Similarity-based and Iterative Label Noise Filters for Monotonic Classification

José-Ramón Cano
Dept. of Computer Science,
EPS of Linares, University of Jaén,
Linares 23700, Jaén, Spain
jrcano@ujaen.es

Julián Luengo
Dept. of Computer Science and AI.
University of Granada,
18071 Granada, Spain
julianlm@decsai.ugr.es

Salvador García
Dept. of Computer Science and AI.
University of Granada,
18071 Granada, Spain
salvagl@decsai.ugr.es

Abstract

Monotonic ordinal classification has received an increasing interest in the latest years. Building monotone models from these problems usually requires datasets that verify monotonic relationships among the samples. When the monotonic relationships are not met, changing the labels may be a viable option, but the risk is high: wrong label changes would completely change the information contained in the data. In this work, we tackle the construction of monotone datasets by removing the wrong or noisy examples that violate monotonicity restrictions. We propose two monotonic noise filtering algorithms to preprocess the ordinal datasets and improve the monotonic relations between instances. The experiments are carried out over eleven ordinal datasets, showing that the application of the proposed filters improve the prediction capabilities over different levels of noise.

1. Introduction

The classification with monotonicity constraints, also known as monotonic classification [1] or isotonic classification [2], is an ordinal classification problem [3] where a monotonic restriction is present. In monotonic classification, a higher value of an input feature, without varying other values, should not decrease its class assignment. The monotonicity of relations between the dependent and explanatory variables is very usual as a prior knowledge form in data classification [4]. Several monotonic classification approaches have been proposed in the specialized literature. They include classification trees and rule induction [5, 6, 7, 8, 9, 10], neural networks [11, 12], instance-based learning [1, 13, 14] and hybrid [15, 16]. Some of them expect the training set to be purely monotone to work correctly. Other classifiers are more permissive with non-monotonic data sets, but they may or may not guarantee the monotonic predictions.

Monotonic classification has been already tackled by means of soft-computing based approaches [17, 18, 19]. As such, these approaches are not unrelated with the problems that arise in the classification paradigm. Among them, noise is specially problematic since it obscures the relationship between the features of an instance and its class [20, 21]. Among other consequences, noise can adversely impact the classification performances of induced classifiers. Some authors have already created noise-robust classifiers under the soft-computing paradigm [22, 23], but adapting all classifier types is unfeasible. By extrapolating to monotonic classification, noise also alters the monotonicity constraints present in the data.

In order to test the performance of monotonic classifiers, the usual trend is to generate data sets that fulfill the monotonicity constraint. The main argument is that models trained on monotonic data sets often have better predictive performance than models trained on the original data [24]. Monotonic data sets can be created by generating artificial data [25] or by relabeling of real data [14, 26, 27].

This paper proposes a different generic approach to deal with the construction of monotonic models by any classifier. In previous works, facing examples that violate the monotonic constraints was tackled by repairing such violations, generating a conformal dataset. This process of "switching" instances [28] has been deeper explored in highly imbalanced datasets, where the overlapping makes the relabeling process even more challenging [29]. Although appealing, the switching may induce a bias in the data when not correctly performed. As alternative, we will consider the examples that do not fulfill the monotonic constraints as noisy examples. This is a novel approach in which we propose the application of noise filtering algorithms in a preprocessing stage for monotonic ordinal classification. The use of preprocessing to generate quality data is a common procedure, even involving soft-computing based approaches to this task [30].

In this work, two classical noise filtering algorithms have been readjusted to this domain. The algorithms

URI: https://hdl.handle.net/10125/63949
978-0-9981331-3-3

considered are the Edited Nearest Neighbor (ENN [31]) and the Iterative Partition Filtering (IPF [32]). The monotonic version of them are noted as MENN and MIPF, respectively. Both filters are reworked to detect non-monotonic examples by signaling them as noisy examples to be detected and removed. The remaining samples are used as input in well-known monotonic classifiers. Our objective is to analyze their performances in comparison with leaving the conflictive instances, showing the safety and suitability of these proposed methods. In order to do so, several datasets are used and different level of noise is introduced in the data. These controlled amount of induced noise will help us to better evaluate the evolution in the classifiers' performance when applying our methods.

The rest of this manuscript is organized as follows. In Section 2 we present the monotonic classification problem and briefly describe the monotonic classifiers used in the study. Section 3 is devoted to describe the label noise problem and the filtering algorithms proposed. Section 4 describes the experimental framework. Section 5 studies the use of monotonic filters to remove non-monotonic instances. Finally, Section 6 concludes the paper.

