Fuzzy Monotonic *K*-Nearest Neighbor versus Monotonic Fuzzy *K*-Nearest Neighbor

Hong Zhu, *Student Member*, *IEEE*, Xizhao Wang, *Fellow*, *IEEE*, and Ran Wang, *Member*, *IEEE*

***Abstract*—In real-life applications, monotonic classification is a widespread task, where the improvement of a particular input value cannot result in an inferior output. A common drawback of the existing algorithms for monotonic classification is their sensitivity to noise data which particularly refer to monotonicity violations in the monotonic circumstance. Motivated by weakening the impact of noises, the Fuzzy Monotonic *K*-Nearest Neighbor (FM*K*NN) is proposed in this paper, which constructs monotonic classifiers by taking advantage of the fuzzy dominance relation between a pair of instances, especially that between incomparable instances for the first time. Through tuning the thresholds of fuzzy dominance relation degrees, FM*K*NN intends to decrease the disturbance caused by noises which considerably affect the selection range of the *K*-Nearest Neighbors in different extent. The experimental results show that the best average improvement degrees of FM*K*NN in terms of the *K*NN-based and non-*K*NN-based classifiers on all the involved datasets arrive at 28%, 11% and 29% with respect to ACCU, MAE and NMI, respectively, which demonstrates the superiority of our proposed FM*K*NN over other state-of-the-art monotonic classifiers including the Monotonic Fuzzy *K*-Nearest Neighbor (MF*K*NN) which disperses the impact of noise data by converting crisp class labels into class membership vectors.**

***Index Terms*—monotonic classification, *K*-nearest neighbor, robustness improvement, incomparable instances, fuzzy dominance relation.**

I. Introduction

Monotonic classification [1] is a special classification task where there exist monotonicity constraints between the class label and some features, it means that both the class label and the feature values are ordinal, and the class label should not decrease with the increase of the feature values when the rest remain the same. Formally, monotonic classification is to predict the class label $f(\mathbf{x})$ of a feature vector $\mathbf{x}$, such that $\mathbf{x} \succeq \mathbf{x}' \Rightarrow f(\mathbf{x}) \geq f(\mathbf{x}')$, where " $\succeq$ " and "$\geq$ " represent the dominance relationship among feature vectors and class labels, respectively. For example, a car with better main performances and lower price is more likely to win higher acceptance from consumers. Monotonic classifications are important tasks because they come up frequently in real-word application scenarios, such as bankruptcy risk assessment [2], housing price setting [3], credit rating [4] and lecture evaluation [5].

Generally, conventional algorithms cannot solve monotonic classification problems, because they do not take the monotonicity constraints into account, and as a result, they cannot maintain the monotonicity of prediction results. For example, the Shannon's information entropy based decision tree induction is an efficient and effective model for common classification. However, it cannot learn monotonic classification rules even given a monotonic dataset, because the Shannon's information entropy cannot reflect the ordinal structure in monotonic classification [6]. In recent years, more and more researchers have devoted to the study of monotonic classification problems, which results in various of approaches emerging. Generally, these approaches can be divided into two categories. The first category is to build monotonic predictors by taking monotonicity constraints into the framework of conventional classification or regression algorithms, such as instance-based classifiers [7], decision trees [8]-[15], ensemble learning [16]-[19], neural networks [20]-[24], and support vector machines [25], [26]. The second category is to monotonize the datasets using some preprocessing techniques when there is a small quantity of instances violating monotonicity constraints. The most commonly used preprocessing operations include relabeling [27]-[29], feature selection [30], [31], instance selection [32], [33] and training set selection [34], [35].

Among all the methods, instance-based monotonic predictors have a significant superiority when the approximated target function is very complex but can still be described by a collection of less complex local approximations. Instead of estimating the target function once for the entire instance space, they complete the approximation task locally and differently for each new instance to be predicted. So far,

This work was supported in part by the National Natural Science Foundation of China (Grants 61976141, 62176160, 61772344, and 61732011), in part by the Basic Research Project of Knowledge Innovation Program in ShenZhen (JCYJ20180305125850156), in part by the Natural Science Foundation of Shenzhen (University Stability Support Program no. 20200804193857002), and in part by the Interdisciplinary Innovation Team of Shenzhen University. (*Corresponding authors: Xizhao Wang and Ran Wang*.)

Hong Zhu was with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China. She is now with the School of Artificial Intelligence, Shenzhen Polytechnic, Shenzhen 518055, China (e-mail: zhuhong@szu.edu.cn).

Xizhao Wang is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China (e-mail: xizhaowang@ieee.org).

Ran Wang is with the College of Mathematics and Statistics, Shenzhen University, Shenzhen 518060, China, with the Shenzhen Key Laboratory of Advanced Machine Learning and Applications, Shenzhen University, Shenzhen 518060, China, and also with the Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China (e-mail: wangran@szu.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org

several instance-based approaches to generating monotonic predictors have appeared. The earliest achievement is the Ordered Learning Model (OLM) [33] presented by Ben-David, which constructs a rule base composed of instances satisfying monotonicity constraints. When predicting a new instance, OLM does not emphasize the use of nearby training instances but just simply takes the maximum class label of all the instances that are inferior to the predicted instance, which magnifies the impact of data far away from the predicted instance. Another instance-based monotonic classifier is the Ordered Stochastic Dominance Learner (OSDL) [7] proposed by Cao-Van, which is a probabilistic classifier interpreting the monotonicity constraints in terms of stochastic dominance. However, only when the monotonic classification problems are binary, can OSDL maintain the monotonicity of predictions.

In order to overcome the drawbacks of the above methods, Duivesteijn and Feelders proposed the Monotonic *K*-Nearest Neighbor (M*K*NN) [36], which is an instance-based monotonic predictor based on the classical *K*-Nearest Neighbor (*K*NN) [37]. M*K*NN provides a strategy for monotonic prediction by only making use of the nearby instances, which can not only decrease the bad influence of very distant instances, but also improve the operating efficiency. Besides, M*K*NN can approximate both discrete-valued and real-valued target functions, thus it is suitable for both classification and regression with monotonicity constraints. Furthermore, it is easy to see that the work mechanism of M*K*NN is quite simple.

Although M*K*NN has the above mentioned advantages, it is extremely susceptible to monotonicity violations. Before using M*K*NN, preprocessing measure must be taken [38]. Beyond this, a modification of M*K*NN, called Monotonic Fuzzy *K*-Nearest Neighbor (MF*K*NN) [39], was designed from the angle of algorithm improvement based on the Fuzzy *K*-Nearest Neighbor (F*K*NN) [40]. It increases the robustness of M*K*NN against noises by introducing the calculation method of class fuzzy memberships, without the need for data preprocessing.

In monotonic classification problems discussed in this article, the noise refers to the label noise which means the incorrect observed label. We assume the label noise is noisy completely at random (NCAR), that is the occurrence of a labelling error is independent from the vector of features and the true class itself [41]. The label noises lead to violation of monotonicity between features and the label.

The main contribution of this paper is that we propose a new modification of M*K*NN, named Fuzzy Monotonic *K*-Nearest Neighbor (FM*K*NN), which is able to handle noises without the need of data preprocessing. Different from the existing fuzziness based monotonic classifiers [42]-[43], such as MF*K*NN in which only the class membership information is used, FM*K*NN uses the fuzzy dominance relations [44] between instances, especially between the incomparable instances, for the first time. In order to highlight the advantage of our work, we make a comprehensive comparison between FM*K*NN and MF*K*NN theoretically and experimentally. The similarities and differences between the two methods are listed, as well as their respective characteristics. By conducting contrast experiments with both *K*NN-based and non-*K*NN-based monotonic classifiers, we validate that our proposed FM*K*NN has significant advantage over MF*K*NN when addressing monotonic classification with much noises and incomparable instance pairs.

The rest of this paper is organized as follows. In Section II, we provide the background knowledge about monotonic classification and fuzzy dominance relation. In Section III, we introduce two existing *K*NN-based monotonic classifiers M*K*NN and MF*K*NN briefly, which give rise to our work in this article. In Section IV, we propose our FM*K*NN model in detail. In Section V, we discuss the similarities and differences between FM*K*NN and MF*K*NN theoretically. In Section VI, we present the experimental framework used in different empirical studies. In Section VII, we conduct contrast experiments and analyze the results. Finally, in Section VIII, we summarize the contributions and conclude this paper.

Throughout this paper, we use *K*NN for *K*-Nearest Neighbor, M*K*NN for Monotonic *K*-Nearest Neighbor, F*K*NN for Fuzzy *K*-Nearest Neighbor, MF*K*NN for Monotonic Fuzzy *K*-Nearest Neighbor, FM*K*NN for Fuzzy Monotonic *K*-Nearest Neighbor, '$\geq$' and '$\leq$' for the ordinal relation between two variables, '$\succeq$' and '$\preceq$' for the dominance relation between two instances.

II. Background knowledge

In this section, we introduce the background knowledge involved in this paper.

A. Monotonic Classification

Here we use a series of definitions to explain the knowledge about monotonic classification problems.

Let $U = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be the universe of instances, where $\mathbf{x}_i = [x_i^1, x_i^2, \cdots, x_i^n]$ is the feature vector of the *i*-th instance, $y_i \in \{l_1, l_2, \cdots, l_d\}$ is the target decision value, *N* is the total number of training instances, *n* is the number of features, and *d* is the number of decision values.

Compared with general classification problems, monotonic classification problems have two unique characteristics.

1) ***Both features and the decision variable are fully ordered****:* the values of the features and the decision variable are either fully ordered symbolic or numeric. We use '$\geq$' and '$\leq$' to indicate the ordinal relationship among feature values and decision values. For example, $\mathbf{x}_i^k \geq \mathbf{x}_j^k$ represents the *k*-th feature value of the *i*-th instance is not inferior to that of the *j*-th one, where $i,\ j \in \{1, 2, \cdots, N\}, k \in \{1, 2, \cdots, n\}$. Particularly, in the Employee Selection problem, for the Education Background feature, the doctoral degree is superior to the master's degree, which can be expressed as "doctoral degree $\geq$ master's degree".

Definition 1: If $\mathbf{x}_i^k \geq \mathbf{x}_j^k$ holds well for any $k \in \{1, 2, \cdots, n\}$, then we say $\mathbf{x}_i$ dominates $\mathbf{x}_j$, denoted by $\mathbf{x}_i \succeq \mathbf{x}_j$ or $\mathbf{x}_j \preceq \mathbf{x}_i$.

Definition 2: For instances $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$, if there exists a dominance relationship between their feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e., $\mathbf{x}_i \succeq \mathbf{x}_j$ or $\mathbf{x}_i \preceq \mathbf{x}_j$, then we say $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ are comparable; otherwise, they are incomparable.

