class label of an instance x after the elimination of its replicas is obtained by the median of the resulting memberships, as shown in Line 13 of Algorithm 1. However, this vector will be used in the classification function with the membership aggregation as stated in the next subsection.

Then, MonFkNN estimates the memberships of the remaining instances, which corresponds to Lines 13–24 of Algorithm 1. This estimation is made using the information of the nearest neighbors of each instance. However, these nearest neighbors are extracted with a monotonic nearest neighbor rule (MkNN) instead of a traditional rule as we aim for memberships that respect monotonic constraints as much as possible. Algorithm 2 exemplifies the extraction of these monotonically constrained neighbors for a given instance x as in MkNN.

Algorithm 2. Monotonic nearest neighbor rule

```

1: function NeighborsAsMkNNx - tested sample,  $r$  - range of
   valid classes,  $k$  - considered neighbors,  $typeRange$  - inRange
   or outRange,  $\{\mathcal{D}, y\}$  - Training data-set
2:   initialize:  $nn = \{\}$ 
3:   for  $x_i \in \mathcal{D}$  do
4:     if  $typeRange == outRange$  or  $y_i \in r$  then
5:       if  $Size(nn) < k$  then
6:         Insert  $x_i$  in  $nn$ 
7:       else
8:          $x_{max} = \arg \max_{x_j \in nn} \|x - x_j\|$ 
9:         if  $\|x - x_i\| < \|x - x_{max}\|$ 
10:          Replace  $x_{max}$  by  $x_i$  in  $nn$ 
11:         end if
12:       end if
13:     end if
14:   end for
15:   output:  $nn$ 
16: end function

```

In this case, Algorithm 2 is configured as an *inRange* variant as pointed out in Line 16 of Algorithm 1. That is, the nearest neighbors of an example x_i are constrained to a range $r_i = [y_{min}, y_{max}]$ of possible classes (Line 17), which preserves the monotonicity of the data-set.

Once the nearest neighbors for each example x_i are obtained, the information of the neighbor classes is fused into x_i class memberships (Line 20). For an instance x_i , the membership $u(x_i, l)$ to class l is computed with the following expression:

$$u(x_i, l) = \begin{cases} RCr + (nn_l/k) * (1 - RCr) & \text{if } y_i = l \\ (nn_l/k) * (1 - RCr) & \end{cases} \quad (6)$$

where nn_l is the number of nearest neighbors of the class l , k the total number of neighbors extracted for instance x_i and y_i is the original class label of the example x_i . RCr is a new parameter called “Real Class relevance”.

Apart from the use of the monotonic nearest neighbor rule, the inclusion of RCr is another main difference between our approach MonFkNN and the original Fuzzy k -NN. RCr can be seen as the minimum membership assigned to original class y_i of the instance x_i , in case there are no neighbors labeled with y_i . In FkNN, RCr corresponds to the value of 0.51, that is, every instance maintains its real class, even those noisy examples surrounded by other classes. By being a parameter, our method lets the user control the treatment of monotonic noise.

There are some values for RCr in the range $[0, 1]$ that have very interesting and distinct behaviors. In the case of a really noisy data-set where no labels can be trusted, RCr could be set to 0. This leaves all the responsibility to the calculation of the range of valid classes r_i and the nearest neighbors. In the presence of instances with the same input values and different classes, the user could choose only to treat them with $RCr = 1$. Finally, if practitioners want to consider the originally labeled instances, we recommend assigning RCr to 0.5. This value ensures that the actual class is within the set of medians. In contrast to Fuzzy k -NN and its 0.51, if all neighbors belong to a same single class that is different to the current class, our method forces to choose in between these two classes. Usually, this last value ($RCr = 0.5$) is a good trade-off, mainly stable and with better performance.

During this process, the impact of monotonic inconsistencies will be either reduced or fixed. The inconsistencies of instances with the same input vectors and different classes are completely fixed by being substituted by only a sample and class memberships with the information of their different classes. The mislabeled samples, i.e. noisy or non-monotonic examples, will have less influence towards their noisy class as they will be surrounded by more appropriated classes and their class memberships will be shared into classes in which they fit monotonically. This is the first mechanism of our method to alleviate the presence of monotonic violations, without the need for relabeling.

3.3. Flexible membership aggregation

After estimating the class memberships of every training instance, our algorithm is ready to predict new examples. This last phase has been designed to cover different needs of monotonic scenarios. In addition to the control of noise treatment, greater flexibility has been sought, allowing users to choose between more accurate or pure monotonic predictions.

Algorithm 3 represents in pseudo-code the whole prediction procedure of our proposal MonFkNN. Particularly, the prediction of a new instance x_i is detailed after having previously computed the monotonically-constrained class memberships of the training set as the previous Algorithm 1 is referred in Line 2.

Algorithm 3. MonFkNN: Prediction stage

As shown in Line 6, MonFkNN embeds another MkNN (Algorithm 2) to obtain the neighbors used in the membership aggregation and final prediction. This MkNN also has two versions, *inRange* and *outRange* versions. They are, however, substantially different when compared to original variants.

The *inRange* alternative is based on the same idea of the original MkNN, where the neighbors of an example must belong to a set of monotonically valid classes. However, this range of classes is obtained using the medians acquired from the class memberships of the training instances constrained by monotonicity, as seen in Line 3 and Line 4. This breakthrough improves our method by increasing monotonic noise robustness. Firstly, an *inRange* nearest neighbor rule removes monotonic inconsistencies in the known data-set as previously shown in Algorithm 1. Then, the second MkNN uses this fixed training set (\mathcal{S}, y) to give monotonic predictions as seen in Algorithm 3.

The *outRange* version of our method is completely different from the previous *outRange* rule. It has been designed with the intention of prioritizing to some extent the predictive ability of the classifier over monotonicity. With this purpose in mind, our method considers any example as a valid neighbor regardless of its class label. In contrast to the original model, no filtering or removal of neighbors outside the valid range is performed. However, their relevance in the membership aggregation can be reduced if needed, thanks to a penalty factor introduced in the aggregation expression.

Then, for a new example x , its nearest neighbors are obtained according to the chosen variant. Their memberships are aggregated with the original FkNN formula with the addition of the penalty factor for the *outRange* version. The following expression shows how this parameter is integrated:

$$u(x, l) = \frac{\sum_{j=1}^K u(x_j, l) * \frac{pOR_j}{\|x-x_j\|^{(m-1)}}}{\sum_{j=1}^K \frac{pOR_j}{\|x-x_j\|^{(m-1)}}} \quad (7)$$

As previously, the membership $u(x, l)$ of the new sample x to the class label l is the result of the sum of the class memberships $u(x_j, l)$ of the neighbors x_j inversely weighted with their distance to x . In the *outRange* version of our method, there is another weighting factor in the contribution to the final memberships, the parameter referred to as “penalty of *outRange*” (pOR). The factor pOR_j is applicable only if the class y_j of the neighbor x_j is not in the valid class range r_x of x as exemplified in Lines 7–13. It can be configured with continuous values from 0 to 1. When it is assigned to 1, no penalty is applied. The value 0 means a full penalty, that is, neighbors with invalid classes will not participate in the membership aggregation. For all practical purposes, this last behavior is equivalent to the *outRange* MkNN. We recommend using 0.5 since it is a good balance between reducing their relevance and considering them in the decision.

Finally, the class prediction of the new example x is the median of the resulting normalized class memberships.

As presented, MonFkNN has been developed to be robust to monotonic noise and versatile in many scenarios. The two versions *inRange* and *outRange* with the parameter pOR and the previously mentioned RCr help to tune the algorithm according to the necessities of different kinds of problems.

Among the possibilities that offer these parameters, we have named two configurations with very distinctive behaviors: Pure Monotonic (MonFkNN-PM or PM) and Approximate Monotonic (MonFkNN-AM or AM) Fuzzy k -NN. The Pure Monotonic configuration corresponds to a value of 0.5 for the RCr

parameter and the use of *inRange* rule to obtain the memberships of new instances. This approach aims to give predictions with the minimum violations of monotonicity. In every part of the algorithm, it prioritizes monotonicity over very accurate predictions.

MonFkNN-AM prioritizes the predictive ability and relaxes the monotonic constraints. The memberships of the training set are obtained by the treatment of samples with the same feature values and different classes. Those unique examples will have a membership of 1 to the actual class and 0 for the rest. This behavior is achieved with $RCr = 1$. Then, as we are looking for more accurate predictions, all instances can be considered to be valid neighbors and to contribute to the final aggregation. Those instances with invalid class labels, however, will contribute with only half of their class memberships ($pOR = 0.5$).

Our proposal MonFkNN is available at the GitHub Repository.¹

3.4. Differences between standard FkNN and MonFkNN: theoretical discussion

Standard FkNN and MonFkNN have a similar mathematical formulation. In other words, the expressions used by MonFkNN in the training class membership extraction (Eq. (6)) and in the membership aggregation (Eq. (7)) are the same as those used by FkNN (Eqs. (3) and (2)), for $RCr = 0.51$ and $pOR = 1$. The global behavior of our method is however still completely different to the standard FkNN, due to significant algorithmic differences. Table 1 summarizes the main differences between standard FkNN and our proposal MonFkNN.