2. Monotonic classification

The property of monotonicity commonly appears in domains of our lives such as natural sciences, natural language, game theory or economics [4, 33]. For instance, the case of bankruptcy prediction in companies, where appropriate actions can be taken in time considering the information based on financial indicators taken from their annual reports. The monotonicity is clearly present in the comparison of two companies where one dominates the other on all financial indicators, which supposes that the overall evaluation of the second cannot be higher than the evaluation of the first. This rule could be applied to the credit rating strategy used by banks [34] as well as for the bankruptcy prediction strategy.

The monotonic ordinal data can be defined as following. Let *D* be a data set with *f* ordinal attributes $A_1, ..., A_f$ and one output class attribute *Y* having *c* possible ordinal values. The data set consists of *n* examples x_i. A partial ordering $\preceq$ on *D* is defined as

$$x \preceq x' \Leftrightarrow A_j(x) \leq A_j(x'), \forall j = 1, ..., f \qquad (1)$$

Two examples *x* and *x'* in space *D* are *comparable* if either $x \preceq x'$ or $x' \preceq x$, otherwise *x* and *x'* are *incomparable*. Two examples *x* and *x'* are *identical* if *x* = *x'* and *non-identical* if $x \neq x'$;

Considering this notation, we denote a pair of comparable examples (*x*,*x'*) *monotone* if

$$x \preceq x' \wedge x \neq x' \wedge Y(x) \leq Y(x') \qquad (2)$$

or

$$x = x' \wedge Y(x) = Y(x') \qquad (3)$$

A data set *D* with *n* examples is monotone if all possible pairs of examples are either monotone or incomparable.

3. Label noise filtering in monotonic classification

As in standard classification, the correct labeling of the training set is crucial to obtain accurate models that will correctly predict new examples. Most classification algorithms assume that the labeling in the data is correct and follows the underlying distribution without any disturbances. However, in the real world this assumption is naive. In classification, the noise may affect the data registration of the input attributes or the labeling process made automatically or by an expert [20]. If the noise has affected the input attributes, it is usually named *attribute noise*. On the other hand, *class noise* or *label noise* means that the noise has corrupted the correct label of some instances. Some studies have analyzed the impact of these types of noise, indicating that class noise is more harmful than attribute noise, as the bias introduced is greater. For this reason, in this work we will focus on label noise and the different ways to tackle its greater impact.

In this work, we focus our attention on the approaches to deal with class noise in the specialized literature for classification, which are often grouped into three families.

1. We can adapt the classifiers to take into account noise and become *robust learners*, less influenced by noise. Some classifiers were designed as robust against noise from their very conception, as Kernel Logistic Regression [35], a robust SVM [36] or robust Fisher discriminant analysis learners [37, 38] just to mention a few. We can find adaptations of noise sensitive learners as AdaBoost [39] to become robust learners in recent proposals [40].

2. While robust learners can deal with noisy data directly, not all classifiers have been modified to be robust. The solutions designed to make a classifier robust cannot be easily extrapolated to other learners. Thus, a popular option is to apply

noise filtering methods [41, 42, 43], which work at data level before the classifier is applied. Noise filters aim to detect and eliminate the corrupted instances that would hinder the model built by the classifier, enabling any classifier model to work with noisy datasets.

3. Noise filters are a popular option due to their easy application and independence with the classifier. However, the cost of eliminating instances cannot be disregarded, especially in highly noisy problems. In these cases, the amount of instances eliminated would be high enough to produce a data shift in the class borders. An optimal preprocessing technique would recover the noisy instances, relabeling them with their true label. This family of techniques are known as *data correcting methods* [44, 45].

Frénay and Verleysen [20] point out that filtering noisy instances is more efficient than correcting them [46, 47]. Thus, we will focus on filtering approaches for noise.

We propose the application of data preprocessing techniques to the original data, which have been successfully applied in the past in similar domains [48, 49, 50, 51, 52]. In particular, we consider readjusted noise filtering algorithms to tackle the monotonic domain [20]. These methods identify and remove some of the examples belonging to the data set presenting a negative effect to the fulfillment of the monotonicity constraints, keeping unaltered the rest of the information held in the original data set. Our selection of popular noise filters are justified based on those that obtained the best performances in other learning domains, such as standard classification [53], imbalanced classification [54] or semi-supervised classification [55].

The family of filters adapted can be considered as "meta-algorithms" that exploit partial knowledge extracted from base filters to decide whether to filter an example or not. Thus, as long as there exists an equivalent monotonic classifier for the original base method used, the adaptation of a noise filter for standard classification could be possible. However, this procedure involves taking into account new aspects that were not present in standard classification. We have considered the violation of monotonicity constraints as additional information that the filter can use to control the noise removal process. Next, we describe the proposed filters.