The dominance relation between two instances is partially ordered and denoted by '$\succeq$' and '$\preceq$'. In mathematics, a

partially ordered set (also poset) consists of a set together with a binary relation called a "partial order". The word "partial" is used as an indication that not every pair of elements needs to be comparable. In real datasets, it is common that not any two instances are comparable.

*2) **There are monotonicity constraints between features and the decision variable**:* that is to say, an instance with better performance on all the features cannot be assigned to an inferior decision value, which can be expressed as: if $\mathbf{x}_i \succeq \mathbf{x}_j$, then $y_i \geq y_j$; if $\mathbf{x}_i \preceq \mathbf{x}_j$, then $y_i \leq y_j$.

Definition 3: For any two instances $(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j) \in U$, if $\mathbf{x}_i^k \geq \mathbf{x}_j^k \rightarrow y_i \geq y_j$ holds under the premise that the two instances perform the same on all the remaining features, then we say the decision variable is monotonic with respect to the *k*-th feature.

Definition 4: If any two instances in dataset U satisfy the monotonicity constraints, then we say U is monotonic.

Definition 5: A classifier is monotonic if $\mathbf{x}_i \succeq \mathbf{x}_j \rightarrow \hat{y}_i \geq \hat{y}_j$ holds well for any two instances $(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j) \in U$, where $\hat{y}_i$ and $\hat{y}_j$ are respectively the prediction results of the two instances generated by the classifier.

In order to address the monotonic classification, we need to generate a classifier which aims to approximate a monotonic target function as accurately as possible by making a trade-off between the prediction accuracy and the maintenance degree of monotonicity.

B. Fuzzy Dominance Relations

As explained above in Section II-A, it is not always true that two instances are comparable. Two instances are incomparable when there is no dominance relationship between their feature vectors. In the theory of fuzzy mathematics, the dominance relation between incomparable instances can be measured.

Fuzzy dominance relation not only can represent whether an instance dominates another, but also can measure how much the former is superior to the latter, especially for the incomparable instances. Here we put forward a method of calculating the degree of fuzzy dominance relation between two instances. The following are relevant definitions and descriptions in detail.

Definition 6: A fuzzy dominance relation ***R*** is a fuzzy set on the product set $U \times U$, which is described by a membership function $\mu_R : U \times U \rightarrow [0, 1]$. If the cardinality of U is finite, the fuzzy dominance relation can also be represented by an $N \times N$ matrix $(r_{ij})_{N\times N}$, where r_{ij} is interpreted as the dominance degree of instance $(\mathbf{x}_i, y_i)$ over $(\mathbf{x}_j, y_j)$.

The fuzzy dominance degree of $(\mathbf{x}_i, y_i)$ over $(\mathbf{x}_j, y_j)$ with respect to the *k*-th feature is marked as r_{ij}^k and can be computed by the following equation [44]:

$$r_{ij}^k = \frac{1}{1 + e^{-a(\mathbf{x}_i^k - \mathbf{x}_j^k)}}, \tag{1}$$

where a is a positive parameter. According to the characteristics of (1) which is a Logsig transfer function essentially, we can know that when $\mathbf{x}_i^k \leq \mathbf{x}_j^k$, $r_{ij}^k \in [0,\ 0.5]$; otherwise, $r_{ij}^k \in [0.5,\ 1]$.

Next, to get the overall fuzzy dominance degree of $(\mathbf{x}_i, y_i)$ over $(\mathbf{x}_j, y_j)$, a weighted aggregation operation on each r_{ij}^k is worked out, where the correlation coefficients are adopted as the weights which can reflect the degree of statistical monotonicity between features and the decision variable. The aggregation equation is shown as:

$$r_{ij} = \sum_{k=1}^{n} Corr(feature_k, d) * r_{ij}^k, \tag{2}$$

where $Corr(feature_k, d)$ is the correlation coefficient between the *k*-th feature $feature_k$ and the decision variable d, r_{ij} is the overall fuzzy dominance degree of $(\mathbf{x}_i, y_i)$ over $(\mathbf{x}_j, y_j)$.

Generally, the overall fuzzy dominance degree between two instances takes a value in the range [0,1], so r_{ij} needs to be normalized to the range [0,1] after obtained by (2).

In particular, after normalization, $r_{ij} < \frac{1}{2}$ represents that $(\mathbf{x}_i, y_i)$ is more likely to be inferior to $(\mathbf{x}_j, y_j)$; $r_{ij} = \frac{1}{2}$ shows that $(\mathbf{x}_i, y_i)$ equals to $(\mathbf{x}_j, y_j)$; $\frac{1}{2} < r_{ij} < 1$ means that $(\mathbf{x}_i, y_i)$ is more likely to be superior to $(\mathbf{x}_j, y_j)$; $r_{ij} = 1$ indicates that $(\mathbf{x}_i, y_i)$ dominates $(\mathbf{x}_j, y_j)$ absolutely. Usually, $r_{ij} + r_{ji} = 1$ holds well for every $i, j \in \{1, 2, \cdots, N\}$.

III. Monotonic classifiers based on *K*NN

In this section, we particularly introduce the existing monotonic classifiers based on *K*NN, which lead to our research directly in this article.

A. Monotonic K-Nearest Neighbors (MKNN)

The classical *K*NN can approximate both discrete-valued and real-valued target functions, and correspondingly for an unseen instance, the estimation result is the most common class label or the mean numeric decision value of its *K*-nearest neighbors.

For an unseen instance, after fixing a distance matric, *K*NN selects the first *K* instances which are closest to it as its *K*-nearest neighbors. Fig. 1 shows the selection strategy intuitively, where the plane geometries stand for the instances described with only two features. In particular, the triangle represents the instance to be predicted, besides, the circles and squares denote instances from two different categories. Based on the principle of closest distance, for the triangle, when $K = 1$, Circle A is selected as its 1-nearest neighbor, and when $K = 5$, Circles A, B, C, and Squares D, E are selected as its 5-nearest neighbors.

When dealing with monotonic classification problems, the conventional *K*NN will lose its efficiency, because in its whole prediction process, the monotonicity constraints are not concerned. To overcome this problem, the M*K*NN was designed.

The difference between *K*NN and M*K*NN lies in the strategy of selecting the nearest neighbors. In M*K*NN, all the instances are assumed to satisfy the monotonicity constraints. We denote the instance to be predicted as $(\mathbf{x}_0, y_0)$, and use $\hat{y}_0$ to stand for the predictive decision value. According to the monotonicity constraints, $\hat{y}_0$ is supposed to lie in the interval $[y_{\min}, y_{\max}]$, where $y_{\min}$ is the superlative class label of all the training instances which are inferior to $(\mathbf{x}_0, y_0)$ on each feature, and conversely $y_{\max}$ is the lowest class label of all the training

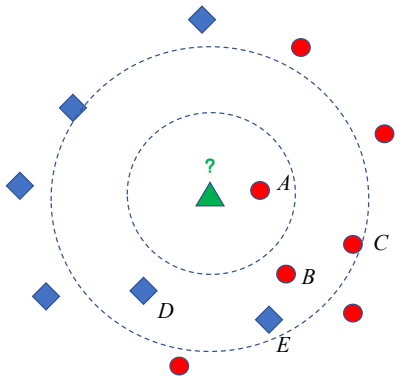

Fig. 1. The selection range of K nearest neighbors in KNN.

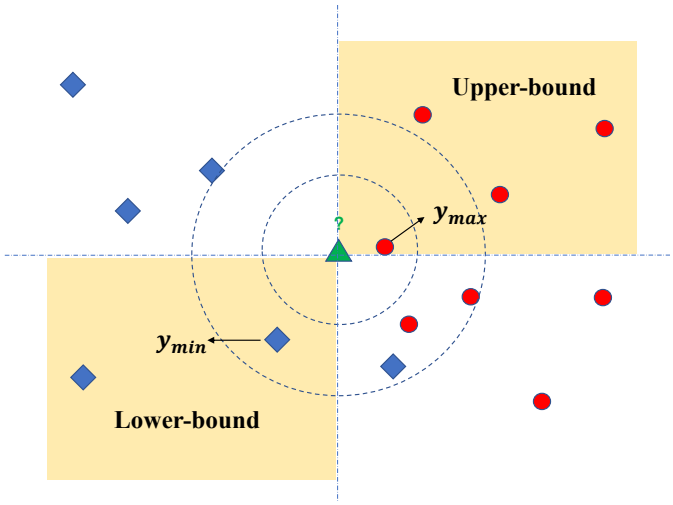

Fig. 2. The lower-bound and upper-bound of a predicted instance in MKNN.

instances which are superior to $(\mathbf{x}_0, y_0)$ on each feature. The K-nearest neighbors of the predicted instance are selected from the training instances whose class labels lie in $[y_{\min}, y_{\max}]$. In comparison with the traditional KNN, MKNN is more likely to guarantee the predictive results of the unseen instances to satisfy the monotonicity constraints by setting the ranges for selecting the K-nearest neighbors.

Fig. 2 visually shows the key points of MKNN. The lower-bound and upper-bound highlighted here are respectively the potential regions where points that determine $y_{\min}$ and $y_{\max}$ lie. The $y_{\min}$ and $y_{\max}$ can be mathematically expressed as follows.

$$y_{\min} = \max\{y|(\mathbf{x}, y) \in U \wedge \mathbf{x} \preceq \mathbf{x}_0\}, \tag{3}$$

$$y_{\max} = \min\{y|(\mathbf{x}, y) \in U \wedge \mathbf{x}_0 \preceq \mathbf{x}\}, \tag{4}$$

where U is a monotonic dataset.

After selecting the K-nearest neighbors, KNN and MKNN adopt the same mechanism for prediction. In both models, if the decision variable is symbolic, the prediction is made by using the majority voting strategy which means that the most common class label of the K-nearest neighbors is taken as the final predictive class label; otherwise, if the decision variable is numeric, the prediction is the mean target decision values of the K-nearest neighbors.

Although MKNN can maintain monotonicity to some extent, its ability of resistance to noises is rather weak. The reason is that if the class labels of the instances which determine the interval $[y_{\min}, y_{\max}]$ are polluted by noises, the selection range of the predicted instance will be fixed incorrectly, which leads to a failed classification. The training data must be fully monotonic before the adoption of MKNN.

B. Monotonic Fuzzy K-Nearest Neighbors (MFKNN)

In order to ameliorate the robustness of MKNN against noises, a modified model called MFKNN [39] has been proposed on the basis of FKNN [40]. Essentially, MFKNN is a fusion of MKNN and FKNN. On the one hand, it obtains the strong robustness by following the technique of FKNN for fuzzifying the crisp class label of each instance; on the other hand, it guarantees the monotonicity of the classification results to some extent through adopting the strategy of MKNN for extracting nearest neighbors. MFKNN solves monotonic classification problems polluted by noises without the need for data preprocessing. The process of MFKNN roughly consists of the following three steps.

Step 1. Fuzzify the crisp class label of each training instance.