Each of the differences mentioned in Table 1 is described and explained below:

- The data-set used to compute the training class memberships is modified before applying the neighborhood rule. The inconsistencies of duplicates are eliminated and reduced to a single instance. The classes of the resultant instances are assigned to the median calculated with the frequency of the appearance of duplicates for each class. This procedure could not even be considered in standard classification, where there is no ordering relationship between classes.
- The neighborhood considered for each training instance is constrained to the monotonicity of the data-set. Then, their resultant class memberships are also monotonically constrained. These adaptations completely modify the neighbors contributing in Eq. (3) and the whole procedure. In addition, the value of 0.51 for RCr is discouraged in MonFkNN in favor of 0.5 due to its contribution to the medians of the samples, above-mentioned in Section 3.2.
- The original FkNN and MonFkNN also share the same membership aggregation, i.e. their expressions (Eqs. (3) and (6)) are the same for *InRange* and *outRange* (with $pOR = 1$) versions of MonFkNN. However, their behavior and their predictions are completely different, due to the differences in the nearest neighbor rule, in the training set and class memberships used in the aggregation procedure. As previously explained, the training class memberships extraction of MonFkNN modifies the training set fixing some monotonic inconsistencies. Duplicates are removed and some training samples might change their classes to preserve the monotonicity of the data-set.
- In MonFkNN, the classes of the training samples determine the monotonically valid classes of the unlabeled instances. Thus, training samples with classes not valid for an instance x will be discarded from the neighborhood (*inRange* version) or penal-

¹ <https://github.com/sergiogvz/MonFkNN>.

Table 1
Summary of algorithmic differences between standard FkNN and MonFkNN.

FkNN	MonFkNN
No special treatment of duplicates.	Duplicates are reduced to a single instance.
Standard nearest neighbor rules.	Monotonic nearest neighbor rules.
Standard training membership extraction.	Monotonically constrained class memberships.
Conservation of original classes in the training class membership extraction.	Loss of influence of original class towards monotonicity with $RcR \leq 0.5$.
Value 0.51 in Eq. (3)	Parameter RcR in Eq. (6)
Standard class membership aggregation.	Monotonically constrained membership aggregation.
No penalty to any neighbors in Eq. (2)	pOR Penalty to out-of-range neighbors in Eq. (7).
Final class as highest membership	Final class as median of class memberships

ized with the parameter pOR (*outRange* version). The configuration *outRange* version with $pOR = 1$ is also discouraged since the final purpose of MonFkNN is to take monotonic constraints into consideration, at least to some extent.

- These mechanics acquire different neighbors to those drawn by FkNN for the same test sample, that is, different class memberships and prediction. Finally, the median as the final class of the class membership vector already implies a significant change in the behavior of the method.

These differences between our proposal and the traditional FkNN are clearly supported by the experiments carried out in Section 5.1.

4. Experimental framework

This section is devoted to introducing the experimental framework used in the different empirical studies of the paper. In our experiments, we have included 12 data-sets of a good variety of problems presenting real monotonic constraints. The data-sets can be seen in Table 2, where the number of instances, attributes and classes are detailed for each data-set in the column Ins., At. and Cl., respectively. The column At. Directions indicates the monotonic direction of the relationship between each attribute and the class: direct (+) or inverse monotony (-). This information is extracted from the description of the problems involving the data-sets. The column Comparable Pairs shows the percentage of pairs of comparable samples over the total number of pairs. Two instances x_i and x_j are comparable if their inputs have an order relation, i.e. $x_i \geq x_j$ or $x_i \leq x_j$. On average, one-third of the total number of pairs of these data-sets are comparable and potential

Table 2
Description of the 12 data-sets used.

Data-set	Ins.	At.	Cl.	At. Directions	Comparable Pairs
<i>artiset</i>	1000	2	10	All direct directions	49.79%
<i>balance</i>	625	4	3	{-, -, +, +}	25.64%
<i>bostonhousing4cl</i>	506	13	4	{-, +, -, +, -, +, -, -, -, +, -}	14.85%
<i>car</i>	1728	6	4	All direct directions	14.36%
<i>ERA</i>	1000	4	9	All direct directions	16.77%
<i>ESL</i>	488	4	9	All direct directions	70.65%
<i>LEV</i>	1000	4	5	All direct directions	24.08%
<i>machineCPU</i>	209	6	4	{-, +, +, +, +, +}	49.53%
<i>qualitative_bankruptcy</i>	250	6	2	All inverse directions	43.77%
<i>SWD</i>	1000	10	4	All direct directions	12.62%
<i>windsorhousing</i>	546	11	2	All direct directions	27.07%
<i>wisconsin</i>	683	9	2	All direct directions	58.04%

violations of monotonicity in the classification process. This quite large amount cannot be neglected.

These data-sets are chosen as the most frequently used in the monotonic classification literature. The classical monotonic set *ERA*, *ESL*, *LEV* and *SWD* [2] are also considered in the study. Additionally, the data-set *artiset* is employed for a comparative study on monotonic noise robustness of MonFkNN (see SubSection 5.4). *Artiset* is an artificial data-set with two attributes (x_1, x_2) and nCl number of classes. For attributes $x_1, x_2 \in [0, 1]$, the class is computed as the truncation of the outcome of the following formula:

$$f(x_1, x_2) = (x_1 + \frac{x_2^2 - x_1^2}{2}) * nCl$$

A 10-fold cross-validation scheme (10-fcv) is carried out to run the different classifiers over these sets. Their partitions have been extracted from the KEEL repository [38].

The classifiers involved in the empirical comparisons are:

- Monotonic k -NN (MkNN) [15]
- Ordinal Stochastic Dominance Learning (OSDL) [31]
- Ordinal Learning Module (OLM) [2]
- Monotonic Multi-Layer Perceptron network (MonMLP) [27]
- C4.5 decision tree for monotonic induction (MID) [3]
- Rank Discrimination Measure Tree (RDMT) [32]
- Partially Monotonic Decision Tree (PMDT) [34]

Table 3 details the parameters chosen according to the recommendations found in the original papers. As a requirement of MkNN, a relabeling technique [16] is applied to training data-sets before fitting MkNN. On the contrary, the rest of the algorithms, including MonFkNN, do not need this relabeling procedure. Therefore, all the results shown for MkNN are obtained with relabeled training sets, while other methods are trained with the original training data-sets.

In order to evaluate the classifiers' proficiency, we have employed three measures of different aspects of their performance: predictive capability, error cost and monotonicity. Standard accuracy is used to evaluate the predictive capability of the models. Mean Absolute Error (MAE) is computed as the average differences of the true instance ranks and the predicted ranks. To evaluate monotonicity, Non-Monotonic Index (NMI) [9] measures the ratio of pairs of samples (NMP) that break monotonicity among the total of pairs, with N being the number of samples in the data-set:

$$NMI = \frac{NMP}{N^2 - N}$$

These measures are computed over a set merged from the test predictions of 10-fcv sets for each data-set and classifier. Finally, the Wilcoxon statistical test, Friedman rank test [20,19] with Holm

Table 3
Parameters considered for the algorithms compared.

Algorithm	Parameters
MkNN [15] OSDL [31]	$k = 5$, distance = euclidean, neighborsType = inRange balanced = No, classificationType = median, lowerBound = 0, upperBound = 1 tuneInterpolationParameter = No, weighted = No, interpolationStepSize = 10, interpolationParameter = 0.5
OLM [2]	modeResolution = conservative modeClassification = conservative
MonMLP [27]	default parameters, hidden1 = 8 iter.max = 1000, monotonic = all att
MID [3] RDMT [32]	$R = 1$, confidence = 0.25, items per leaf = 2 $H =$ Pessimistic rank discrimination measure, measureThreshold = 0, items per leaf = 2
PMDT [34] FkNN [25] MonFkNN	threshold $\theta = 0$, items per leaf = 2 $k = 5$, $K = 9$, distance = euclidean $k = 5$, $K = 9$, distance = euclidean
Pure Monotonic Approximate Monotonic	$RCr = 0.5$, neighborsType = inRange $RCr = 1$, neighborsType = outRange, $pOR = 0.5$

post hoc procedure [24] and Bayesian Sign test [5] are used to validate the results of the empirical comparisons. In the Bayesian Sign test, a distribution of the differences of the results achieved by methods A and B is computed thanks to the Dirichlet Process. This distribution is shown in a graphical space divided into 3 regions: left, rope and right. The location of the majority of distribution in these sectors indicates the final decision of the pairwise Bayesian non-parametric sign test: superiority of algorithm B (left sector), statistical equivalence (rope sector) and superiority of algorithm A (right sector). For the accuracy and MAE results, we have set the inferior and superior limit of the rope region to -0.01 and 0.01 , respectively. However, we have adjusted the limits to -0.0001 and 0.0001 for NMI since NMI values tend to be significantly smaller due to the big difference between the numbers of comparable instance pairs and all possible pairs. The R package rNPBST [10] has been used to extract the graphical representations of the Bayesian Sign tests analyzed in the following empirical studies.

5. Results and analysis

This section presents the results of the empirical studies and their analyses. First, the two configurations of MonFkNN are compared in SubSection 5.1, showing their different strengths. Then, our proposal is compared to methods from the state-of-the-art in terms of prediction capability and monotonicity in SubSection 5.3 and SubSection 5.3, respectively. In SubSection 5.4, the last experiment tests the noise robustness of MonFkNN in contrast to MkNN.