3.1. Monotonic Edited Nearest Neighbor

The Monotonic Edited Nearest Neighbor (MENN) evolves from the classical Edited Nearest Neighbor algorithm [31]. Each instance in the training set is removed if it does not agree with the majority of its k monotonic nearest neighbors. As decrescent algorithm, it is not affected by the order of presentation of instances. The use of the monotonic k-nearest neighbors instead of the classical k-nearest neighbors rule constitutes the adaptation of this algorithm to the monotonic scope. The pseudo-code of MENN is presented in Algorithm 1.

Algorithm 1 MENN algorithm.

```
function MENN($T$ - training data, $k$ - number of nearest neighbor)
    initialize: $S = T$
    for all $x \in S$ do
        $X' = \emptyset$
        $y_{min} = max\{class(x') | x' \in T \wedge x' \leq x\}$
        $y_{max} = min\{class(x') | x' \in T \wedge x \leq x'\}$
        for $i = 1$ to $k$ do
            Find $x'_i \in T$ s.t. $x \neq x'_i$ and $||x - x'_i|| = \min_{x^j \in (T \setminus X')} ||x - x^j||$ and $class(x'_i) \in [y_{min}, y_{max}]$
            $X' = X' \cup \{x'_i\}$
        end for
        if $class(x) \neq majorityClass(X')$ then
            $S = S \setminus \{x\}$
        end if
    end for
    return $S$
end function
```

3.2. Monotonic Iterative Partition Filtering

The Monotonic Iterative Partition Filtering (MIPF) is a global noise filter which applies a classifier to several subsets of the training data set to detect possible noisy examples. It removes noisy instances in multiple iterations until the number of identified noisy examples, for a number of consecutive iterations, is less than a percentage of the size of the original training data set [32]. The classifier embedded in the classic Iterative Partition Filtering algorithm is the C4.5 [56]. For our purposes, we replace it with the ordinal interpretation of C4.5 [57]. It is worth mentioning that the ordinal C4.5 does not produce monotonic models but ensures ordinal classification. MIPF is described in Algorithm 2.

Algorithm 2 MIPF algorithm.

```
function MIPF($T$ - dataset with Monotonic Violations, $\Gamma$ - number of subsets, $y$ - amount of good data to be eliminated in each step, $p$ - minimum percentage of noisy instances to continue)
    initialize: $T_G = \{\}$, $F$ = Ordinal C4.5
    repeat
        Split the training data set $T$ into $T_i, i = 1 \ldots \Gamma$ equal sized subsets
        for each subset $T_i$ do
            Use $\{T_j, j \neq i\}$ to train $F$ resulting in $F^i$ different classifiers
        end for
        $D_N = \{\}, D_G = \{\}$
        for each instance $t$ in $T$ do
            Classify $t$ with every $F^i$
            if $t$ is voted as noisy then
                $D_N = D_N \cup t$
            end if
        end for
        $D_G = \{t_l \in T | t_l \notin D_N; l = 1, ..., y\}$
        $T_G = T_G \cup D_G$
        $T = T - \{D_N \cup D_G\}$
    until $|D_N| < p \cdot |T|$
    return $T \cup T_G$
end function
```

4. Experimental framework

In this section, we present the experimental framework developed to analyze the proposal of application of three well-known noise filtering algorithms readjusted to work in this domain.

The study includes eleven data sets whose class attribute can be expressed as ordinal and presents monotonic relationship with the features. Four data sets are actual classical ordinal classification data sets commonly used in this field (Era, Esl, Lev and Swd [1]). The other 7 are regression data sets whose class attribute was discretized into 4 categorical values, maintaining the class distribution balanced. All of the eleven data sets are classical problems used in the classification scope and extracted from the UCI [58] and KEEL[1] repositories [59, 60].

In order to evaluate the performance of the different approaches with different amounts of monotonic violations, we have generated three corrupted versions of each dataset. These altered versions are created by changing a $noise\%$ of instances by relabeling them with a new class label. The new label can only be the precedent or the following one, thus generating realistic disorders in terms of monotonic violations. Such a noise introduction scheme follows the NAR mechanism as described in [20], in which the true label has influence in the observed (and possibly corrupted) label. We have applied $noise\% = 10\%, 20\%$ and 30% levels only in the training partitions to simulate from low to highly noisy scenarios. Since the true labels are known, we can later examine the performance of the preprocessing approaches in term of well and wrongly filtered instances.

[1]http://www.keel.es/datasets.php

Table 1: Description of the 11 data sets used in the study.