In MFKNN, firstly the crisp class label of each instance needs to be fuzzified by adopting the mechanism mentioned in FKNN, which can disperse the influence of a noisy datum to its nearest neighbors. The fuzzification operation is shown in detail as follows.

Case 1. For the instances whose feature values are repeated but their class labels are different, the class membership degrees are computed as the frequencies of each category in the entire training set.

$$u((\mathbf{x}_i, y_i), l_j) = \frac{|\{(\mathbf{x}, y) \in U|\mathbf{x} = \mathbf{x}_i \wedge y = l_j\}|}{|\{(\mathbf{x}, y) \in U|\mathbf{x} = \mathbf{x}_i\}|}, \tag{5}$$

where $u((\mathbf{x}_i, y_i), l_j)$ represents the membership degree of the i-th instance with respect to the j-th category, $i = 1, 2, \cdots N$, $j = 1, 2, \cdots, C$.

Case 2. For the instances apart from those in *Case 1*, their class membership degrees are calculated based on the class distribution of the K-nearest neighbors which are extracted by using the strategy mentioned in MKNN.

$$u((\mathbf{x}_i, y_i), l_j) = \begin{cases} RCr + (nn_{l_j}/K) * (1 - RCr), \text{ if } y_i = l_j; \\ (nn_{l_j}/K) * (1 - RCr), \text{ if } y_i \neq l_j, \end{cases} \tag{6}$$

where K is the number of nearest neighbors, nn_{l_j} is the number of nearest neighbors belonging to category l_j, RCr is a parameter called "Real Class relevance" which takes values from [0,1]. If there are no neighbors labeled with y_i around instance $(\mathbf{x}_i, y_i)$, then RCr is considered as the minimum membership of $(\mathbf{x}_i, y_i)$ to category y_i.

Step 2. Calculate the class memberships of the predicted instance.

For a predicted instance $(\mathbf{x}_0, y_0)$, MFKNN extracts its K-nearest neighbors in the same manner as MKNN. Then the

memberships of $(\mathbf{x}_0, y_0)$ to each class is computed by the following equation [43]:

$$u((\mathbf{x}_0, y_0), l_j) = \frac{\sum_{r=1}^{k} u((\mathbf{x}_r, y_r), l_k) * \frac{1}{\|\mathbf{x}_0 - \mathbf{x}_r\|^{q-1}}}{\sum_{r=1}^{k} \frac{1}{\|\mathbf{x}_0 - \mathbf{x}_r\|^{q-1}}}, \tag{7}$$

where $(\mathbf{x}_r, y_r)$ are the K-nearest neighbors of instance $(\mathbf{x}_0, y_0)$, $r = 1, 2, \cdots, K$, q is a parameter which controls the distances between $(\mathbf{x}_0, y_0)$ and its K-nearest neighbors.

Step 3. Obtain the predictive result by converting the class memberships into a crisp class label.

In FKNN which is used to solve traditional classification problems, the crisp class label of the predicted instance is determined by the maximum membership principle, which means that the class label with the maximum membership is considered as the predictive result, however, this rule is not suitable for monotonic classifications.

In MFKNN, the crisp class label of the predicted instance is calculated as the mean value of l_{MIN} and l_{MAX}, which has been proved in [45], where

$$MIN = \min\left\{ m \in \{1, 2, \cdots, C\} | \sum_{i=1}^{m} u((\mathbf{x}_0, y_0), l_i) \geq \frac{1}{2} \right\}, \tag{8}$$

$$MAX = \max\left\{ m \in \{1, 2, \cdots, C\} | \sum_{i=m}^{C} u((\mathbf{x}_0, y_0), l_i) \geq \frac{1}{2} \right\}. \tag{9}$$

IV. Fuzzy Monotonic K-Nearest Neighbors

In this section, we propose another modification of MKNN called Fuzzy Monotonic K-Nearest Neighbors (FMKNN) which aims to enhance the resistance ability of MKNN to noises without the need of data preprocessing. In comparison with the original MKNN, the extraction range of nearest neighbors in FMKNN is softened by applying the fuzzy dominance degree to describing the dominance relation between any two instances.

A. Method of Extracting K-Nearest Neighbors in FMKNN

Similar to the original MKNN, FMKNN also limits the decision value of a predicted instance $(\mathbf{x}_0, y_0)$ to an interval $[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}]$, but unlike M$KNN, FMK$NN defines this interval on the basis of fuzzy dominance relations between instances rather than the absolute dominance relations. $y_{\min}^{\delta_1}$ and $y_{\max}^{\delta_2}$ are shown as follows.

$$y_{\min}^{\delta_1} = \max\left\{ y | (\mathbf{x}, y) \in U \wedge \mathbf{x}_0 \succeq_{\delta_1} \mathbf{x} \right\}, \tag{10}$$

$$y_{\max}^{\delta_2} = \min\left\{ y | (\mathbf{x}, y) \in U \wedge \mathbf{x} \succeq_{\delta_2} \mathbf{x}_0 \right\}, \tag{11}$$

where U is a monotonic dataset, $\delta_1 \in \left(\frac{1}{2}, 1\right)$ and $\delta_2 \in \left(\frac{1}{2}, 1\right)$ are two thresholds which control the fuzzy dominance degree between $\mathbf{x}_0$ and a training instance $\mathbf{x}$. $\mathbf{x}_0 \succeq_{\delta_1} \mathbf{x}$ represents that $\mathbf{x}_0$ fuzzily dominates $\mathbf{x}$ in the degree ranging from $[\delta_1, 1]$, and $\mathbf{x} \succeq_{\delta_2} \mathbf{x}_0$ denotes that $\mathbf{x}$ fuzzily dominates $\mathbf{x}_0$ in the degree ranging from $[\delta_2, 1]$.

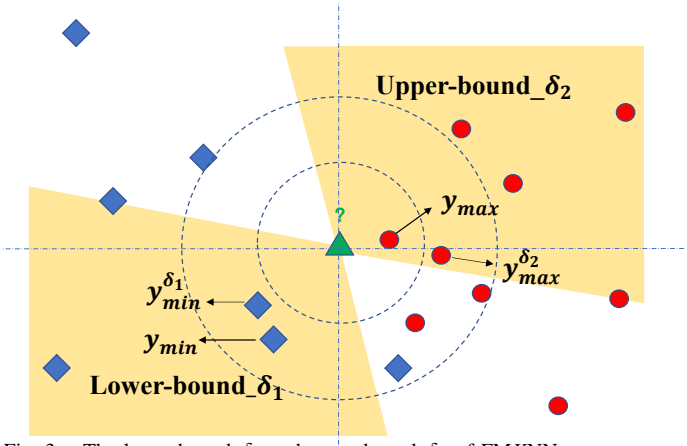

Fig. 3. The lower-bound_δ_1 and upper-bound_δ_2 of FMKNN.

Corresponding to the explanations of $y_{\min}$ and $y_{\max}$ in MKNN, in FMKNN $y_{\min}^{\delta_1}$ is the superlative class label of all the training instances that are fuzzily dominated by $(\mathbf{x}_0, y_0)$ in the degree ranging from $[\delta_1, 1]$, and conversely $y_{\max}^{\delta_2}$ is the lowest class label of all the training instances which fuzzily dominate $y_{\max}^{\delta_2}$ in the degree ranging from $[\delta_2, 1]$.

By tuning the thresholds δ_1 and δ_2, the lower-bound and upper-bound will be extended slightly in different degrees, so that the instances which determine the endpoints of the interval $[y_{\min}, y_{\max}]$ can be ignored if they are polluted by noises. The extended bounds are noted as lower-bound_δ_1 and upper-bound_δ_2, and are shown in Fig. 3.

Once the thresholds δ_1 and δ_2 are fixed, the interval $[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}]$ is determined, then the K-nearest neighbors of $(\mathbf{x}_0, y_0)$ are extracted from the training instances whose decision values lie in $[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}]$.

Different from the original MKNN where the dominance relations between the predicted instance and its K-nearest neighbors are absolute, in FMKNN the dominance relations are fuzzified, and MKNN can be considered as a special case of FMKNN.

B. The Flow of FMKNN

The entire process of FMKNN for predicting a new instance $(\mathbf{x}_0, y_0)$ mainly includes the following four steps, and can be exemplified in Algorithm 1.

Step 1. Calculate the degree of fuzzy dominance relation r_{i0} between each training instance $(\mathbf{x}_i, y_i)$ and the predicted instance $(\mathbf{x}_0, y_0)$ with (2), and obtain $y_i \in \left[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}\right]$ with (10) and (11).

Step 2. Extract the candidates for nearest neighbors of $(\mathbf{x}_0, y_0)$ from the training data whose target decision values are in the interval $[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}]$, and calculate the Euclidean distances between the predicted instance $(\mathbf{x}_0, y_0)$ and each candidate $(\mathbf{x}_C, y_C)$ extracted in Step 2 with (12).

Step 3. Select out the K-nearest neighbors with the first K minimum distances to $(\mathbf{x}_0, y_0)$.

Step 4. Predict the decision value $\hat{y}_0$ of $(\mathbf{x}_0, y_0)$ by using the majority voting strategy for symbolic decision variables and

$$\operatorname{dis}\left(\left(\mathbf{x}_0, y_0\right), \left(\mathbf{x}_C, y_C\right)\right) = \sqrt{\left(x_{01} - x_{C1}\right)^2 + \cdots + \left(x_{0k} - x_{Ck}\right)^2 + \cdots + \left(x_{0n} - x_{Cn}\right)^2 + \left(y_0 - y_C\right)^2} \tag{12}$$

averaging the target decision values of the *K*-nearest neighbors for numeric decision variables.

Algorithm 1 FM*K*NN Algorithm

Input:

Training dataset–*D*, thresholds which control the fuzzy dominance degree between the predicted instance and training data–δ_1 and δ_2, the number of selected nearest neighbors–*K*, the instance to be predicted–$(\mathbf{x}_0, y_0)$.

Output:

The decision value of the predicted instance–$\hat{y}_0$

Initialize null matrices $\boldsymbol{pre\_y}_{\boldsymbol{min}}^{\delta_1}$, $\boldsymbol{pre\_y}_{\boldsymbol{max}}^{\delta_2}$ and ***DIS***;
for $(\mathbf{x}_i, y_i) \in D$ **do**
  Compute r_{i0} with **expression (2)**;
  if $r_{i0} \leq \delta_1$ **then**
    Store y_i into $\boldsymbol{pre\_y}_{\boldsymbol{min}}^{\delta_1}$;
  else if $r_{i0} \geq \delta_2$
    Store y_i into $\boldsymbol{pre\_y}_{\boldsymbol{max}}^{\delta_2}$;
  end if
Calculate the maximum value in $\boldsymbol{pre\_y}_{\boldsymbol{min}}^{\delta_1}$ with **expression (10)** and denote it as $y_{\min}^{\delta_1}$;
Calculate the minimum value in $\boldsymbol{pre\_y}_{\boldsymbol{max}}^{\delta_2}$ with **expression (11)** and denote it as $y_{\max}^{\delta_2}$; ▷ Step 1

end for
for $(\mathbf{x}_i, y_i) \in D$ **do**
  if $y_i \in \left[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}\right]$ **then**
    Calculate the Euclidean distance dis_{0i} between $(\mathbf{x}_0, y_0)$ and $(\mathbf{x}_i, y_i)$ with **expression (12)** and store it to ***DIS***;
  end if
end for ▷ Step 2
Find the *K*-nearest neighbors of $(\mathbf{x}_0, y_0)$, which correspond to the first *K* minimum distances of ***DIS***; ▷ Step 3

Obtain $\hat{y}_0$ by using the majority voting strategy for symbolic decision variables and averaging the target decision values of the *K*-nearest neighbors for numeric decision variables; ▷ Step 4

return $\hat{y}_0$.