5.1. Evaluation of Monotonic Fuzzy k -NN approaches. Pure Monotonic vs Approximate Monotonic

A comparison between the Pure and Approximate Monotonic version of MonFkNN stresses the different behaviors and aspects of their performance. Additionally, the performance differences between the original FkNN and MonFkNN are analyzed. Table 4 shows the results of FkNN and the two configurations of our proposal MonFkNN in terms of Accuracy, MAE and NMI. Bold-face font indicates the best results obtained for each data-set and metric.

In Table 4, the differences between both approaches (PM and AM) can be seen clearly. Just as they were designed, MonFkNN-AM has better accuracy on average, while MonFkNN-PM achieves monotonically reliable predictions. Both have good, stable results in terms of MAE, with AM coming out slightly on top.

AM configuration obtains the most accurate predictions for more than 50% of the benchmark used. On the other hand, the PM model achieves better results according to monotonicity in 10 of the 12 data-sets used, with large differences in *Windsorhousing* and *MachineCPU* problems. When compared with FkNN, MonFkNN greatly improves the performance of the original algorithm. Both versions of MonFkNN (PM and AM) are better on average for each of the three different measures. Particularly, there is an overwhelmingly large difference between FkNN and MonFkNN-PM in terms of monotonicity. FkNN is better only for 3 data-sets when taking just accuracy and MAE into consideration. However, it does not outperform the monotonic predictions of MonFkNN.

This improvement is also reflected in the Wilcoxon statistical test applied to the results achieved using these methods. Table 5 presents the hypothesis of equivalence of the Wilcoxon test for $\alpha = 0.1$ on the pairwise comparison of FkNN (1) and our two proposals (MonFkNN-PM (2) and MonFkNN-AM (3)). As shown in Table 5, MonFkNN-AM is statistically better than FkNN in terms of accuracy and MAE with p -Values under 0.1. Considering monotonicity, MonFkNN-PM and -AM statistically outperform FkNN with very low p -Values. Overall, MonFkNN is clearly superior to FkNN in scenarios with monotonic constraints.

The reasons for these differences in results are clear and mainly due to their algorithmic differences. MonFkNN has learning procedures with notions in the order relation of classes and the monotonic constraints between input and output, which explain an overall better performance in terms of MAE and NMI. Additionally, MonFkNN has a greater awareness and treatment of noisy data, which helps obtain better accuracy.

Since monotonicity is usually prioritized in classification with monotonic constraints, we will use MonFkNN-PM in the following empirical studies.

5.2. Comparison with the state-of-the-art: prediction capabilities

Here we evaluate the performance of our approach in comparison to methods from the state-of-the-art of monotonic classification. In this comparison, we look for a balance between accurate and monotonic predictions. Therefore, we compare the results obtained in terms of the selected metrics independently. Then, we draw our conclusions and check if our approach behaves well in the different aspects of classification with monotonic constraints.

First, we evaluate the prediction capability of our method. Table 6 gathers the accuracy results for the different data-sets obtained by the tested algorithms. With these outcomes, MonFkNN-PM performs overwhelmingly better than the rest in terms of accuracy. Our approach achieves the most accurate predictions on average with a wide margin. Additionally, it obtains the best results for 5 data-sets, with particularly remarkable cases, such as *balance*. PMDT is the second best method in terms of accuracy and it is the only method that come close to the performance of MonFkNN-PM. However, it obtains the overall best results for one data-set only (*bostonhousing*).

As mentioned before, we have used the Friedman rank test and the Bayesian Sign test to corroborate the significance of the differences of our approach and the selected methods. Table 7 includes the outcome of the Friedman rank and Holm tests in relation to the obtained Accuracy results. MonFkNN-PM is ranked first with a high ranking value compared to others. All the hypotheses of equivalence are rejected with small p -values with the exception of PMDT, which would be rejected for $\alpha = 0.1$. The distance between the ranks of MonFkNN-PM and PMDT is still quite large.

Fig. 1 graphically represents the difference between MonFkNN-PM and other methods and its statistical significance in terms of

Table 4
Results for the Pure and Approximate Monotonic Fuzzy *k*-NN.

	Accuracy			MAE			NMI		
	FkNN	MonFkNN-PM	MonFkNN-AM	FkNN	MonFkNN-PM	MonFkNN-AM	FkNN	MonFkNN-PM	MonFkNN-AM
<i>artiset</i>	0.9339	0.9309	0.9349	0.0661	0.0691	0.0651	0.0000	0.0000	0.0000
<i>balance</i>	0.8896	0.9307	0.9008	0.1424	0.0853	0.1168	0.0000	0.0000	0.0001
<i>bostonhousing4cl</i>	0.7174	0.6561	0.7134	0.3241	0.3972	0.3261	0.0004	0.0000	0.0001
<i>car</i>	0.9311	0.9740	0.9834	0.0793	0.0295	0.0195	0.0002	0.0000	0.0000
<i>ERA</i>	0.1730	0.2420	0.2430	1.6660	1.2813	1.2993	0.0141	0.0052	0.0052
<i>ESL</i>	0.6783	0.7036	0.7131	0.3484	0.3149	0.3053	0.0014	0.0004	0.0003
<i>LEV</i>	0.6020	0.6377	0.6110	0.4330	0.3927	0.4223	0.0021	0.0004	0.0009
<i>machineCPU</i>	0.6699	0.7033	0.6699	0.3589	0.3158	0.3493	0.0058	0.0002	0.0017
<i>qualitative_bankruptcy</i>	0.9960	0.9960	0.9960	0.0040	0.0040	0.0040	0.0000	0.0000	0.0000
<i>SWD</i>	0.5350	0.5807	0.5833	0.5180	0.4370	0.4380	0.0027	0.0007	0.0003
<i>windsorhousing</i>	0.7857	0.7576	0.7839	0.2143	0.2424	0.2161	0.0062	0.0005	0.0051
<i>wisconsin</i>	0.9678	0.9653	0.9663	0.0322	0.0347	0.0337	0.0000	0.0000	0.0000
<i>Avg:</i>	0.7400	0.7565	0.7583	0.3489	0.3003	0.2996	0.0027	0.0006	0.0012

Table 5
Wilcoxon test applied to the results obtained by Fuzzy *k*-NN algorithms: FkNN (1), MonFkNN-PM (2) and MonFkNN-AM (3)

Comparison	R ⁺	R ⁻	Hypothesis ($\alpha = 0.1$)	p-Value
<i>Accuracy:</i>				
(2) vs. (1)	49.0	17.0	Not Rejected	0.1748
(3) vs. (1)	61.5	16.5	Rejected	0.0847
<i>MAE:</i>				
(2) vs. (1)	51.0	15.0	Not Rejected	0.1230
(3) vs. (1)	57.0	9.0	Rejected	0.0322
<i>NMI:</i>				
(2) vs. (1)	76.5	1.50	Rejected	0.0012
(3) vs. (1)	72.5	5.50	Rejected	0.0059

Table 6
Results in terms of Accuracy achieved by the tested algorithms.

	MonFkNN-PM	MkNN	OSDL	OLM	MonMLP	MID	RDMT	PMDT
<i>artiset</i>	0.9309	0.9199	0.1952	0.7948	0.9463	0.7237	0.8749	0.8539
<i>balance</i>	0.9307	0.8624	0.6352	0.8320	0.9131	0.7808	0.7216	0.7792
<i>bostonhousing4cl</i>	0.6561	0.6126	0.2787	0.5277	0.3979	0.6739	0.6304	0.6739
<i>car</i>	0.9740	0.9711	0.9549	0.9543	0.8474	0.8027	0.7297	0.9682
<i>ERA</i>	0.2420	0.1990	0.2320	0.1690	0.2380	0.2760	0.2390	0.2430
<i>ESL</i>	0.7036	0.6332	0.6721	0.5738	0.7234	0.6414	0.5635	0.6598
<i>LEV</i>	0.6377	0.4630	0.6400	0.4250	0.6167	0.6070	0.5210	0.6370
<i>machineCPU</i>	0.7033	0.6890	0.2919	0.6746	0.6730	0.6220	0.6555	0.6507
<i>qualitative_bankruptcy</i>	0.9960	0.9960	0.9160	0.9800	0.6427	0.9840	0.9840	0.9920
<i>SWD</i>	0.5807	0.5200	0.5840	0.4160	0.5063	0.5540	0.5180	0.5830
<i>windsorhousing</i>	0.7576	0.5861	0.4927	0.7564	0.7790	0.8205	0.8022	0.7564
<i>wisconsin</i>	0.9653	0.9649	0.9590	0.8873	0.8604	0.9517	0.9502	0.9561
<i>Avg:</i>	0.7565	0.7014	0.5710	0.6659	0.6787	0.7031	0.6825	0.7294

Table 7
Holm test applied to the Accuracy results among the tested algorithms.