Data set	Ins.	At.	Cl.
Balance	625	4	3
BostonHousing	506	12	4
Car	1728	6	4
Era	1000	4	9
Esl	488	4	9
Lev	1000	4	5
CPU	209	6	4
QualitativeBankruptcy	250	6	2
Swd	1000	10	4
WindsorHousing	545	11	2
Wisconsin	683	9	2

All the algorithms are run using run a 10-fold cross validation scheme (10-fcv). For all the training partitions, three different noisy versions are generated (with different seeds) for each noise level. Therefore, we obtain 30 executions per dataset and noise level.

Table 1 shows the names of the data sets, their number of instances, attributes, and classes. In this paper, instances having missing values have been ignored.

In order to compare the four monotonic filters, we will use four metrics commonly employed in the monotonic classification field. They are listed as follows:

- Accuracy (ACC) is computed as the percentage of correctly classified instances. Is a traditional measure in the classification topic that we include as a reference metric.

- Non-Monotonicity Index 1 (NMI1) [61], is defined as the number of clash-pairs divided by the total number of pairs of examples in the data set:

$$\text{NMI1} = \frac{1}{n(n-1)} \sum_{x \in D} \text{NClash}(x) \qquad (4)$$

where x is an example from the data set D.

NClash(x) is the number of examples from D that do not meet the monotonicity restrictions (or clash) with x and n is the number of instances in D.

- Non-Monotonicity Index 2 (NMI2) [27], is defined as the number of non-monotone examples divided by the total number of examples:

$$\text{NMI2} = \frac{1}{n}\sum_{x\in D}\text{Clash}(x) \qquad (5)$$

where Clash(x) = 1 if x clashes with some examples in D, and 0 otherwise. If Clash(x) = 1, x is called a non-monotone example.

- Non-Comparable. This is a metric related to the number of pairs of non comparable instances in the data set. Two instances x and x' are non-comparable if they do no satisfy $x \preceq x' \wedge x \neq x'$. This measure is also considered due to the fact that for some monotonic classifiers, it is harder to construct accurate models agreeing the number of non-comparable pairs raises.

- *Size* of the subset selected using the noise filtering algorithms. We include it to analyze the noise removing capabilities of each method.

Table 2 shows the parameter configuration for all the methods used. Since we want to focus on the effect of the filtering step, the classifiers' parameters have been fixed to those recommended by their respective authors in the original publications. We have also found that such parameter values have been used in several monotonic classification publications in the state-of-the-art.

We have experimentally adjusted the values for the parameters of the proposed filters, starting with the typical values used in standard classification found in other publications about noise filtering. From that starting point, we have applied a grid search, ranging from $k = 1$ to $k = 10$ for MENN and *numberPartitions*$=$ 2 to *numberPartitions*$=$ 10 for MIPF. The criteria for such an optimization has been the average accuracy obtained by all the classifiers across all datasets. The base classifier's parameters are similar to those used in the classification step.

5. Experimental results

This section is devoted to analyze the results obtained, providing a summary of results including graphics and statistical outcome. We present the results considering two perspectives:

Table 2: Parameters considered for the algorithms compared.

Algorithm	Parameters
MENN	$k = 3$
MIPF	numberPartitions = 5, consensus filter confidence = 0.25, 1 items per leaf
MkNN	k = 3, distance = euclidean
OLM	modeResolution = conservative modeClassification = conservative
OSDL	classificationType = media, balanced = No weighted = No, tuneInterpolationParameter = No, lowerBound = 0, upperBound = 1 interpolationParameter = 0.5, interpolationStepSize = 10
MID	confidence = 0.25, 2 items per leaf, R = 1

1. We compare the behavior of the algorithms using Accuracy. In addition to the noise removal algorithms, we include the results obtained using the original data sets as input and data sets after relabeling the training partitions (keeping the tests partitions as they are). This analysis is carried out in Section 5.1.

2. We study the algorithms using different metrics to study how the filtering and relabeling process affects the monotonic properties of the datasets: NMI1, NMI2, *Non-Comparable* and *Size*. Section 5.2 is devoted to study such measures.

5.1. Performance measures

In this section, we provide the accuracy and MAE results for all the classifiers and preprocessing strategies described above. These measures are computed over the test partitions after applying MENN and MIPF. We also include the absence of preprocessing, named as *No preprocessing*, to show the consequences of leaving an increasing amount of monotonic violations in the training set.

Table 3 shows the averaged accuracy values for all the datasets. The algorithm with the best value is stressed in bold. As we can appreciate, MIPF is the best performing technique on average, except for OSDL at 30% noise.