C. An Example

Here an example is given to illustrate vividly how our proposed FM*K*NN can counter the influence of noise data, which is shown as Fig. 4 where the dots, pentagrams, triangles and squares respectively represent two-dimensional monotonic instances belonging to class 1, class 2, class 3 and class 4 in order of categories from low to high, and point *A* stands for an unseen instance to be predicted. The coordinates of each point represent two feature values of an instance. Obviously,

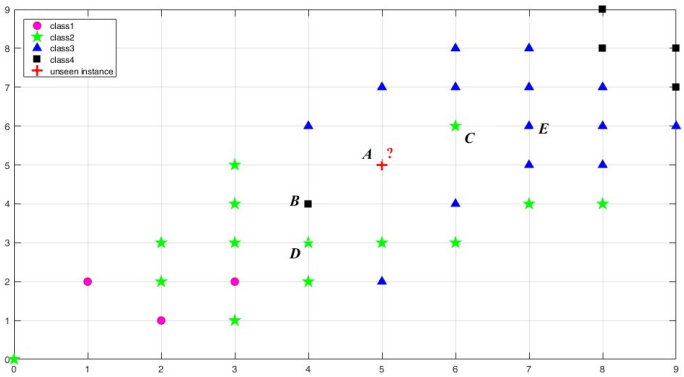

Fig. 4. A vivid example of FM*K*NN.

we can see that points *B* and *C* are noises which violate the monotonicity constraints.

Under the prediction rule of M*K*NN, the predicted class label $f(A)$ of the unseen instance *A* would be controlled by its closest inferior neighbor *B* and closest superior neighbor *C*, that is $4 < f(A) < 2$, which is unreasonable.

However, in our proposed FM*K*NN, according to (1) and (2), the fuzzy dominance degree δ_1 of *A* over *B* is 0.7272, and δ_2 of *C* over *A* is 0.7536. By fine-tuning the values of δ_1 and δ_2, such as letting $\delta_1 = 0.8080$ and $\delta_2 = 0.8285$, noises *B* and *C* are skipped, then *D* and *E* would be the instances which limit $f(A)$ to interval [2,3]. Then we can predict $f(A)$ by implementing the *K*-nearest neighbors strategy on the instances belonging to class 2 and 3.

V. Fuzzy Monotonic -Nearest Neighbors vs. Monotonic Fuzzy *K*-Nearest Neighbors Theoretically

Both MF*K*NN and FM*K*NN are proposed to improve the robustness of M*K*NN without the need for data preprocessing, and the best performances of the two algorithms are obtained by finding a balance between the classification accuracy and the maintenance degree of monotonicity. In this section, in order to illustrate their respective characteristics, the differences between the two algorithms are discussed from the following aspects.

*1) **Types of addressed problems**:* MF*K*NN can only solve classification problems with symbolic decision variables by converting crisp class labels into the form of class memberships, without concerning regression problems with monotonicity constraints.

However, FM*K*NN can deal with both classification and regression problems with monotonicity constraints. When the decision variable is symbolic, the prediction of the predicted instance is obtained by using the majority voting strategy which means that the most common class label of the *K*-nearest neighbors is considered as the final result, and when the decision variable is numeric, the average target decision

value of the *K*-nearest neighbors is assigned to the predicted instance.

*2) **Mechanisms for fuzzifying**:* In MF*K*NN, the fuzzifying mechanism acts on converting crisp class labels into fuzzy class membership vectors, which can decrease the impacts of all the noise data whose class labels are wrongly changed.

In FM*K*NN, the fuzzifying mechanism aims to fuzzify the dominance relations between two instances, especially between incomparable instances, through which the noises that determine the selection range of the *K*-nearest neighbors can be ignored.

*3) **Strategy for extraction of the K-nearest neighbors**:* In MF*K*NN, the *K*-nearest neighbors of a predicted instance are selected from the training instances whose class labels are in the interval $[y_{\min}, y_{\max}]$ which is controlled by the instances superior or inferior to the predicted instance absolutely. Besides, both Step 1 and Step 2 of MF*K*NN involve the extraction of *K*-nearest neighbors.

In FM*K*NN, the decision values of the *K*-nearest neighbors are limited to the interval $[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}]$ which is determined by the instances fuzzily dominating or dominated by the predicted instance. Additionally, only Step 2 of FM*K*NN concerns the extraction of *K*-nearest neighbors.

*4) **Mechanisms for resisting noises**:* In MF*K*NN, the impacts of all the noise data are decomposed during the process of converting crisp class labels into class membership vectors. The reason is that for each instance, its class membership to every category is computed based on the class distribution of its multiple nearest neighbors as (6), therefore, even though some neighbors are polluted, the calculation results of class memberships can only be slightly affected.

In FM*K*NN, by fuzzifying the dominance relations between instances, the decision value range of the *K*-nearest neighbors to be selected can be slightly extended in different degrees. Therefore, if the instances which determine the endpoints of interval $[y_{\min}, y_{\max}]$ are noises whose decision values are wrongly changed, they can be replaced with their incomparable instances whose decision values are the same with the original decision values of the noise data before contaminated.

*5) **Strategy for maintaining the monotonicity of predictions**:* The strategy for maintaining the monotonicity of MF*K*NN is the same with that of the original M*K*NN, which is implemented by restricting the predicted class label to an interval $[y_{\min}, y_{\max}]$ determined by the training instances which dominate or are dominated by the predicted instance absolutely.

In FM*K*NN, the interval $[y_{\min}^{\delta_1}, y_{\max}^{\delta_2}]$ where the predicted decision values lie is determined by the training instances which have fuzzified dominance relations with the predicted instance. Through fuzzifying the dominance relations, the noises that impact the interval $[y_{\min}, y_{\max}]$ can be ignored, so that the monotonicity of FM*K*NN can be maintained better than that of MF*K*NN.

VI. Experimental framework

In order to highlight our motivation and contribution, we compared FM*K*NN [1] with other well-known monotonic classifiers taken from the state-of-the-art, which include both *K*NN-based and non-*K*NN-based classifiers. Most of the classifiers have been adopted to do experimental comparisons with a newly proposed algorithm for monotonic classification [10], [12], [30], [32], [33], [35], [39], [42]. The two groups of experiments were carried out independently on both artificial and real-world monotonic classification tasks. The following are the details of the experiments.

[1] http://github.com/Rose1987/FMKNN

A. Involved Classifiers

1) KNN-based classifiers:

- Original *K*-Nearest Neighbour (*K*NN) [37]
- Fuzzy *K*-Nearest Neighbour (F*K*NN) [40]
- Monotonic *K*-Nearest Neighbour (M*K*NN) [36]
- Monotonic Fuzzy *K*-Nearest Neighbour (MF*K*NN) [39]

2) non-KNN-based classifiers:

- Ordinal Stochastic Dominance Learning (OSDL) [7]
- Ordinal Learning Module (OLM) [33]
- Monotonic Multi-Layer Perceptron network (MonMLP) [24]
- C4.5 decision tree for Monotonic Induction (MID) [48]
- Rank Discrimination Measure Tree (RDMT) [15]
- Partially Monotonic Decision Tree (PMDT) [8]

B. Performance Metrics

An ideal monotonic classifier should keep a good balance between the classification accuracy and the monotonicity degree of predictions. In the experiments, we adopt the classification accuracy (ACCU) [46], the mean absolute error (MAE) [47] and the non-monotonicity index (NMI) [48] to evaluate the performances of monotonic classifiers.

ACCU is the proportion of correctly classified testing instances in all the testing instances, which is a commonly used metric for measuring conventional classification problems.

$$ACCU = \frac{TI}{TI + FI}, \tag{13}$$

where TI is the number of instances that are correctly classified; FI is the number of instances that are incorrectly classified.

Different from ACCU, MAE is more suitable for monotonic classifications. It can reflect the order information of the predictive results, which is the average absolute differences between the predicted ranks and the target ones.

$$MAE = \frac{1}{M}\sum_{i=1}^{M} |\mathcal{O}(\hat{y}_i) - \mathcal{O}(y_i)|, \tag{14}$$

where *M* is the number of instances to be predicted, y_i is the target decision value of the *i*-th instance, $\hat{y}_i$ is the predicted decision value of the *i*-th instance, $\mathcal{O}(y_i)$ is the rank of y_i in the sequence of the decision values $\{l_1, l_2, \cdots, l_C\}$ with order $l_1 < l_2 < \cdots < l_C$, and similarly $\mathcal{O}(\hat{y}_i)$ is that of $\hat{y}_i$, that is if $y_i = l_\alpha$, then $\mathcal{O}(y_i) = \alpha$, where $\alpha = 1, 2, \cdots, C$.

Although MAE can measure prediction results from the perspective of order information, it does not consider the monotonicity constraints. In order to present a comprehensive

evaluation, we need another metric NMI to measure the capability of the classifiers for maintaining monotonicity. NMI was first provided by Ben-David [1], which is the ratio of instance pairs that violate the monotonicity constraints.

$$NMI = \frac{NMP}{M^2 - M}, \tag{15}$$

where NMP is the number of monotonicity violation instance pairs.

C. Datasets

In order to compare the performances of FM*K*NN and MF*K*NN, we conduct experiments on 20 monotonic datasets which are derived from UCI [50], KEEL [51] and the homepage of WEKA (http://www.cs.waikato.ac.nz/ml/weka/). All the datasets are widely used in the field of monotonic classification [10], [12], [15], [17], [30]-[33], [35], [42], [43]. The main characteristics of these datasets are listed in Table I, where Ins., Al. and Cl. represents the number of training instances, features and categories, respectively; I.P. is the proportion of incomparable instance pairs in the total instance pairs; N.P. denotes the percentage of the instance pairs violating the monotonicity constraints in the comparable instance pairs.