Control Method: MonFkNN-PM (2.04)				
i	Algorithm (Rank)	Z	p-Value	Hypothesis ($\alpha = 0.05$)
7	OLM (6.13)	4.083	0.00004	Rejected
6	OSDL (5.42)	3.375	0.00073	Rejected
5	RDMT (5.38)	3.333	0.00085	Rejected
4	MonMLP (4.67)	2.625	0.00866	Rejected
3	MID (4.42)	2.375	0.01754	Rejected
2	MkNN (4.21)	2.167	0.03026	Rejected
1	PMDT (3.75)	1.708	0.08757	Not Rejected

accuracy. In order to save space and avoid plotting 7 heat-maps for each metric, we have only included PMDT, as it is the best and most recent algorithm among the monotonic decision trees [34]. As mentioned before, the position of the majority of the distribution in these maps determines the decision of the test: the right

sector means the statistical superiority of MonFkNN-PM over the compared method, the rope sector is the statistical equivalency and the left side indicates the superiority of the other algorithm.

These heat-maps clearly indicate the significant superiority of MonFkNN-PM over all methods except PMDT as the computed dis-

tributions are always located in the right region. The most significant outcome is the comparison with OLM (Fig. 1c), even though it does not obtain the worst results. For *MkNN* (Fig. 1a) and *OSDL* (Fig. 1b), there are a few cases where their performances are statistically equivalent to *MonFkNN-PM*. On the contrary, *MonMLP* is significantly more accurate in a few data-sets, although the *MonFkNN-PM* is clearly superior (Fig. 1d). Considering the comparison with *PMDT* (Fig. 1e), the majority of the distribution is located in the statistical equivalence. However, it is still shifted to the right with a large number of points, indicating a better performance for *MonFkNN-PM*. Almost none support the performance of *PMDT*.

Error costs could be essential for monotonic ranking problems. Table 8 shows the error in the form of MAE made by the evaluated classifiers. As was the case in accuracy performance, *MonFkNN-PM* clearly performs better than the rest, with the smallest error on average and for 4 of the data-sets. It also achieves similar results in problems where other algorithms come out on top, such as *LEV* or *wisconsin*.

Table 9 shows the ranking of the methods and *p*-values obtained with the post hoc test for the MAE comparison. As in the accuracy tests, our proposal is once again ranked as the best method with a solid statistical significance as compared to almost all algorithms. *PMDT* still achieves similar results to *MonFkNN-PM* with a *p*-value that does not reject the hypothesis for $\alpha = 0.05$, but does for $\alpha = 0.1$. In this case, the *p*-value of *PMDT* is smaller and its rank difference with our proposal is larger than that obtained in terms of accuracy.

Fig. 2 shows the Bayesian Sign test on pairwise comparison with our method according to MAE. As shown by the distributions in the right part of the majority of the figures, *MonFkNN-PM* is definitely better when considering error costs. This is more statistically significant as compared to *OLM* (Fig. 2c), where nearly the entire distribution is in the right region. *MonFkNN*, *MkNN* and *OSDL* share

some good results, but these last two are not statistically better than the former in any circumstance as seen in Fig. 2a and Fig. 2b. As we have also seen in the accuracy comparison, Fig. 2d points out the statistical superiority of *MonFkNN-PM* over *MonMLP*, but the latter has a better MAE in some cases. Given Fig. 2e, *MonFkNN-PM* and *PMDT* can be considered to be statistically the same in terms of error costs. However, *MonFkNN-PM* performs better statistically than *PMDT* in an important part of the benchmark, as a fragment of the distribution is located on the right side and almost none are found on the left.

5.3. Comparison with the state-of-the-art: monotonicity

Now we will analyze the performance according to the monotonicity of our proposal compared to methods chosen from the state-of-the-art. Table 10 shows the NMI results achieved by the selected models. In this case, the competition is close. Monotonic decision trees (*MID*, *RDMT*, and *PMDT*) clearly obtain less monotonic predictions. *MID* has the worst behavior considering only monotonicity and *PMDT* is the most monotonic decision tree classifier. *OLM* and *MonMLP* are slightly better than *PMDT*, but they still do not come close to the best methods. *MonFkNN-PM*, *MkNN*, and *OSDL* perform similarly. *MonFkNN-PM* and *OSDL* are slightly better on average. It is worth mentioning the existence of simpler data-sets, such as *artiset* and *wisconsin*, in relation to monotonicity as almost every algorithm accomplishes the same good results. The best results for the more complex sets are shared by the different methods.

Table 11 summarizes the comparison according to monotonicity with the Friedman statistical test results. In this case, *MonFkNN-PM* is barely selected as the control method. For half of the benchmark (*OSDL*, *MkNN*, *MonMLP* and *OLM*), the hypotheses of equivalence are not rejected for $\alpha = 0.05$. On the contrary, all

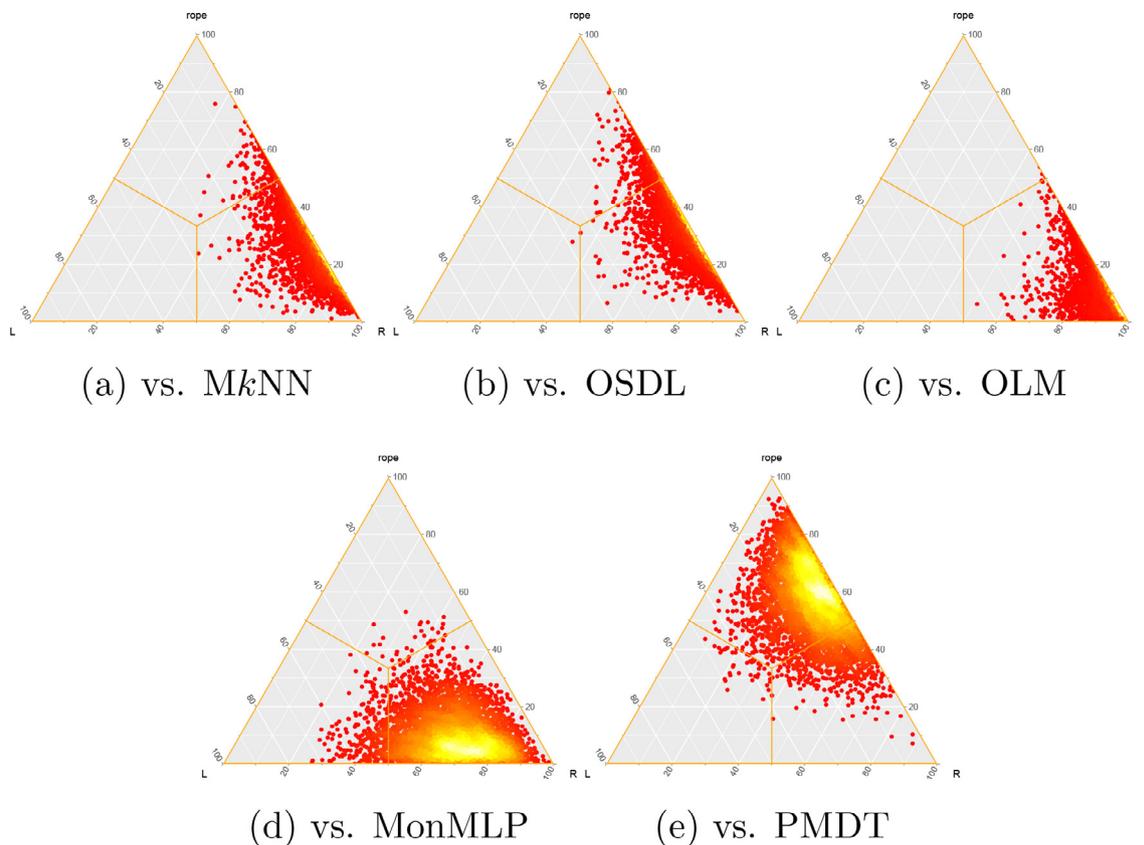


Fig. 1. Bayesian Sign Test heat-map for *MonFkNN-PM* vs. the rest in terms of accuracy.

Table 8
Results in terms of MAE achieved by the tested algorithms.

	MonFkNN-PM	MkNN	OSDL	OLM	MonMLP	MID	RDMT	PMDT
<i>artiset</i>	0.0691	0.0771	1.6897	0.2082	0.0537	0.3123	0.1251	0.1471
<i>balance</i>	0.0853	0.1504	0.4912	0.1920	0.0992	0.3360	0.3840	0.2560
<i>bostonhousing4cl</i>	0.3972	0.4901	0.9368	0.5988	0.7655	0.3893	0.4249	0.3676
<i>car</i>	0.0295	0.0359	0.0475	0.0538	0.1599	0.2506	0.3079	0.0365
<i>ERA</i>	1.2813	1.4270	1.2850	2.1500	1.2317	1.2970	1.3060	1.2870
<i>ESL</i>	0.3149	0.3791	0.3607	0.4734	0.2910	0.3934	0.4918	0.3750
<i>LEV</i>	0.3927	0.5740	0.3920	0.6680	0.4170	0.4290	0.5430	0.3940
<i>machineCPU</i>	0.3158	0.3301	0.9043	0.3589	0.3413	0.4211	0.3589	0.3732
<i>qualitative_bankruptcy</i>	0.0040	0.0040	0.0840	0.0200	0.3573	0.0160	0.0160	0.0080
<i>SWD</i>	0.4370	0.4840	0.4370	0.7630	0.5167	0.4750	0.4990	0.4340
<i>windsorhousing</i>	0.2424	0.4304	0.5073	0.2436	0.2210	0.1795	0.1978	0.2436
<i>wisconsin</i>	0.0347	0.0337	0.0410	0.1127	0.1396	0.0483	0.0498	0.0439
<i>Avg:</i>	0.3003	0.3680	0.5980	0.4869	0.3828	0.3790	0.3920	0.3305