On the right side of Table 3 the rankings of Friedman's test are shown. Friedman's test reject the null-hypothesis in all cases. MIPF is again the best ranked algorithm, except for OSDL when we consider the original datasets. When Holm's post-hoc indicates a p-value < 0.05 the cell of the compared algorithm is grayed, whereas a p-value < 0.1 is indicated by underling the rank of the compared algorithm. The large amounts of shaded cells supports the choice of MIPF as the best performing algorithm, while no preprocessing

Table 3: Average accuracy results for the filters and relabel with all the classifiers and each noise level. Best values are stressed in bold. Friedman rankings and Holm's post-hoc test $p - values$ are also provided.

		Accuracy averages				Friedman's rankings			
	Preprocessing	0% (Original)	10%	20%	30%	0% (Original)	10%	20%	30%
MKNN	No preprocessing	0.69	0.65	0.60	0.55	2.14	2.36	2.55	2.45
	MENN	0.61	0.58	0.54	0.51	2.45	2.55	2.36	2.45
	MIPF	**0.71**	**0.70**	**0.69**	**0.65**	**1.41**	**1.41**	**1.01**	**1.09**
MID	No preprocessing	0.72	0.69	0.65	0.61	1.86	2.05	2.09	1.91
	MENN	0.60	0.58	0.55	0.51	2.73	2.73	2.55	2.82
	MIPF	**0.73**	**0.71**	**0.69**	**0.65**	**1.41**	**1.23**	**1.36**	**1.27**
OLM	No preprocessing	0.56	0.50	0.49	0.47	2.5	2.73	2.36	2.36
	MENN	0.54	0.50	0.47	0.46	1.91	2.14	2.45	2.45
	MIPF	**0.60**	**0.59**	**0.57**	**0.55**	**1.59**	**1.68**	**1.18**	**1.18**
OSDL	No preprocessing	0.59	0.46	0.44	0.40	**1.82**	2.32	2.27	2.36
	MENN	0.58	0.52	0.49	0.47	2.18	2.41	2.27	2.36
	MIPF	**0.59**	**0.58**	**0.56**	**0.54**	2.00	**1.27**	**1.45**	**1.27**

is the only alternative that it is not statistically different for 0% or MID in some cases.

In summary, the application of a noise filtering stage based on MIPF is beneficial for all the classifiers considered. In particular, the combination of MID and MIPF seems to be the most robust combination, showing the best accuracy values across all noise levels. While some classifiers, as OSDL, are less affected by a previous preprocessing stage based on noise filtering, sensitive classifiers as MKNN take more advantage from noise filtering comparing No preprocessing against any other filtering technique. Nevertheless, MENN is not the best choice in noisy monotonic classification, indicating that instance selection algorithms may not perform well acting as noisy filters. In the next section we will try to get some insights on why MIPF is able to attain better performance than the compared algorithms and why MENN performs poorly.

5.2. Monotonicity metrics

Table 4 is dedicated to the monotonic metrics considered. The table is structured into five columns, first for the name of the algorithm and others for the noise levels studied. The results, grouped by metric, are the average metric values of the eleven data sets for the noise filtering algorithms and No preprocessing. The best result is stressed in bold.

All the metrics results in Table 4 are intrinsically related, but NMI1, NMI2 and *Non-Comparable* are specific for the monotonic classification problem. Observing *Size*, the highest reduction rates corresponds to MENN. Please recall that MENN was not performing as well as it should. Thus, we may deduce that MENN is eliminating too many instances, due to its instance selection nature. MIPF is more balanced, removing mostly the noisy instances and keeping more information in the dataset to be exploited.

Average NMI1 and NMI2 indicate the grade of monotonicity in a data set: we must take as reference value NMI1 and NMI2 values for *No preprocessing* at 0% noise level. It is clear that in original data set the values are higher, while the monotonic noise removal techniques introduced in this paper reduce them. While MENN obtains the best results in NMI1, NMI2 shows MPIF as the best performing algorithm. NMI1 is an absolute measure and MENN is the technique that removes more instances, thus decreasing the number of monotonic violations in absolute numbers. Since NMI2 is a relative measure, where the number of examples is taken into account, MIPF is showing less violation per instance. As a result, we should consider NMI2 as a better indicator of a successful noise filter in monotonic classification problems.

Finally, we also want to pay attention to *Non-comparable* values, as they indicate the amount of monotonic violations that remains in the dataset after applying the different preprocessing techniques. The case of MENN is interesting, as it aims to reduce the number of violations as much as possible, and thus achieving the best results for *Non-comparable*. Since MIPF are the best performing algorithm in terms of accuracy, we may conclude that extreme behaviors as

Table 4: Average of the monotonicity metrics with respect to monotonic noise filtering algorithms.