Actually, it is difficult to find a real large-scale dataset with a big percentage of noises. In order to make the experimental results more convincing, an artificial monotonic dataset named Artiset is generated according to (16), where there are 20000 instances and N.P.=34.15%. The noises in Artiset are introduced under the mechanism of uniform label noises which are noisy completely at random (NCAR), by using the *rand*(*u*,*v*) function in Matlab, which outputs a matrix with *u* rows and *v* columns, whose elements are uniformly distributed random numbers ranged in [0,1]. In addition, because if the noises are also involved in the testing process, the testing accuracy which measures the performance of a classifier will be affected negatively, the noises in Artiset are introduced only in the training partition.

$$g(x_1, x_2) = round\left(\left(x_1 + \frac{x_2^2 - x_1^2}{2}\right) * Cl\right), \tag{16}$$

where *round*$(\cdot)$ is a function which rounds a real number to an integer.

In the original datasets involved in the experiments, different features have different dimensions and magnitudes. If the datasets are directly used in the *K*NN-based algorithms, the feature with larger magnitude will play an exaggerated role on the distance between instances. As a consequence, the accuracy of the experimental results will be seriously affected. Therefore, it is necessary to normalize the original feature values before training. In this article, the Min-Max normalization is adopted, of which the expression is

$$x' = \frac{x_0 - min}{\max - \min}, \tag{17}$$

where x_0 is the original value of an instance on a feature; min is the minimum value of all the instances on this feature, while max is on the contrary. After the Min-Max normalization, all the original feature values fall into the range [0, 1].

TABLE I
DESCRIPTION OF THE 20 SELECTED DATASETS

Dataset	Ins.	Al.	Cl.	I.P.	N.P.	Source
Arcene	900	10000	2	97.98%	0.00%	UCI
Boston Housing	506	13	4	85.15%	3.98%	UCI
Breast Cancer	699	9	2	38.43%	$3.93 \times 10^{-2}\%$	UCI
Car Evaluation	1728	6	4	85.64%	$3.89 \times 10^{-2}\%$	UCI
Machine CPU	209	6	4	50.47%	0.00%	UCI
Parkinson's Disease	756	754	2	99.73%	0.00%	UCI
Q_Bankruptcy	250	6	2	56.23%	0.00%	UCI
WDG40	5000	40	3	99.96%	0.00%	UCI
Wine	178	13	3	96.31%	2.41%	UCI
Australian	690	14	2	97.80%	1.53%	KEEL
Pima	768	8	2	92.42%	2.16%	KEEL
Vowel	990	14	11	98.77%	10.46%	KEEL
Wine-red	1599	12	6	98.83%	3.09%	KEEL
Wisconsin	683	9	2	41.96%	$1.31 \times 10^{-2}\%$	KEEL
Balance	625	4	3	74.36%	31.01%	WEKA
ERA	1000	4	9	83.23%	17.44%	WEKA
ESL	488	4	9	29.35%	1.32%	WEKA
LEV	1000	4	5	75.92%	5.23%	WEKA
SWD	1000	10	4	87.38%	6.95%	WEKA
Artiset	20000	2	5	50.28%	34.15%	- -

VII. EXPERIMENTAL STUDIES

In the experiments, all the classifiers are separately implemented by carrying out the 10-fold cross validation (10-CV) scheme on each dataset. The parameters of algorithms are firstly initialized and used for experiments, and then the parameter with best performance is finally selected for each algorithm and is listed in Table II. All the results are shown in Tables III-VIII, and the best indicator value for each dataset has been marked in bold.

TABLE II
PARAMETERS SET FOR THE ALGORITHMS COMPARED

Algorithm	Parameters
*K*NN	*K*=5, distance = euclidean, neighborsType = inRange
F*K*NN	*K* = 9, *k* = 5, distance = euclidean
M*K*NN	*K* = 5, distance = euclidean, neighborsType = inRange
MF*K*NN	*K* = 9, *k* = 5, distance = euclidean
OSDL	balanced = No, classificationType = median, lowerBound = 0, upperBound = 1, tuneInterpolationParameter = No, weighted = No, interpolationStepSize = 10, interpolationParameter = 0.5
OLM	modeResolution = conservative modeClassification = conservative
MonMLP	default parameters, hidden1 = 8 iter.max = 1000, monotonic = all att
MID	R = 1, confidence = 0.25, items per leaf = 2
RDMT	H = Pessimistic rank discrimination measure, measureThreshold = 0, items per leaf = 2
PMDT	threshold $\theta = 0$, items per leaf = 2
FM*K*NN	*K*=5, $\delta_1 = 0.8080$, $\delta_2 = 0.8285$

A. Experimental Results and Discussions on KNN-based Classifiers

Table III shows the prediction abilities of the five *K*NN-based models. We can clearly see that the prediction accuracies of the traditional *K*NN and F*K*NN are extremely poor in most

TABLE III
COMPARISON RESULTS WITH *K*NN-BASED CLASSIFIERS IN ACCU

Dataset	*K*NN	F*K*NN	M*K*NN	MF*K*NN	FM*K*NN
Arcene	0.4510	0.7321	0.6516	0.6347	**0.8739**
Boston Housing	0.0132	0.7174	0.6126	0.6561	**0.9072**
Breast Cancer	0.6128	0.8055	0.6391	0.5981	**0.8929**
Car Evaluation	0.9459	0.9311	0.9711	0.9740	**0.9843**
Machine CPU	0.1270	0.6699	0.6890	0.7033	**0.8011**
Parkinson's Disease	0.4052	0.8631	0.5589	0.5161	**0.9639**
Q_Bankruptcy	0.9867	0.9060	0.5128	0.9561	**0.9960**
WDG40	0.6981	0.8039	0.5991	0.6672	**0.9192**
Wine	0.7162	0.7930	0.7121	0.6493	**0.8673**
Australian	0.6682	0.8007	0.7659	0.7033	**0.8892**
Pima	0.5581	0.7996	0.7238	0.6900	**0.8513**
Vowel	0.6821	0.8601	0.8003	0.6338	**0.9428**
Wine-red	0.5162	0.7991	0.7290	0.5990	**0.9001**
Wisconsin	0.7653	0.9678	0.9649	0.9653	**0.9707**
Balance	0.1658	0.8896	0.8624	0.9307	**0.9864**
ERA	0.1933	0.1730	0.1990	0.2420	**0.7351**
ESL	0.6781	0.6783	0.6332	0.7036	**0.8095**
LEV	0.5287	0.6020	0.4630	0.6377	**0.8693**
SWD	0.5667	0.5350	0.5200	0.5807	**0.7416**
Artiset	0.8058	0.9339	0.9199	0.9653	**0.9815**

TABLE IV
COMPARISON RESULTS WITH *K*NN-BASED CLASSIFIERS IN MAE

Dataset	*K*NN	F*K*NN	M*K*NN	MF*K*NN	FM*K*NN
Arcene	6.3639	4.8929	0.8957	3.3062	**0.1396**
Boston Housing	2.7303	0.3241	0.4901	0.3972	**0.1500**
Breast Cancer	3.3852	2.6139	1.2531	2.0394	**0.5627**
Car Evaluation	0.0637	0.0793	0.0359	0.0295	**0.0137**
Machine CPU	3.7778	0.3589	0.3301	0.1667	**0.1055**
Parkinson's Disease	9.1015	6.3120	0.9015	5.3007	**0.6203**
Q_Bankruptcy	0.0133	0.5040	0.0040	0.0032	**0.0029**
WDG40	2.9591	3.2218	0.5261	1.9036	**0.1938**
Wine	6.5136	4.6813	1.9203	2.8125	**0.0931**
Australian	4.2316	5.0634	2.3901	4.0032	**0.5163**
Pima	3.6152	2.5178	1.5638	2.0614	**0.9623**
Vowel	5.9128	4.9003	1.0926	2.0821	**0.3333**
Wine-red	2.6901	2.0351	0.6203	1.8823	**0.6037**
Wisconsin	0.0311	0.0322	0.0337	0.0347	**0.0293**
Balance	1.3262	0.1424	0.1504	0.0853	**0.0318**
ERA	1.5033	1.6660	1.4270	1.2813	**0.0862**
ESL	0.3630	0.3484	0.3791	0.3149	**0.2855**
LEV	0.4333	0.4330	0.5740	0.3927	**0.1672**
SWD	0.4700	0.5180	0.4840	0.4370	**0.2100**
Artiset	0.1375	0.0661	0.0771	0.0691	**0.0395**

TABLE V
COMPARISON RESULTS WITH *K*NN-BASED CLASSIFIERS IN NMI

Dataset	*K*NN	F*K*NN	M*K*NN	MF*K*NN	FM*K*NN
Arcene	0.2063	0.1900	0.2104	0.1064	**0.0573**
Boston Housing	0.3003	0.0004	**0.0000**	0.0001	**0.0000**
Breast Cancer	0.0951	0.0630	0.1066	0.0088	**0.0032**
Car Evaluation	0.2021	0.0002	**0.0000**	**0.0000**	**0.0000**
Machine CPU	0.0825	0.0058	**0.0000**	0.0017	**0.0000**
Parkinson's Disease	0.1938	0.0851	0.0659	0.0091	**0.0051**
Q_Bankruptcy	0.1350	0.3000	0.1089	0.0600	**0.0330**
WDG40	0.2008	0.1654	0.1132	0.0805	**0.0634**
Wine	0.1229	0.0960	0.0815	0.0837	**0.0209**
Australian	0.1006	0.0991	0.0617	0.0723	**0.0097**
Pima	0.3004	0.2098	0.2007	0.1615	**0.0859**
Vowel	0.2934	0.3000	0.2187	0.1520	**0.0674**
Wine-red	0.1867	0.1532	0.1020	0.0806	**0.0032**
Wisconsin	0.3033	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Balance	0.0868	**0.0000**	0.0001	0.0001	**0.0000**
ERA	0.0069	0.0141	0.0056	0.0052	**0.0007**
ESL	0.0310	0.0014	0.0012	0.0004	**0.0002**
LEV	0.0123	0.0010	0.0036	0.0009	**0.0005**
SWD	0.2004	0.0027	0.0005	0.0007	**0.0002**
Artiset	0.1431	**0.0000**	**0.0000**	**0.0000**	**0.0000**

cases. The reason is that both of the two models don't take the monotonicity constraints into account. Besides, M*K*NN doesn't show any advantage on the prediction accuracy. MF*K*NN can provide satisfactory prediction accuracies on most of the datasets, however, our proposed FM*K*NN achieves the highest classification accuracy on almosgt all the datasets. The advantage of FM*K*NN over MF*K*NN is significant, especially on the datasets with high proportion of incomparable instance pairs and noises, such as Arcene, Boston Housing, Parkinson's Disease, WDG40, Wine, Australian, Pima, Vowel, Wine-red, ERA and SWD.