Table 9
Holm test applied to the MAE results among the tested algorithms

Control Method: MonFkNN-PM (2.00)				
i	Algorithm (Rank)	Z	p-Value	Hypothesis ($\alpha = 0.05$)
7	OLM (6.17)	4.167	0.00003	Rejected
6	RDMT (5.54)	3.542	0.00040	Rejected
5	OSDL (5.29)	3.292	0.00099	Rejected
4	MID (4.96)	2.958	0.00309	Rejected
3	MonMLP (4.25)	2.250	0.02445	Rejected
2	MkNN (4.04)	2.042	0.04119	Rejected
1	PMDT (3.75)	1.750	0.08011	Not Rejected

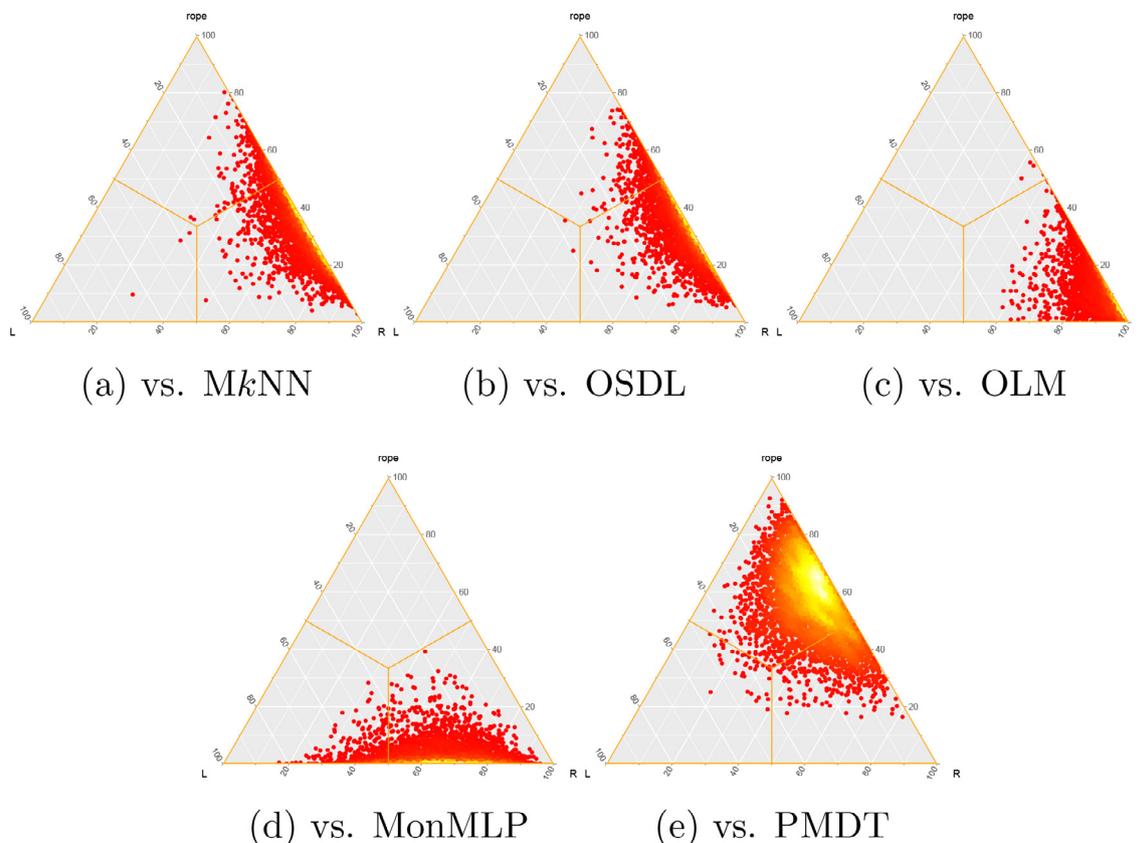


Fig. 2. Bayesian Sign Test heat-map for MonFkNN-PM vs. the rest in terms of MAE.

Table 10
Results in terms of NMI achieved by the tested algorithms.

	MonFkNN-PM	MkNN	OSDL	OLM	MonMLP	MID	RDMT	PMDT
<i>artiset</i>	0.0000	0.0000	0.0000	0.0000	0.0000	0.0039	0.0000	0.0001
<i>balance</i>	0.0000	0.0001	0.0006	0.0000	0.0000	0.0017	0.0029	0.0010
<i>bostonhousing4cl</i>	0.0000	0.0000	0.0000	0.0003	0.0007	0.0022	0.0010	0.0010
<i>car</i>	0.0000	0.0000	0.0000	0.0000	0.0001	0.0046	0.0002	0.0000
<i>ERA</i>	0.0052	0.0056	0.0049	0.0063	0.0026	0.0082	0.0085	0.0058
<i>ESL</i>	0.0004	0.0012	0.0006	0.0025	0.0003	0.0021	0.0066	0.0032
<i>LEV</i>	0.0004	0.0010	0.0004	0.0043	0.0008	0.0018	0.0086	0.0006
<i>machineCPU</i>	0.0002	0.0000	0.0000	0.0014	0.0001	0.0037	0.0047	0.0028
<i>qualitative_bankruptcy</i>	0.0000	0.0000	0.0003	0.0000	0.0079	0.0002	0.0000	0.0000
<i>SWD</i>	0.0007	0.0005	0.0009	0.0015	0.0004	0.0020	0.0000	0.0010
<i>windsorhousing</i>	0.0005	0.0000	0.0000	0.0000	0.0000	0.0030	0.0002	0.0059
<i>wisconsin</i>	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000
<i>Avg:</i>	0.0006	0.0007	0.0006	0.0014	0.0011	0.0028	0.0027	0.0018

Table 11
Holm test applied to the NMI results among the tested algorithms.

Control Method: MonFkNN-PM (2.9583)				
i	Algorithm (Rank)	Z	p-Value	Hypothesis ($\alpha = 0.05$)
7	MID (7.00)	4.042	0.00005	Rejected
6	RDMT (6.33)	3.375	0.00074	Rejected
5	PMDT (5.75)	2.792	0.00524	Rejected
4	OLM (4.13)	1.167	0.24335	Not Rejected
3	MonMLP (3.63)	0.667	0.50499	Not Rejected
2	MkNN (3.13)	0.167	0.86763	Not Rejected
1	OSDL (3.08)	0.125	0.90052	Not Rejected

monotonic decision trees are statistically worse than MonFkNN-PM by a wide margin. The best monotonic decision tree (PMDT) does not reach good performance in terms of monotonicity of the best algorithms. This is probably due to the greedy construction of monotonic constraints into the tree.

In Fig. 3, the statistical comparisons of the NMI results are represented with Bayesian Sign Test heat-maps. These plots show similar conclusions extracted from the previous table with NMI results. MonFkNN is significantly superior to PMDT (Fig. 3e). In Fig. 3c, the right-shifted distribution points out that MonFkNN-PM is better than OLM. Although they share a part of the distribution in the rope section, OLM has too few individuals in its left section (Fig. 3c). When compared with MkNN (Fig. 3a), OSDL (Fig. 3b) and MonMLP (Fig. 3c), big parts of the distributions are located in all the decision sectors. Even though their distributions are slightly shifted to the right (Fig. 3a and Fig. 3b), the core of the distributions are found in the rope. Then, we can roughly assume statistical equivalence.

In summary, MonFkNN-PM obtains significantly better results in terms of accuracy and error cost than almost all of the considered methods. Our approach also achieves the most monotonic predictions alongside OSDL. MonFkNN-PM is slightly and non-statistically better than PMDT in terms of accuracy and error costs, but the former overwhelmingly outperforms PMDT considering monotonicity. Therefore, MonFkNN-PM is an overall better method.

The main reason behind the remarkable performance of MonFkNN is its capability of not sacrificing any objective of monotonic classification. Usually, some classifiers, such as OSDL, sacrifice accurate predictions in order to accomplish monotonic models. The results of OSDL for *artiset* and *bostonhousing* and the outcome of MkNN for *balance* are good examples of this statement. On the other hand, other methods, such as monotonic decision trees and particularly PMDT, achieve accurate predictions but break the monotonic constraints in their predictions more frequently. However, the MonFkNN procedure of training class membership extraction is designed to mitigate the influence of non-

monotonic noisy data, without the need to aggressively modify the training data as done by relabeling in MkNN. The MonFkNN prediction stage offers the flexibility of choice for most accurate or monotonic predictions. Additionally, MonFkNN includes technologies that are more appropriate for ordinal and monotonic classification, such as median as a final class.

5.4. On the robustness of Monotonic Fuzzy k-NN to monotonic noise

With this last empirical study, we aim to test the robustness of MonFkNN-PM to the presence of monotonic violations or noise in the training sets as compared to MkNN. Thus, we have introduced different amounts of noisy instances in the training partitions of the artificial data-set *Artiset*. Then, the performance of MonFkNN-PM and MkNN is measured and compared in terms of accuracy, MAE and NMI while the noise ratio increases.