	Preprocessing	0% (Original)	10%	20%	30%
NMI1	No preprocessing	0.021	0.024	0.026	0.028
	MENN	**0.005**	**0.007**	**0.008**	**0.009**
	MIPF	0.017	0.018	0.019	0.021
NMI2	No preprocessing	0.649	0.822	0.858	0.876
	MENN	0.385	0.467	0.495	0.524
	MIPF	**0.368**	**0.405**	**0.455**	**0.487**
Non-Comparable	No preprocessing	63265.24	67441.60	70829.14	74165.93
	MENN	**25806.34**	**10088.76**	**8691.59**	**8729.69**
	MIPF	38383.60	33834.46	30491.37	28302.81
Size	No preprocessing	657.41	657.41	657.41	657.41
	MENN	**372.39**	**277.89**	**257.80**	**249.65**
	MIPF	537.28	503.53	477.15	453.03

those shown by MENN are not desirable: while the former does not solve most of the violations, the latter tends to remove too many instances to eliminate the violations and altering the information contained in the dataset.

At this point, MENN is the preprocessing technique that is able to obtain the lowest amounts of non-comparable instances. However, we observed in Section 5.1 that MENN is not the best performing algorithm. Since MENN also creates the most reduced datasets in terms of size, we may conclude that MENN is removing too many instances, which would lead to less violations of monotonic restrictions as shown by NMI1, NMI2 and *Non-comparable* values. This excessive removal will create an information loss in the dataset that penalizes the model obtained and thus showing poor performance in Accuracy and MAE values.

Finally, we must point out that the usage of monotonicity metrics alone cannot describe the ability of noise preprocessing algorithms in monotonic classification, as they can be largely minimized by removing too many instances as MENN does. Maintaining a good proportion of clean instances is crucial to enable the classifiers to obtain generalizable models. MIPF is the best approach analyzed to this respect.

6. Conclusions

In this paper we have proposed the use of noise filtering algorithms as a preprocessing stage to decrease the monotonicity violations present in the original data. We have analyzed two noise removal algorithms, adapted to the monotonic domain, using different prediction rates and metrics over a number of ordinal data sets, coming from standard classification and regression problems.

Our results show that monotonic noise removal algorithms are able to remove instances which negatively affect to the monotonicity of real data, altering the lowest possible the concepts represented in the original data and improving the efficiency and efficacy of the monotone classifiers. Among the two filtering methods, MIPF can preserve and even to improve the prediction performances offered by classical monotonic classifiers such as MkNN, OLM, OSDL and MID.

We have also shown that monotonicity metrics cannot describe what constitutes a good filtering, as they can be biased by removing too many instances. While we have shown that filtering can greatly help to diminish the impact of noisy instances in monotonic classification, there is still promising options to explore: correct reparation of instance can greatly help to improve even further the results of this work. Since the monotonicity metrics can deceive the noise filters, other measures can be designed to avoid the greedy removal of preprocessing techniques.

Acknowledgment

This work was supported by TIN2017-89517-P, by the Spanish ”Ministerio de Economía y Competitividad” and by ”Fondo Europeo de Desarrollo Regional” (FEDER) under Project TEC2015-69496-R and the Project BigDaP-TOOLS - Ayudas Fundación BBVA a Equipos de Investigación Científica 2016.

References

[1] A. Ben-David, L. Serling, and Y. Pao, "Learning and classification of monotonic ordinal concepts," *Computational Intelligence*, vol. 5, pp. 45–49, 1989.

[2] B. Malar and R. Nadarajan, "Evolutionary isotonic separation for classification: theory and experiments," *Knowledge and Information Systems*, vol. 37, no. 3, pp. 531–553, 2013.

[3] K. Antoniuk, V. Franc, and V. Hlaváč, "V-shaped interval insensitive loss for ordinal classification," *Machine Learning*, vol. 103, no. 2, pp. 261–283, 2016.

[4] W. Kotlowski and R. Slowiński, "On nonparametric ordinal classification with monotonicity constraints," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2576–2589, 2013.

[5] A. Ben-David, "Monotonicity maintenance in information theoretic machine learning algorithms," *Machine Learning*, vol. 19, pp. 29–43, 1995.

[6] R. Potharst and J. Bioch, "Decision trees for ordinal classification," *Intelligent Data Analysis*, vol. 4, pp. 97–111, 2000.

[7] K. Cao-Van and B. De Baets, "Growing decision trees in an ordinal setting," *International Journal of Intelligent Systems*, vol. 18, pp. 733–750, 2003.