Tables IV and V comprehensively reflect the monotonicity maintenance abilities of the five *K*NN-based models. Similar to the results presented in Table III, the traditional *K*NN and F*K*NN perform poorly on most of the datasets. M*K*NN only shows its effectiveness on datasets which are purely monotonic, such as Machine CPU, Q_Bankruptcy and Artiset, which confirms the strong sensitivity of M*K*NN to noise data. In comparison with MF*K*NN, FM*K*NN shows obvious advantage on maintaining the monotonicity of predictions, especially on the datasets polluted badly by noises.

According to the results presented in Tables III-V, for each metric, we calculate the average improvement degree of FM*K*NN in terms of other *K*NN-based classifiers with the best result on all the datasets, and discover that FM*K*NN improves the ACCU, MAE and NMI by 28%, 4% and 29%, respectively.

Moreover, in order to test whether our proposed FM*K*NN has significant advantage over the other four models statistically, the results are statistically analyzed by using the Friedman test [52], [53] and its corresponding post-hoc test (the Nemenyi test) [54], which can make statistical comparison of multiple classifiers over numerous datasets. The comparison results are presented in the form of Friedman test figure where the less overlap the two algorithms have, the more significant the difference is. The Friedman test figures of *K*NN-based classifiers on ACCU, MAE and NMI are shown in Fig. 5 with the confidence level $\alpha = 0.05$. As shown in Fig. 5, in terms of ACCU, FM*K*NN outperforms the other four algorithms significantly, while MF*K*NN performs worse than FM*K*NN and F*K*NN, and there is no significant differences among MF*K*NN, M*K*NN and *K*NN; in terms of MAE, FM*K*NN has obvious advantage over MF*K*NN, F*K*NN and *K*NN and slight advantage over M*K*NN, while MF*K*NN performs worse than FM*K*NN and M*K*NN, and doesn't show prominent advantage over F*K*NN and *K*NN; in terms of NMI, FM*K*NN outperforms all the other algorithms significantly, while MF*K*NN performs worse than FM*K*NN and doesn't have obvious advantage over M*K*NN, F*K*NN and *K*NN.

Additionally, in order to compare MF*K*NN and FM*K*NN in improving the anti-noise performance of M*K*NN, we implemented the three models on five datasets which include Arcene, Machine CPU, Parkinson's Disease, WDG40 and Artiset, by gradually increasing the ratio of noises artificially. Figs. 6-8 show the average impact of noise ratio on ACCU, MAE and NMI, respectively.

From Figs. 6-8 we can see that, as the ratio of noise data increases, the performances of all three models decreases. The downturns of the ACCU polylines are obvious, besides the MAE and NMI polylines show upward trends. The difference among the three models in the results is that the rise or fall

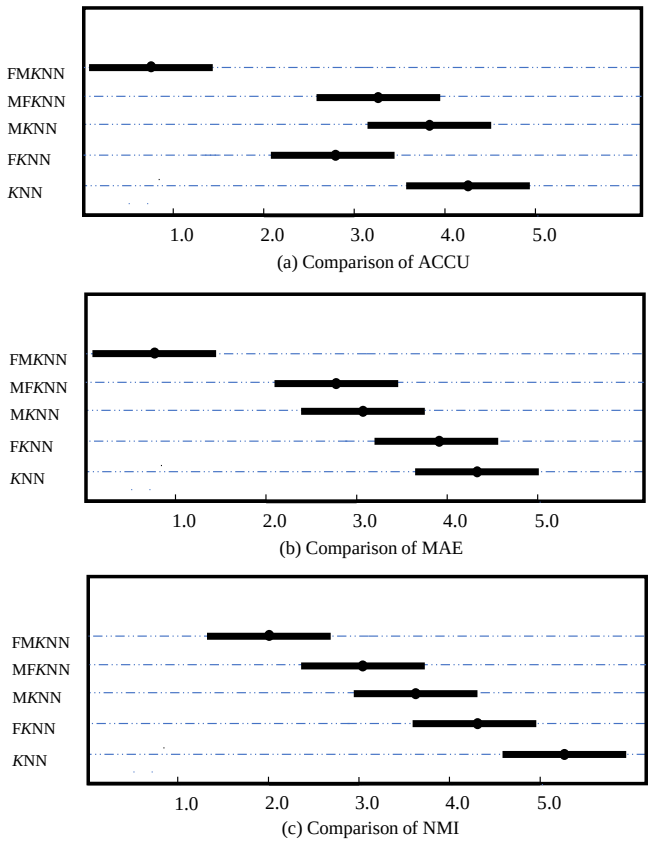

Fig. 5. Friedman test figure of *K*NN-based classifiers.

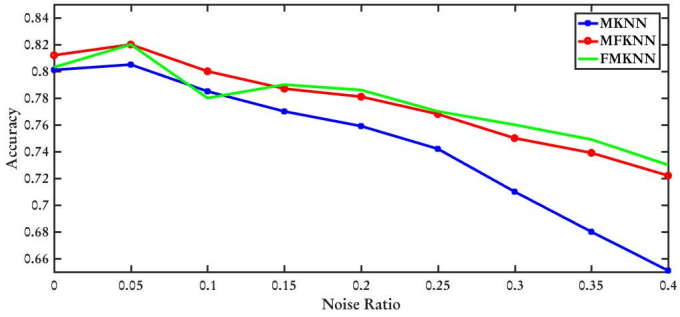

Fig. 6. Impact of noise ratio on ACCU.

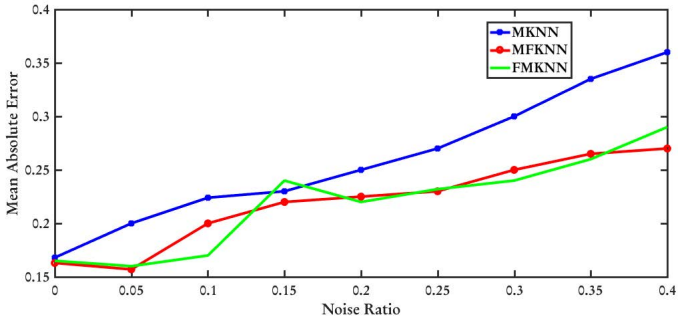

Fig. 7. Impact of noise ratio on MAE.

amplitude of the polylines in terms of ACCU, MAE and NMI of M*K*NN is larger than those of MF*K*NN and FM*K*NN, especially when the proportion becomes bigger. The larger

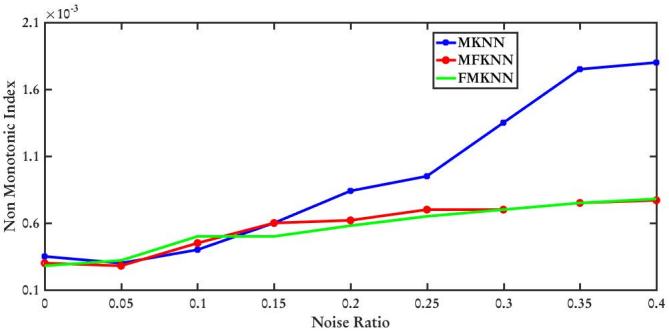

Fig. 8. Impact of noise ratio on NMI.

TABLE VI
COMPARISON RESULTS WITH NON-*K*NN-BASED CLASSIFIERS IN ACCU

Dataset	OSDL	OLM	MonMLP	MID	RDMT	PMDT	MF*K*NN	FM*K*NN
Arcene	0.6729	0.7068	0.8120	0.5563	0.8412	0.7982	0.8331	**0.9320**
Boston Housing	0.2787	0.5277	0.3979	0.6739	0.6304	0.6739	0.6561	**0.9072**
Breast Cancer	0.0720	0.6984	0.8001	0.8201	0.6937	0.7925	0.8066	**0.8924**
Car Evaluation	0.9549	0.9543	0.8474	0.8027	0.7297	0.9682	0.9740	**0.9881**
Machine CPU	0.3146	0.7003	0.6973	0.7033	0.6670	0.6923	0.6628	**0.8372**
Parkinson's Disease	0.7714	0.7268	0.6258	0.5986	0.6649	0.7259	0.7992	**0.8561**
Q_Bankruptcy	0.8957	0.9758	0.5944	0.9118	0.9192	0.9549	0.9960	**0.9978**
WDG40	0.7001	0.6301	0.7256	0.8045	0.8842	0.9002	0.8264	**0.9153**
Wine	0.8056	0.8416	0.5979	0.5694	0.8468	0.5976	0.8009	**0.9156**
Australian	0.6816	0.8130	0.6941	0.5684	0.6568	0.7618	0.8894	**0.9513**
Pima	0.5618	0.6545	0.6894	0.7564	0.5949	0.8492	0.7165	**0.8956**
Vowel	0.6498	0.5768	0.8741	0.5448	0.6746	0.8746	0.7741	**0.9041**
Wine-red	0.5741	0.6549	0.5498	0.7225	0.8613	0.8564	0.8741	**0.9421**
Wisconsin	0.9311	0.8679	0.8900	0.9001	0.9087	0.9653	0.9130	**0.9708**
Balance	0.6352	0.8320	0.9131	0.7808	0.7216	0.7792	0.9307	**0.9864**
ERA	0.2320	0.1690	0.2380	0.2760	0.2390	0.2430	0.2420	**0.7351**
ESL	0.6721	0.5738	0.7234	0.6414	0.5635	0.6598	0.7036	**0.8995**
LEV	0.6400	0.4250	0.6167	0.6070	0.5210	0.6370	0.6377	**0.8693**
SWD	0.5840	0.4160	0.5063	0.5540	0.5180	0.5830	0.5807	**0.8416**
Artiset	0.1952	0.7948	0.9463	0.7237	0.8749	0.8539	0.9309	**0.9767**

changing amplitude of M*K*NN illustrates that it is more sensitive to noises compared with MF*K*NN and FM*K*NN. From the figures, we can also notice that MF*K*NN and FM*K*NN perform basically the same when the proportion of noises increases.

The results presented in Figs. 6-8 confirm that when dealing with monotonic classification problems, M*K*NN is quite sensitive to noises that violate monotonicity constraints, because its performances on prediction accuracy, MAE and NMI deteriorate significantly with the increasing of the proportion of noises. On the contrary, both MF*K*NN and FM*K*NN are affected by noises gently, which illustrates that MF*K*NN and FM*K*NN have much stronger robustness than the original M*K*NN.

B. Experimental Results and Discussions on non-KNN-based Classifiers

Table VI shows the prediction accuracies of six non-*K*NN-based monotonic classifiers and those of MF*K*NN and FM*K*NN. We can clearly see that FM*K*NN and MF*K*NN can provide much higher prediction accuracy than the other six classifiers in most cases. Moreover, we can also see that FM*K*NN outperforms MF*K*NN on most of the datasets.

Table VII shows the mean absolute error between the outputs of the eight classifiers and the real class levels of samples on each dataset. According to the principle that the smaller the error, the stronger the approximation ability of a classifier, we can see that FM*K*NN shows stronger approximation ability than the other classifiers including MF*K*NN.