In order to increase the impact of class noise, we have randomly under-sampled every training set to 25% of their instances. Then, a subset of randomly selected instances is converted to noise by changing their class labels. This label modification is done according to the adjacent classes of the implicated instance. Specifically, a large number of neighbors are computed for the future noisy example x_i . 15 nearest neighbors were the value used in this experiment. Next, the neighbors with the same class as x_i are removed and a new class is randomly obtained in relation to the presence ratio of other classes in its filtered neighbors. This ensures a certain degree of proximity between the changed sample and its new class.

This process is executed following the same cross-validation scheme mentioned earlier. Since the noise generation has a random component, the experiment was repeated three times with different seeds, averaging the obtained results. After the noise generation and before the execution of MkNN, a relabeling technique [16] was applied to the resultant data-sets.

Fig. 4 shows the impact of increasing noise on the number of monotonic violations in *Artiset* training sets. This effect is measured by the Non-Monotonic Index (NMI) over the resulting train-

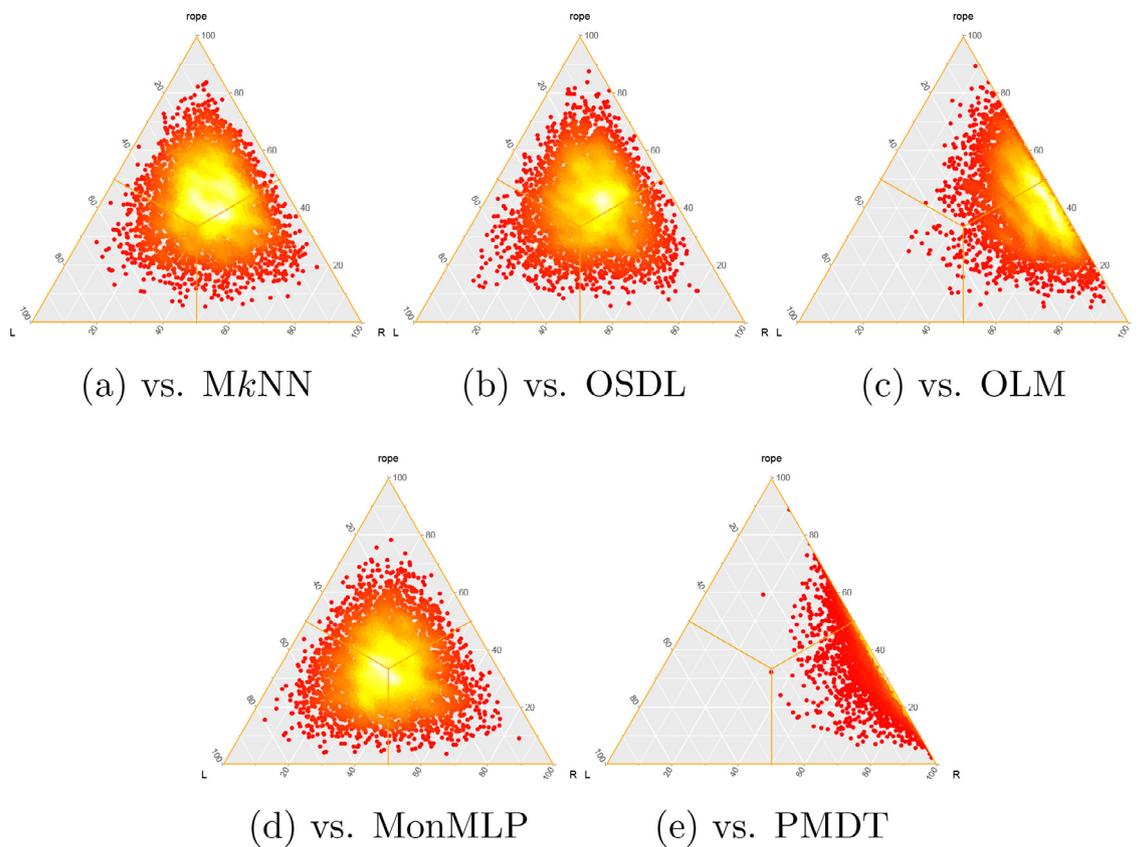


Fig. 3. Bayesian Sign Test heat-map for MonFkNN-PM vs. the rest in terms of NMI.

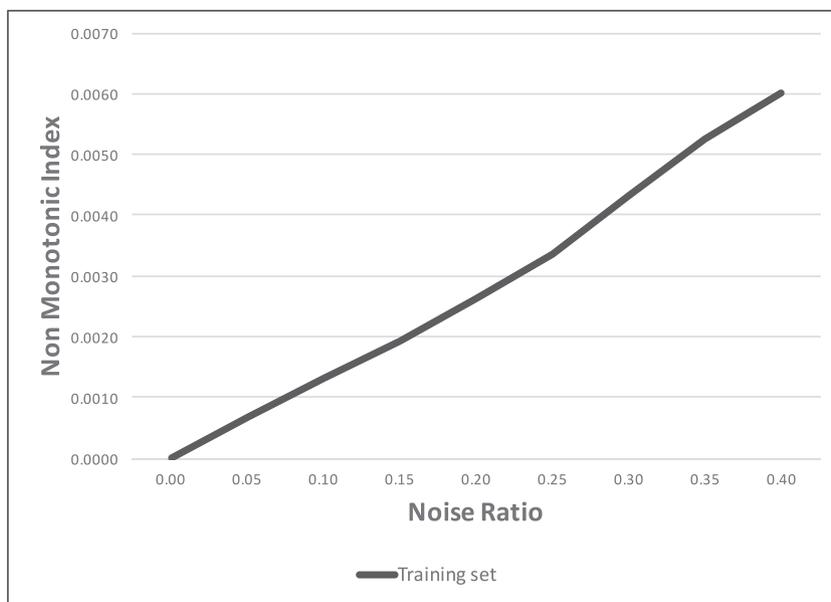


Fig. 4. Impact of the addition of class noise in Artiset on monotonic violations measured by NMI.

ing samples. As previously mentioned, class noise significantly aggravates the monotonicity of the data-sets. The increase in NMI is directly proportional to the increase in noise as clearly shown in Fig. 4.

Fig. 5 shows the performance of MonFkNN-PM and *Mk*NN (darker and lighter lines, respectively) on the basis of precision (5a), MAE (5b) and NMI (5c), with the progression of noise. As expected,

while the amount of noise grows, the performance of both methods get worse, that is, their accuracy decreases and errors and non-monotonic predictions increases. However, there are some big differences between classifiers.

Firstly, the behavior of MonFkNN-PM facing noise is clearly better than that of *Mk*NN in every tested aspect. The black lines are always located above the lighter ones in Fig. 5a, which indicates

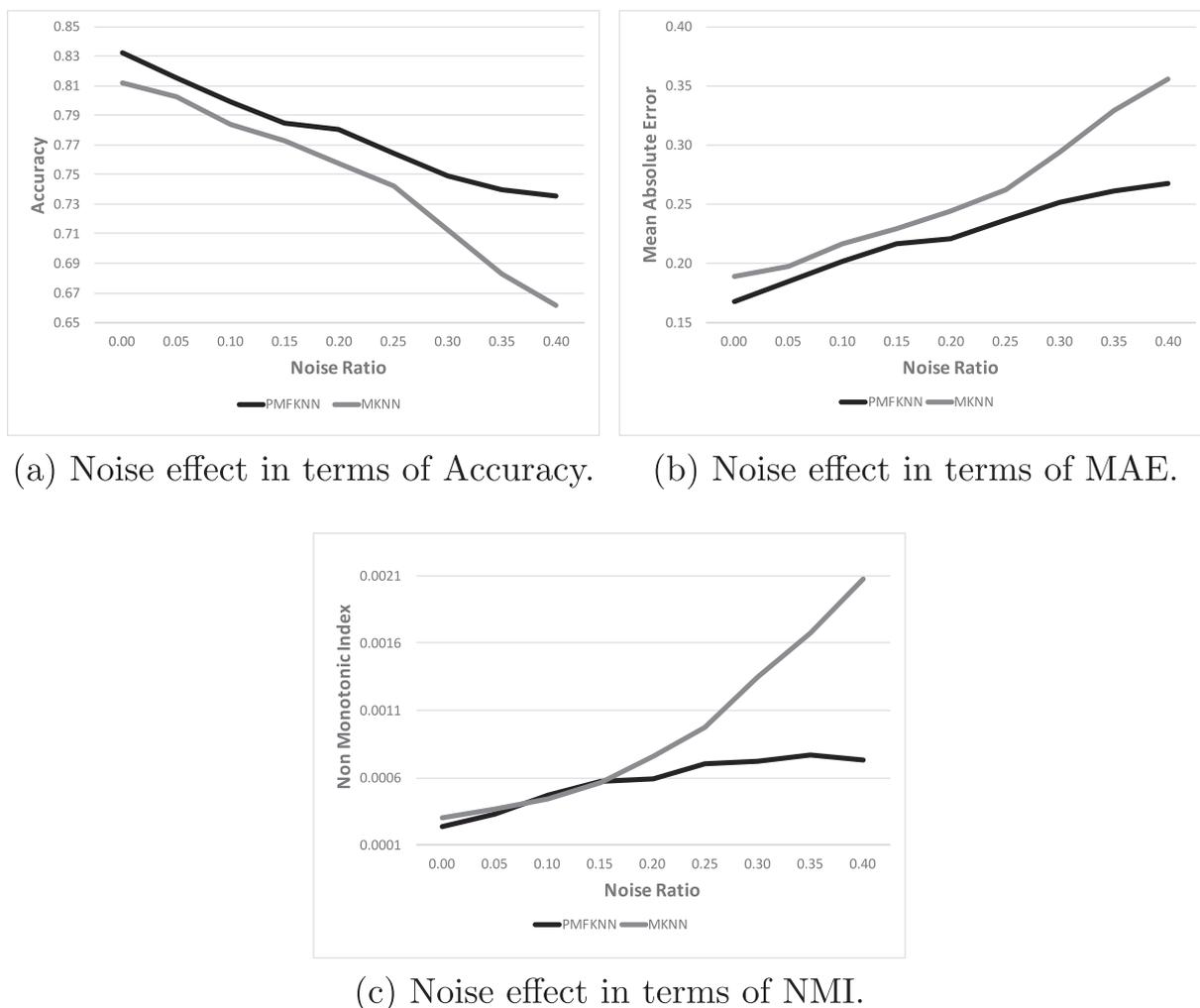


Fig. 5. Comparison of MonFkNN-PM and MkNN performance on Artiset data-set with the different amounts of noisy samples.