[8] W. Kotłowski and R. Słowiński, "Rule learning with monotonicity constraints," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 537–544, ACM, 2009.

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

[10] J. Alcalá-Fdez, R. Alcalá, S. González, Y. Nojima, and S. García, "Evolutionary fuzzy rule-based methods for monotonic classification," *IEEE Transactions on Fuzzy Systems, in press. DOI: 10.1109/TFUZZ.2017.2718491*, 2017.

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

[12] H. Zhu, E. C. Tsang, X.-Z. Wang, and R. A. R. Ashfaq, "Monotonic classification extreme learning machine," *Neurocomputing*, vol. 225, pp. 205–213, 2017.

[13] 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, pp. 115–142, 2008.

[14] W. Duivesteijn and A. Feelders, "Nearest neighbour classification with monotonicity constraints.," in *ECML/PKDD (1)*, vol. 5211 of *Lecture Notes in Computer Science*, pp. 301–316, Springer, 2008.

[15] J. García, A. M. AlBar, N. R. Aljohani, J.-R. Cano, and S. García, "Hyperrectangles selection for monotonic classification by using evolutionary algorithms," *International Journal of Computational Intelligence Systems*, vol. 9, no. 1, pp. 184–201, 2016.

[16] J. García, H. M. Fardoun, D. M. Alghazzawi, J.-R. Cano, and S. García, "Mongel: monotonic nested generalized exemplar learning," *Pattern Analysis and Applications*, vol. 20, no. 2, pp. 441–452, 2017.

[17] P. Pattaraintakorn, N. Cercone, and K. Naruedomkul, "Rule learning: Ordinal prediction based on rough sets and soft-computing," *Applied Mathematics Letters*, vol. 19, no. 12, pp. 1300–1307, 2006.

[18] C. Moewes and R. Kruse, "Evolutionary fuzzy rules for ordinal binary classification with monotonicity constraints," in *Soft Computing: State of the Art Theory and Novel Applications*, pp. 105–112, Springer, 2013.

[19] J. Alcalá-Fdez, R. Alcalá, S. González, Y. Nojima, and S. García, "Evolutionary fuzzy rule-based methods for monotonic classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1376–1390, 2017.

[20] B. Frénay 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, 2014.

[21] J. A. Sáez, M. Galar, J. Luengo, and F. Herrera, "Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition," *Knowledge and information systems*, vol. 38, no. 1, pp. 179–206, 2014.

[22] R. Jensen and C. Cornelis, "Fuzzy-rough nearest neighbour classification and prediction," *Theoretical Computer Science*, vol. 412, no. 42, pp. 5871–5884, 2011.

[23] W. An and M. Liang, "Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises," *Neurocomputing*, vol. 110, pp. 101–110, 2013.

[24] A. Feelders, "Monotone relabeling in ordinal classification.," in *IEEE International Conference on Data Mining (ICDM)*, pp. 803–808, 2010.

[25] R. Potharst, A. Ben-David, and M. C. van Wezel, "Two algorithms for generating structured and unstructured monotone ordinal data sets," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 4-5, pp. 491–496, 2009.

[26] M. Rademaker, B. De Baets, and H. De Meyer, "Optimal monotone relabelling of partially non-monotone ordinal data," *Optimization Methods and Software*, vol. 27, no. 1, pp. 17–31, 2012.

[27] I. Milstein, A. Ben-David, and R. Potharst, "Generating noisy monotone ordinal datasets," *Artificial Intelligence Research*, vol. 3, no. 1, pp. 30–37, 2014.

[28] G. Martínez-Muñoz and A. Suárez, "Switching class labels to generate classification ensembles," *Pattern Recognition*, vol. 38, no. 10, pp. 1483–1494, 2005.

[29] S. Gónzalez, S. García, M. Lázaro, A. R. Figueiras-Vidal, and F. Herrera, "Class switching according to nearest enemy distance for learning from highly imbalanced data-sets," *Pattern Recognition*, vol. 70, pp. 12–24, 2017.

[30] N. Verbiest, E. Ramentol, C. Cornelis, and F. Herrera, "Preprocessing noisy imbalanced datasets using smote enhanced with fuzzy rough prototype selection," *Applied Soft Computing*, vol. 22, pp. 511 – 517, 2014.

[31] D. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.

[32] T. Khoshgoftaar and P. Rebours, "Improving software quality prediction by noise filtering techniques," *Journal of Computer Science and Technology*, vol. 22, pp. 387–396, 2007.