Table VIII shows the NMI values of the eight classifiers. We can see that FM*K*NN outperforms the other seven classifiers

TABLE VII
COMPARISON RESULTS WITH NON-*K*NN-BASED CLASSIFIERS IN MAE

Dataset	OSDL	OLM	MonMLP	MID	RDMT	PMDT	MF*K*NN	FM*K*NN
Arcene	2.0135	1.8923	2.0643	3.0445	0.9231	0.8167	0.7124	**0.1235**
Boston Housing	0.9368	0.5988	0.7655	0.3893	0.4249	0.3676	0.3972	**0.3005**
Breast Cancer	1.5912	2.0451	1.5331	1.6710	2.0541	2.4681	1.2611	**1.2098**
Car Evaluation	0.0475	0.0538	0.1599	0.2506	0.3079	0.0365	0.0295	**0.0019**
Machine CPU	0.8980	0.3622	0.3603	0.4109	0.3600	0.3533	0.3158	**0.1055**
Parkinson's Disease	3.0156	3.8213	4.2613	2.1968	3.2364	1.3513	2.6568	**0.7156**
Q_Bankruptcy	0.0905	0.0194	0.3682	0.0155	0.0157	0.0078	0.0040	**0.0029**
WDG40	5.2684	5.6231	4.5816	5.0279	3.6484	4.0561	3.2340	**2.0010**
Wine	0.9264	1.0256	1.3618	2.0125	0.8264	1.0546	0.8843	**0.3158**
Australian	6.2315	4.2658	5.2687	5.2698	3.2985	2.0645	4.2982	**1.0234**
Pima	2.2131	2.5469	2.9751	3.1246	2.1664	1.0524	3.1657	**0.4599**
Vowel	2.4568	3.5084	2.4575	3.0645	1.9546	2.0004	1.6549	**0.8814**
Wine-red	8.4316	8.1034	9.0213	7.1546	6.1349	4.1245	7.4618	**1.4565**
Wisconsin	0.0410	0.1127	0.1396	0.0483	0.0498	0.0439	0.0347	**0.0211**
Balance	0.4912	0.1920	0.0992	0.3360	0.3840	0.2560	0.0853	**0.0318**
ERA	1.2850	2.1500	1.2317	1.2970	1.30660	1.2870	1.2813	**0.0862**
ESL	0.3607	0.4734	**0.2910**	0.3934	0.4918	0.3750	0.3149	0.3855
LEV	0.3920	0.6680	0.4170	0.4290	0.5430	0.3940	0.3927	**0.1672**
SWD	0.4000	0.7630	0.5167	0.4750	0.4990	0.4340	0.4370	**0.2100**
Artiset	1.6897	0.2082	0.0638	0.3123	0.1251	0.1471	0.0691	**0.0537**

TABLE VIII
COMPARISON RESULTS WITH NON-*K*NN-BASED CLASSIFIERS IN NMI

Dataset	OSDL	OLM	MonMLP	MID	RDMT	PMDT	MF*K*NN	FM*K*NN
Arcene	0.3201	0.2659	0.2579	0.3065	0.0935	0.2216	0.1026	**0.0529**
Boston Housing	**0.0000**	0.0003	0.0007	0.0022	0.0010	0.0010	**0.0000**	**0.0000**
Breast Cancer	0.4282	0.4002	0.3551	0.2586	0.1026	0.3599	0.1546	**0.0249**
Car Evaluation	**0.0000**	**0.0000**	0.0001	0.0046	0.0002	**0.0000**	**0.0000**	**0.0000**
Machine CPU	**0.0000**	0.0014	0.0001	0.0037	0.0047	0.0028	0.0002	**0.0000**
Parkinson's Disease	0.2151	0.3265	0.1057	0.1546	0.3021	0.0541	0.1235	**0.0301**
Q_Bankruptcy	0.1265	0.4871	0.2351	0.1055	0.0656	0.2416	0.0216	**0.0054**
WDG40	0.4657	0.0513	0.2154	0.3468	0.4531	0.3508	0.2549	**0.0023**
Wine	0.2653	0.3216	0.0847	0.3152	0.2156	0.1245	0.1569	**0.0315**
Australian	0.3165	0.2549	0.5976	0.3591	0.0426	0.3254	0.2167	**0.0254**
Pima	0.1544	0.1024	0.2005	0.2533	0.2548	0.0961	0.1354	**0.0364**
Vowel	0.2259	0.2589	0.2359	0.1369	0.2005	0.1950	**0.0715**	0.1058
Wine-red	0.2358	0.0345	0.2154	0.1698	0.2354	0.1852	0.1548	**0.0026**
Wisconsin	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0001	**0.0000**	**0.0000**	**0.0000**
Balance	0.0006	**0.0000**	**0.0000**	0.0017	0.0029	0.0010	**0.0000**	**0.0000**
ERA	0.0049	0.0063	0.0026	0.0082	0.0085	0.0058	0.0052	**0.0007**
ESL	0.0006	0.0025	0.0003	0.0021	0.0066	0.0032	0.0004	**0.0002**
LEV	0.0004	0.0043	0.0008	0.0018	0.0086	0.0006	0.0004	**0.0001**
SWD	0.0009	0.0015	0.0004	0.0020	**0.0000**	0.0010	0.0007	0.0001
Artiset	**0.0000**	**0.0000**	**0.0000**	0.0039	**0.0000**	0.0001	**0.0000**	**0.0000**

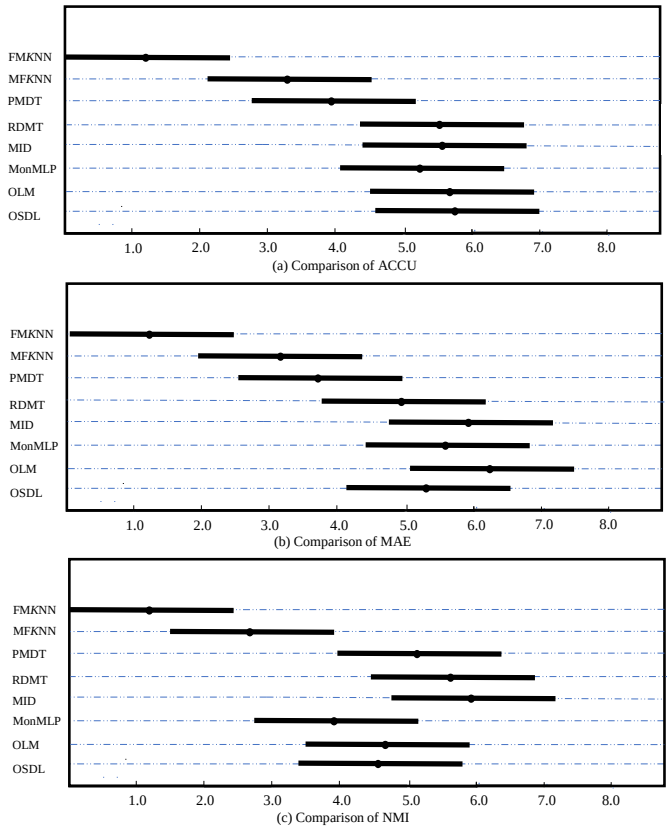

Fig. 9. Friedman test figure of non-*K*NN-based classifiers.

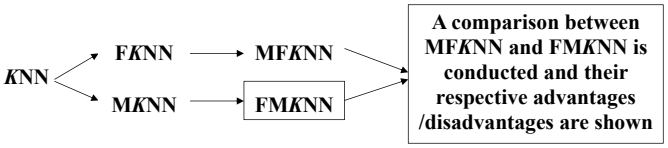

Fig. 10. The relations diagram of the five discussed models.

obviously, especially on the datases with more incomparable instance pairs and noises, such as Boston Housing, ERA, LEV, SWD and Balance.

According to the results presented in Tables VI-VIII, for each metric, we calculate the average improvement degree of FM*K*NN in terms of other non-*K*NN-based classifiers with the best result on all the datasets, and discover that FM*K*NN improves the ACCU, MAE and NMI by 24%, 11% and 20%, respectively.

Additionally, the Friedman statistical test and the Nemenyi post-hoc test are adopted again to analyze the results from Tables VI-VIII, and the comparison results are shown in Fig. 9. We can see that FM*K*NN presents significant advantage over all the other six non-*K*NN based classifiers in terms of ACCU, MAE and NMI, while MF*K*NN performs worse than FM*K*NN in terms of all the metrics and nearly doesn't show obvious advantage over all the other algorithms.

Based on the experimental results shown in Fig. 5 and Fig. 9, a conclusion can be drawn that FM*K*NN has significant advantage over MF*K*NN on all the involved datasets.

VIII. CONCLUSION

In this paper, to enhance the robustness of M*K*NN for managing monotonic classification problems, we proposed a new modification model named FM*K*NN, which makes a first attempt to construct monotonic classifiers by making use of the fuzzy dominance relations between instances, especially between incomparable instances. Besides, we conducted comparative experiments on FM*K*NN and other models including *K*NN-based and non-*K*NN-based classifiers. In particular, we made a comprehensive comparison between FM*K*NN and MF*K*NN theoretically and experimentally. The relations among *K*NN, F*K*NN, M*K*NN, MF*K*NN and FM*K*NN are shown in Fig. 10 where the work proposed in this paper is marked with a black box.

Theoretically, the goals and the guiding ideas of MF*K*NN and FM*K*NN are basically the same. Both of them aim to enhance the resistance capability of M*K*NN to noises by making a trade-off between accuracy and the monotonicity maintanence degree of the predictions. The differences between the two models are the types of the problems to be addressed, mechanisms for fuzzifying, methods of resisting noises, strategies for extraction of the *K*-nearest neighbors, and ways to maintain the monotonicity of predictions.

Experimentally, by observing the experimental results on both *K*NN-based and non-*K*NN-based classifiers comprehensively, we can know that the best average improvement degrees

of FM*K*NN in terms of the *K*NN-based and non-*K*NN-based classifiers on all the involved datasets arrive at 28%, 11% and 29% with respect to ACCU, MAE and NMI, respectively. Besides, we can confirm that both FM*K*NN and MF*K*NN have much stronger robustness than the original *K*NN, F*K*NN, and M*K*NN and the other six non-*K*NN-based monotonic classifiers when dealing with monotonic classifications. However, we found that FM*K*NN has significant advantage over MF*K*NN from the view of Friedman statistical test and the Nemenyi post-hoc test, particularly on the datasets with much more noises and incomparable instance pairs.

REFERENCES

[1] A. Ben-David, L. Sterling, and Y.H. Pao, "Learning and classification of monotonic ordinal concepts," *Computational Intelligence*, vol. 5, no. 1, pp. 45-49, 1989.

[2] M.J. Kim, and I. Han, "The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms," *Expert Systems with Applications*, vol. 25, no. 4, pp. 637-646, 2003.