greater accuracy, and under them in Figs. 5b and 5c, meaning better MAE and NMI for MonFkNN-PM. Usually, the distance between both methods is large, with the exception of the NMI results obtained for the smallest values of noise. In addition, while the noise ratio increases, their differences also increase.

The slope of deterioration of MonFkNN-PM performance remains stable, even being reduced in some cases, while the MkNN slope becomes steeper as the amount of noise increases. This last event can be clearly seen when the noise ratio reaches the 25% of the instances, where the decline of MkNN is magnified, especially in terms of monotonicity (Fig. 5c). On the other hand, the NMI results of MonFkNN-PM seem to increase at a slower rate by that point. This exhibits the great robustness of MonFkNN-PM to monotonic violations.

Next, the behavior of both methods in relation to noise are analyzed using a graphical example. Fig. 6 is a graphical representation of the predictions and classification boundaries inferred by MkNN and MonFkNN-PM for Artiset with 35% noise. Fig. 6a represents the perfect class surfaces defined by Artiset generation expression (see Section 4) and the training samples. In Fig. 6a, black points represent the noise artificially introduced into the data-set. In Figs. 6b and 6c, the black examples are wrongly classified instances, while the right predictions are colored in white.

The first clear difference between the MkNN and MonFkNN-PM performances shown in Figs. 6b and 6c is the amount of black dots. MonFkNN has far fewer classification mistakes than MkNN. Addi-

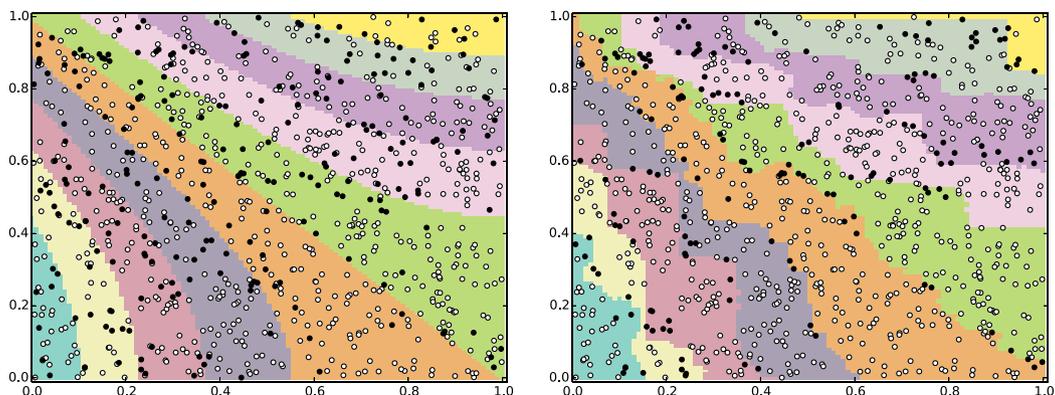
tionally, MonFkNN-PM is better at conserving the right regions for the classes, while MkNN can lose nearly all the entire sections of some of them. The regions in lighter and brighter yellow are shrunk by MkNN in favor of their adjacent classes.

With these experiments, MonFkNN has shown strong robustness to monotonic noise preserving the decision boundaries as precisely as possible, and hence, has performed well in terms of precision, error costs and monotonicity. This robustness is the result of all the procedures included in MonFkNN, but it may also be mainly due to the reduction of the impact of non-monotonic noise during the extraction of the class memberships of the training instances.

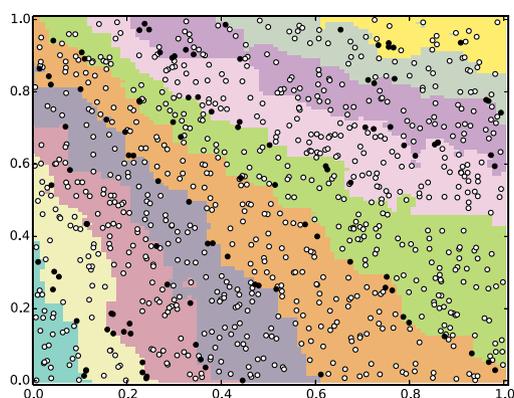
6. Conclusion

In this paper, we proposed a Fuzzy k -Nearest Neighbors model for classification with monotonic constraints. The final class label obtained from membership functions has been revised to respect these constraints. MonFkNN has been designed with different mechanisms to reduce the influence of monotonic violations. As a demonstration of its flexibility, two different model configurations with different behaviors have been presented.

Over the course of the experimental analyses, the great potential of both proposed versions, namely Pure and Approximate Monotonic Fuzzy k -NN, has been shown in relation to monotonic-



(a) Exact decision surface for Artiset. (b) Decision surfaces inferred by $MkNN$. Black points represent noise.



(c) Decision surfaces inferred by $MonFkNN-PM$.

Fig. 6. Classification boundaries inferred by $MkNN$ and $MonFkNN-PM$ from the plotted Artiset with 35% noisy instances. Black points represent the instances wrongly classified by the decision surfaces shown.

ity and accuracy, respectively. Compared to other methods, $MonFkNN$ is significantly better in terms of accuracy and error cost, matching the best NMI results. In addition, it has shown its robustness to large amounts of noise while preserving its good performance.

Future proposals should be robust to monotonic noise in order to obtain accurate and monotonic predictions. $MonFkNN$, as an example, opens possibilities to other fuzzy approaches since they are also potentially reliable against noise. Additionally, fuzzy techniques may be useful when defining different levels of constraints between input and output attributes. That is, some attributes may be more important than others regarding monotonicity. This problem representation may be very useful for monotonic classifiers.

CRediT authorship contribution statement

Sergio González: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Visualization. **Salvador García:** Conceptualization, Writing - review & editing, Visualization, Supervision. **Sheng-Tun Li:** Writing - review & editing. **Robert John:** Writing - review & editing. **Francisco Herrera:** Writing - review & editing, Project administration, Funding acquisition.

Declaration of 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.

Acknowledgements

This work was supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2017-89517-P and a research scholarship (FPU) given to Sergio González by the Spanish Ministry of Education, Culture and Sports. The third author was supported in part by MOST, Taiwan, R.O.C., under MOST 105-2410-H-006-038-MY3. We wish that this work serves as a tribute to our friend Bob for his life dedicated to academics and research, our collaboration and friendship, we miss you.

References

- [1] J. Alcalá-Fdez, R. Alcalá, S. González, Y. Nojima, S. García, Evolutionary fuzzy rule-based methods for monotonic classification, *IEEE Trans. Fuzzy Syst.* 25 (2017) 1376–1390.