[33] T. Tran, D. Phung, W. Luo, and S. Venkatesh, "Stabilized sparse ordinal regression for medical risk stratification," *Knowledge and Information Systems*, vol. 43, no. 3, pp. 555–582, 2015.

[34] 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.

[35] J. Bootkrajang and A. Kabán, "Learning kernel logistic regression in the presence of class label noise," *Pattern Recognition*, vol. 47, no. 11, pp. 3641–3655, 2014.

[36] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.

[37] N. D. Lawrence and B. Schölkopf, "Estimating a kernel fisher discriminant in the presence of label noise," in *ICML*, vol. 1, pp. 306–313, 2001.

[38] C. Bouveyron and S. Girard, "Robust supervised classification with mixture models: Learning from data with uncertain labels," *Pattern Recognition*, vol. 42, no. 11, pp. 2649–2658, 2009.

[39] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[40] Q. Miao, Y. Cao, G. Xia, M. Gong, J. Liu, and J. Song, "Rboost: label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 11, pp. 2216–2228, 2016.

[41] C. E. Brodley and M. A. Friedl, "Identifying Mislabeled Training Data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[42] T. M. Khoshgoftaar and P. Rebours, "Improving software quality prediction by noise filtering techniques," *Journal of Computer Science and Technology*, vol. 22, pp. 387–396, 2007.

[43] S. Verbaeten and A. V. Assche, "Ensemble methods for noise elimination in classification problems," in *Fourth International Workshop on Multiple Classifier Systems*, pp. 317–325, Springer, 2003.

[44] C.-M. Teng, "Correcting Noisy Data," in *Proceedings of the Sixteenth International Conference on Machine Learning*, (San Francisco, CA, USA), pp. 239–248, Morgan Kaufmann Publishers, 1999.

[45] B. Nicholson, V. S. Sheng, and J. Zhang, "Label noise correction and application in crowdsourcing," *Expert Systems with Applications*, vol. 66, pp. 149–162, 2016.

[46] S. Cuendet, D. Z. Hakkani-TÃŒr, and E. Shriberg, "Automatic labeling inconsistencies detection and correction for sentence unit segmentation in conversational speech.," in *MLMI* (A. Popescu-Belis, S. Renals, and H. Bourlard, eds.), vol. 4892 of *Lecture Notes in Computer Science*, pp. 144–155, Springer, 2007.

[47] A. L. B. Miranda, L. P. F. Garcia, A. C. P. L. F. Carvalho, and A. C. Lorena, "Use of classification algorithms in noise detection and elimination," in *HAIS* (E. Corchado, X. Wu, E. Oja, Ã. Herrero, and B. Baruque, eds.), vol. 5572 of *Lecture Notes in Computer Science*, pp. 417–424, Springer, 2009.

[48] S. García, J. Luengo, and F. Herrera, "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining," *Knowledge-Based Systems*, vol. 98, pp. 1–29, 2016.

[49] S. García, J. Derrac, J.-R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 417–435, 2012.

[50] J.-R. Cano, S. García, and F. Herrera, "Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes," *Pattern Recognition Letters*, vol. 29, pp. 2156–2164, 2008.

[51] J.-R. Cano, F. Herrera, M. Lozano, and S. García, "Making CN2-SD subgroup discovery algorithm scalable to large size data sets using instance selection," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1949–1965, 2008.

[52] D. Han, Y. Hu, and G. Wang, "Uncertain graph classification based on extreme learning machine," *Cognitive Computation*, vol. 7, no. 3, pp. 346–358, 2015.

[53] J. A. Sáez, M. Galar, J. Luengo, and F. Herrera, "INFFC: an iterative class noise filter based on the fusion of classifiers with noise sensitivity control," *Information Fusion*, vol. 27, pp. 19–32, 2016.

[54] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering.," *Information Sciences*, pp. 184–203, 2015.

[55] I. Triguero, J. A. Sáez, J. Luengo, S. García, and F. Herrera, "On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification," *Neurocomputing*, vol. 132, pp. 30–41, 2014.

[56] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[57] E. Frank and M. Hall, "A simple approach to ordinal classification," *Lecture Notes in Computer Science*, vol. 2167, pp. 145–156, 2001.

[58] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[59] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 255-287, p. 11, 2010.

[60] I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesus, L. Sánchez, and F. Herrera., "KEEL 3.0: an open source software for multi-stage analysis in data mining," *International Journal of Computational Intelligence Systems*, vol. 10, pp. 1238–1249, 2017.

[61] H. Daniels and M. Velikova, "Derivation of monotone decision models from noisy data," *IEEE Transactions on Systems, Man and Cybernetics - Part C*, vol. 36, pp. 705–710, 2006.