[3] R. Potharst, and A.J. Feelders, "Classification trees for problems with monotonicity constraints," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 1, pp. 1-10, 2002.

[4] C.C. Chen, and S.T. Li, "Credit rating with a monotonicity-constrained support vector machine model," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7235-7247, 2014.

[5] J.R. Cano, N.R. Aljohani, R.A. Abbasi, J.S. Alowidbi, and S. Garcia, "Prototype selection to improve monotonic nearest neighbor", *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 128-135, 2017.

[6] Q. Hu, M. Guo, D. Yu, and J. Liu, "Information entropy for ordinal classification," *Science China Information Sciences*, vol. 53, no. 6, pp. 1188-1200, 2010.

[7] S. Lievens, B. De Baets, and K. Cao-Van, "A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting," *Annals of Operations Research*, vol. 163, no. 1, pp. 115-142, 2008.

[8] S. Pei, and Q. Hu, "Partially monotonic decision trees," *Information Sciences*, vol. 424, pp. 104-117, 2018.

[9] H. Xu, W. Wang, and Y. Qian, "Fusing complete monotonic decision trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 104-117, 2017.

[10] S. Pei, Q. Hu, and C. Chen, "Multivariate decision trees with monotonicity constraints," *Knowledge-Based Systems*, vol. 112, pp. 14-25, 2016.

[11] S. Gonzalez, F. Herrera, and S. Garcia, "Managing monotonicity in classification by a pruned random forest," in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, Cham, Oct. 2015, pp. 53-60.

[12] Q. Hu, X. Che, L. Zhang, D. Zhang, M. Guo, and D. Yu, "Rank entropy-based decision trees for monotonic classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 11, pp. 2052-2064, 2011.

[13] J. W. Lee, D. S. Yeung, and X. Wang, "Monotonic decision tree for ordinal classification," in *SMC'03 Conference Proceedings*, IEEE., Oct. 2003, vol. 3, pp. 2623-2628.

[14] S. Giove, S. Greco, B. Matarazzo, and R. Slowinski, "Variable consistency monotonic decision trees," in *International Conference on Rough Sets and Current Trends in Computing*, Springer, Berlin, Heidelberg, Oct. 2002, pp. 247-254.

[15] C. Marsala, and D. Petturiti, "Rank discrimination measures for enforcing monotonicity in decision tree induction," *Information Sciences*, vol. 291, pp. 143-171, 2015.

[16] S. Gonzalez, F. Herrera, and S. Garcia, "Managing monotonicity in classification by a pruned adaboost," in *International Conference on Hybrid Artificial Intelligence Systems*, Springer, Cham, Apr. 2016, pp. 512-523.

[17] Y. Qian, H. Xu, J. Liang, B. Liu, and J. Wang, "Fusing monotonic decision trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 10, pp. 2717-2728, 2015.

[18] S. Gonzalez, F. Herrera, and S. Garcia, "Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity," *New Generation Computing*, vol. 33, no. 4, pp. 367-388, 2015.

[19] J. Wang, Y. Qian, F. Li, J. Liang, and W. Ding, "Fusing Fuzzy Monotonic Decision Trees," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 5, pp. 887-900, 2019.

[20] J. Yang, Y. Wang, and Q. Hu, "Monotonicity Extraction for Monotonic Bayesian Networks Parameter Learning," in *International Conference on Neural Information Processing*, Springer, Cham, Dec. 2018, pp. 571-581.

[21] N. P. Archer, and S. Wang, "Learning bias in neural networks and an approach to controlling its effect in monotonic classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 962-966, 1993.

[22] F. Fernandez-Navarro, A. Riccardi, and S. Carloni, "Ordinal neural networks without iterative tuning," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 11, pp. 2075-2085, 2014.

[23] H. Daniels, and M. Velikova, "Monotone and partially monotone neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 906-917, 2010.

[24] B. Lang, "Monotonic multi-layer perceptron networks as universal approximators," in *International Conference on Artificial Neural Networks*, 2005, PP. 31-37.

[25] M. C. Lo, B. H. Tsai, and S. T. Li, "Deep Learning for Monotonic Support Vector Machines," in *2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI)*, IEEE., Jul. 2018, pp. 530-535.

[26] C. C. Chen, H. C. Chuang, Y. C. Cheng, and S. T. Li, "Constraint Selection on Monotonic Support Vector Classifiers," in *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, IEEE., Jul. 2016, pp. 641-646.

[27] R. Dykstra, J. Hewett, and T. Robertson, "Nonparametric, isotonic discriminant procedures," *Biometrika*, vol. 86, no. 2, pp. 429-438, 1999.

[28] H. Daniels, and M. Velikova, "Derivation of monotone decision models from non-monotone data," Tilburg University, 2003.

[29] W. W. Pijls, and R. R. Potharst, "Repairing non-monotone ordinal data sets by changing class labels", No. EI 2014-29, 2014.

[30] Q. Hu, W. Pan, L. Zhang, D. Zhang, Y. Song, M. Guo, and D. Yu, "Feature selection for monotonic classification," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 1, pp. 69-81, 2011.

[31] Q. Hu, W. Pan, Y. Song, and D. Yu, "Large-margin feature selection for monotonic classification," *Knowledge-Based Systems*, vol. 31, pp. 8-18, 2012.

[32] S. Gonzalez, S. Garcia, S. T. Li, and F. Herrera, "Chain based sampling for monotonic imbalanced classification", *Information Sciences*, vol. 474, pp. 187-204, 2019.

[33] A. Ben-David, "Automatic generation of symbolic multiattribute ordinal knowledge-based DSSs: Methodology and applications," *Decision Sciences*, vol. 23, no. 6, pp. 1357-1372, 1992.

[34] J. R. Cano, and S. Garc¯a, "A First Attempt on Monotonic Training Set Selection," in *International Conference on Hybrid Artificial Intelligence Systems*, Springer, Cham, Jun. 2018, pp. 277-288.

[35] J. R. Cano, and S. Garc¯a, "Training set selection for monotonic ordinal classification," *Data and Knowledge Engineering*, vol. 112, pp. 94-105, 2017.

[36] W. Duivesteijn, and A. Feelders, "Nearest neighbour classification with monotonicity constraints," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, Berlin, Heidelberg, Sep. 2008, pp. 301-316.

[37] T. Cover, and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21-27, 1967.

[38] J. R. Cano, N. R. Aljohani, R. A. Abbasi, J. S. Alowidbi, and S. Garcia, "Prototype selection to improve monotonic nearest neighbor," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 128-135, 2017.

[39] S. Gonzalez, S. Garcia, S. Li, R. John, and F. Herrera, "Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise," *Neurocomputing*, vol. 439, pp. 106-121, 2021.

[40] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, vol. 4, pp. 580-585, 1985.

[41] B. Frenay, and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845-869, 2013.

[42] J. Alcala-Fdez, R. Alcala, S. Gonzalez, Y. Nojima, and S. Garcia, "Evolutionary fuzzy rule-based methods for monotonic classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1376-1390, 2017.

[43] S. T. Li, and C. C. Chen, "A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1713-1727, 2014.

[44] Q. Hu, D. Yu, and M. Guo, "Fuzzy preference based rough sets," *Information Sciences*, vol. 180, no. 10, pp. 2003-2022, 2010.

[45] H. Levy, "Stochastic dominance: Investment decision making under uncertainty", Springer, 2015.
[46] C. C. Chen, and S. T. Li, "Credit rating with a monotonicity-constrained support vector machine model," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7235-7247, 2014.
[47] W. Verbeke, D. Martens, and B. Baesens, "RULEM: A novel heuristic rule learning approach for ordinal classification with monotonicity constraints," *Applied Soft Computing*, vol. 60, pp. 858-873, 2017.
[48] A. Ben-David, "Monotonicity maintenance in information-theoretic machine learning algorithms," *Machine Learning*, vol. 19, no. 1, pp. 29-43, 1995.
[49] J.R. Cano, P.A. Gutierrez, B. Krawczyk, M. Wozniak, and S. Garcia, "Monotonic classification: an overview on algorithms, performance measures and data sets," *Neurocomputing*, vol. 341, pp. 168-182, 2019.
[50] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
[51] J. Alcala-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. Garcia, L. Sanchez, and F. Herrera,"KEEL Data-Mining Software Tool: Data Set Repository," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 2-3, 2011.
[52] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675-701, 1937.
[53] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86-92, 1940.
[54] P.B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

Hong Zhu received her B.Sc. and M.Sc. degrees from Hebei University, Baoding, China, in 2012 and 2015, respectively, and received her Ph.D. degree in Computer Technology and Its Applications from Macau University of Science and Technology in 2018. Currently, She is a postdoctoral fellow at Shenzhen University. Her main research interests include decision tree, neural networks, cost-sensitive learning and fuzzy systems.

XiZhao Wang Ph.D. (1998), Professor (1998), IEEE Fellow (2012), CAAI Fellow (2017), Editor-in-Chief of Springer Journal Machine Learning and Cybernetics (2010), Deputy chair of CAAI machine learning committee (2012), and knowledge engineering committee (2013), overseas high-level (peacock B class) talent of Shenzhen city (2015), prizewinner of First-Class Award of Natural Science Advances of Hebei Province (2007), and Model Teacher of China (2009).

Prof. Wang received his doctor degree in computer science from Harbin Institute of Technology in September 1998. From 1998 to 2001 Prof. Wang worked at Department of Computing in Hong Kong Polytechnic University as a research fellow. From October 2000 to March 2014 Prof. Wang served in Hebei University as the Dean of school of Mathematics and Computer Sciences. From October 2007 to March 2014 Prof. Wang was the founding Director of Key Lab. on Machine Learning and Computational Intelligence in Hebei Province, China. From September to November in 2013, Prof. Wang was a visiting professor of Canada Simon Fraser University. From December 2013 to January 2014, Prof. Wang was a visiting professor of University of Alberta in Canada. From July to September in 2014, Prof. Wang was a visiting professor of Australia New South Wales University at Canberra. Since March 2014 to now Prof. Wang has moved to college of computer science and software engineering in Shenzhen University as a professor and a director of Big Data Institute.

Prof. Wang's main research interest is machine learning and uncertainty information processing including inductive learning with fuzzy representation, approximate reasoning and expert systems, neural networks and their sensitivity analysis, statistical learning theory, fuzzy measures and fuzzy integrals, random weight network, and the recent topic: machine learning theories and methodologies in Big-Data environment.

Ran Wang received the B.Eng. degree in computer science from the College of Information Science and Technology, Beijing Forestry University, Beijing, China, in 2009, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2014. From 2014 to 2016, she was a Post-Doctoral Researcher with the Department of Computer Science, City University of Hong Kong. She is currently an Associate Professor with the College of Mathematics and Statistics, Shenzhen University, Shenzhen, China. Her current research interests include pattern recognition, machine learning, fuzzy sets and fuzzy logic, and their related applications.