- [2] A. Ben-David, Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications, *Decis. Sci.* 23 (1992) 1357–1372.
- [3] A. Ben-David, Monotonicity maintenance in information-theoretic machine learning algorithms, *Mach. Learn.* 19 (1995) 29–43.
- [4] A. Ben-David, L. Sterling, T. Tran, Adding monotonicity to learning algorithms may impair their accuracy, *Expert Syst. Appl.* 36 (2009) 6627–6634.
- [5] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, *J. Mach. Learn. Res.* 18 (2017) 2653–2688.
- [6] N. Biswas, S. Chakraborty, S.S. Mullick, S. Das, A parameter independent fuzzy weighted k-nearest neighbor classifier, *Pattern Recogn. Lett.* 101 (2018) 80–87.
- [7] J.R. Cano, N.R. Aljohani, R.A. Abbasi, J.S. Alowidbi, S. García, Prototype selection to improve monotonic nearest neighbor, *Eng. Appl. Artif. Intell.* 60 (2017) 128–135.
- [8] J.R. Cano, S. García, Training set selection for monotonic ordinal classification, *Data Knowl. Eng.* 112 (2017) 94–105.
- [9] J.R. Cano, P.A. Gutiérrez, B. Krawczyk, M. Woźniak, S. García, Monotonic classification: an overview on algorithms, performance measures and data sets, *Neurocomputing* 341 (2019) 168–182.
- [10] J. Carrasco, S. García, M. del Mar Rueda, F. Herrera, rNPBST: An R package covering non-parametric and bayesian statistical tests, in: *International Conference on Hybrid Artificial Intelligence Systems*, Springer, pp. 281–292..
- [11] D. Charte, F. Charte, S. García, F. Herrera, A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations, *Prog. Artif. Intell.* (2019) 1–14.
- [12] C.C. Chen, S.T. Li, Credit rating with a monotonicity-constrained support vector machine model, *Expert Syst. Appl.* 41 (2014) 7235–7247.
- [13] K. Dembczyński, W. Kotłowski, R. Słowiński, Learning rule ensembles for ordinal classification with monotonicity constraints, *Fund. Informaticae* 94 (2009) 163–178.
- [14] J. Derrac, S. García, F. Herrera, Fuzzy nearest neighbor algorithms: taxonomy, experimental analysis and prospects, *Inf. Sci.* 260 (2014) 98–119.
- [15] W. Duivesteyn, A. Feelders, Nearest neighbour classification with monotonicity constraints, in: *ECML/PKDD (1)*, pp. 301–316..
- [16] A. Feelders, Monotone relabeling in ordinal classification, in: *ICDM, IEEE Computer Society*, 2010, pp. 803–808..
- [17] F. Fernández-Navarro, A. Riccardi, S. Carloni, Ordinal neural networks without iterative tuning, *IEEE Trans. Neural Network Learn. Syst.* 25 (2014) 2075–2085.
- [18] J. García, A.M. AlBar, N.R. Aljohani, J.R. Cano, S. García, Hyperrectangles selection for monotonic classification by using evolutionary algorithms, *Int. J. Comput. Intell. Syst.* 9 (2016) 184–202.
- [19] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, *Inf. Sci.* 180 (2010) 2044–2064.
- [20] S. García, F. Herrera, An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons, *J. Mach. Learn. Res.* 9 (2008) 2677–2694.
- [21] S. García, J. Luengo, F. Herrera, *Data Preprocessing in Data Mining*, Springer, 2015.
- [22] S. González, S. García, S.T. Li, F. Herrera, Chain based sampling for monotonic imbalanced classification, *Inf. Sci.* 474 (2019) 187–204.
- [23] S. González, F. Herrera, S. García, Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity, *New Gener. Comput.* 33 (2015) 367–388.
- [24] S. Holm, A simple sequentially rejective multiple test procedure, *Scand. J. Stat.* (1979) 65–70.
- [25] J.M. Keller, M.R. Gray, J.A. Givens, A fuzzy k-nearest neighbor algorithm, *IEEE Trans. Syst. Man Cybern.* (1985) 580–585.
- [26] W. Kotłowski, R. Słowiński, On nonparametric ordinal classification with monotonicity constraints, *IEEE Trans. Knowl. Data Eng.* 25 (2013) 2576–2589.
- [27] B. Lang, Monotonic multi-layer perceptron networks as universal approximators, in: *International Conference on Artificial Neural Networks*, Springer, pp. 31–37..
- [28] H. Levy, *Stochastic Dominance: Investment Decision Making under Uncertainty*, Springer, 2015.
- [29] S.T. Li, C.C. Chen, A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge, *IEEE Trans. Fuzzy Syst.* 23 (2015) 1713–1727.
- [30] S. Lievens, B. De Baets, Supervised ranking in the weka environment, *Inf. Sci.* 180 (2010) 4763–4771.
- [31] S. Lievens, B. De Baets, K. Cao-Van, A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting, *Ann. Oper. Res.* 163 (2008) 115–142.
- [32] C. Marsala, D. Petturiti, Rank discrimination measures for enforcing monotonicity in decision tree induction, *Inf. Sci.* 291 (2015) 143–171.
- [33] W. Pan, Q. Hu, Y. Song, D. Yu, Feature selection for monotonic classification via maximizing monotonic dependency, *Int. J. Comput. Intell. Syst.* 7 (2014) 543–555.
- [34] S. Pei, Q. Hu, Partially monotonic decision trees, *Inf. Sci.* 424 (2018) 104–117.
- [35] R. Potharst, A. Ben-David, M.C. van Wezel, Two algorithms for generating structured and unstructured monotone ordinal data sets, *Eng. Appl. Artif. Intell.* 22 (2009) 491–496.
- [36] Y. Qian, H. Xu, J. Liang, B. Liu, J. Wang, Fusing monotonic decision trees, *IEEE Trans. Knowl. Data Eng.* 27 (2015) 2717–2728.
- [37] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, F. Gomide, Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey, *Inf. Sci.* (2019).
- [38] I. Triguero, S. González, J.M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M.J. del Jesús, L. Sánchez, F. Herrera, Keel 3.0: an open source software for multi-stage analysis in data mining, *Int. J. Comput. Intell. Syst.* 10 (2017) 1238–1249.
- [39] L.A. Zadeh, Fuzzy sets, in: *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*, World Scientific, 1996, pp. 394–432..
- [40] H. Zhu, E.C. Tsang, X.Z. Wang, R.A.R. Ashfaq, Monotonic classification extreme learning machine, *Neurocomputing* 225 (2017) 205–213.



Sergio González received his M.Sc. in Computer Science in 2015 and Ph.D. in Computer Science in 2020, both from the University of Granada, Spain. He has published nine papers in international journals and five contributions to national and international congresses. His research interests include data mining, data preprocessing, imbalanced learning, monotonic classification, big data, and deep learning. Dr. González has collaborated with several members of the scientific community in different research projects, including a fellowship with the National Cheng Kung University, Tainan, Taiwan, thanks to the 2017 Summer Program in Taiwan of the Ministry of Science and Technology, the development of KEEL machine learning software, and the application of data science to industrial problems of the company Repsol.



Salvador García is currently a Professor in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. Dr. García has published more than 100 papers in international journals (more than 70 in Q1), with more than 13,000 citations, h-index 43, over 70 papers in international conference proceedings (data from Web of Science). He has been associated with the international program committees and organizing committees of several regular international conferences including IEEE CEC, ICPR, ICDM, IJCAI, etc. As edited activities, he has co-edited two special issues in international journals and he is an associate editor of “Information Fusion” (Elsevier), “Swarm and Evolutionary Computation” (Elsevier) and “AI Communications” (IOS Press) 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, metaheuristics and biometrics. Dr. García has been given some awards and honors for his personal work or for his publications in and conferences, such as IFSA-EUSFLAT 2015 Best Application Paper Award and IDEAL 2015 Best Paper Award. He belongs to the list of the Highly Cited Researchers in the area of Computer Sciences (2014–2020): <http://highlycited.com/> (Clarivate Analytics). His h-index is 54 in Scholar Google, receiving more than 22,000 citations, till date.



Sheng-Tun Li received his PhD degree in computer science from University of Houston, University Park, Texas, USA, in 1995. He is currently a distinguished professor in the Department of Industrial and Information Management & Institute of Information Management at National Cheng Kung University, Taiwan. Dr. Li is author/co-author of ten IT-related textbooks including two translated, over 70 journal papers, and numerous conference papers. His work has been appeared in IEEE Transactions on Fuzzy Systems, IEEE Transactions on Systems, Man, and Cybernetics-Part B, Information Sciences, Fuzzy Sets and Systems, Omega The International Journal of Management Science, Knowledge-Based Systems, Journal of Information Science, Technovation, etc. His research interests include artificial intelligence, machine learning, business intelligence, data mining, and text mining.



Professor John has a first class degree in Mathematics, an MSc Statistics and a PhD in type-2 fuzzy logic. His initial career was as a mathematician in various roles for industry and commerce as an AI consultant. He joined De Montfort University in 1989 and the University of Nottingham in 2013. Bob is currently Professor of Operational Research and Computer Science and Head of the Optimisation and Learning Lab (OPAL) in the School of Computer Science at the University of Nottingham. He is a Senior Member of IEEE, Fellow of the British Computer Society and elected member of the EPSRC college. In the field of type-2 fuzzy logic, his work is widely recognised by the international fuzzy logic community as leading in the aspects of theoretical foundations, as well as practical applications. His work has produced many fundamental new results that have opened the field to new research, enabling a broadening of scope and application. Bob has had significant funding for his research from a variety of sources and he has over 8500 citations on google scholar with an h-index of 39.



Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. 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 in Data Science and Computational Intelligence (DaSCI). He's an academician in the Royal Academy of Engineering (Spain). He has been the supervisor of 51 Ph.D. students. He has published more than 500 journal papers, receiving more than 89000 citations (Scholar Google, H-index 150). He has been nominated as a Highly Cited Researcher (in the fields of Computer Science and Engineering, respectively, 2014 to present, Clarivate Analytics). He currently acts as Editor in Chief of the international journal "Infor-

mation Fusion" (Elsevier). He acts as editorial member of a dozen of journals. He received the several honors and awards, among others: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 and 2012 Papers, 2011 Lotfi A. Zadeh Prize Best paper Award (IFSA Association), 2013 AEPIA Award to a scientific career in Artificial Intelligence, 2014 XV Andalucía Research Prize Maimónides, 2017 Andalucía Medal (by the regional government of Andalucía), 2018 "Granada: Science and Innovation City" award.

His current research interests include among others, Computational Intelligence (including fuzzy modeling, computing with words, evolutionary algorithms and deep learning), information fusion and decision making, and data science (including data preprocessing, prediction, non-standard classification problems, and big data).