Information Sciences 474 (2019) 187–204

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Chain based sampling for monotonic imbalanced classification

Sergio González[a,*], Salvador García[a], Sheng-Tun Li[b], Francisco Herrera[a,c]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain
[b] Department of Industrial and Information Management, National Cheng Kung University, Tainan 701, Taiwan ROC
[c] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 19 March 2018
Revised 18 July 2018
Accepted 26 September 2018
Available online 27 September 2018

Keywords:
Monotonic classification
Imbalanced classification
Sampling techniques
Preprocessing
Data mining

ABSTRACT

Classification with monotonic constraints arises from some ordinal real-life problems. In these real-life problems, it is common to find a big difference in the number of instances representing middle-ranked classes and the top classes, because the former usually represents the average or the normality, while the latter are the exceptional and uncommon. This is known as class imbalance problem, and it deteriorates the learning of those underrepresented classes. However, the traditional solutions cannot be applied to applications that require monotonic restrictions to be asserted. Since these were not designed to consider monotonic constraints, they compromise the monotonicity of the data-sets and the performance of the monotonic classifiers. In this paper, we propose a set of new sampling techniques to mitigate the imbalanced class distribution and, at the same time, maintain the monotonicity of the data-sets. These methods perform the sampling inside monotonic chains, sets of comparable instances, in order to preserve them and, as a result, the monotonicity. Five different approaches are redesigned based on famous under- and oversampling techniques and their standard and ordinal versions are compared with outstanding results.

1. Introduction

Ranking and evaluation of assets or even individuals are intrinsic characteristics of human nature. Hence, the presence of ordinal variables is common in tons of real-life data-sets. Credit rating [13,48], house ranking [50] and employee evaluation [6,37] are good examples of their presence in present-day applications.

These problems aim to determine the most valuable items according to their virtues, i. e. classification into ordinal labels according to ordinal attributes. Additionally, these applications usually require a monotonic restriction between the inputs and the class. That is, the class prediction of an individual should not decrease with a better value for a certain variable, fixing the remainder. Otherwise, an unfair evaluation of the individuals can be made. These classification problems with prior knowledge of the order relations between attributes and the class are known as classification with monotonicity constraints or monotonic classification [4]. Failure to respect these constraints are referred to as violations of monotonicity and must be avoided in the class decision of new samples.

When dealing with monotonic classification problems [4], we look for those examples that belong to the most remarkable class, with a higher value. It is reasonable to have fewer samples of really good and remarkable individuals than those

* Corresponding author.
E-mail addresses: sergiogvz@decsai.ugr.es (S. González), salvagl@decsai.ugr.es (S. García), stli@mail.ncku.edu.tw (S.-T. Li), herrera@decsai.ugr.es (F. Herrera).

https://doi.org/10.1016/j.ins.2018.09.062
0020-0255/

considered normal or average. For example, in the evaluation of future employees of a company, there will probably be fewer "excellent candidates" than "average candidates."

This difference in the number of representatives between classes has proven to cause a great loss of prediction accuracy in the minority classes [38,44]. This issue is known as a class imbalance problem or imbalanced classification. Multiple real-life applications present this problem, even those from non-standard classifications, such as Monotonic Classification [4,6,48] or Multi-task learning [36,37]. The majority of the monotonic problems considered in the literature suffers due to this issue. Therefore, the imbalance class distribution must be approached in the scope of monotonic classification.

Traditionally data level approaches [44] have been well accepted because they allow the use of a standard classifier after balancing the skewed training sets by under-/over- sampling. However, these techniques also have their own drawbacks. When using under-sampling, there is the risk of losing relevant information from the treated class. On the other hand, over-sampling can introduce noisy instances.

These approaches are not designed for monotonic classification [4] and do not take monotonic constraints into consideration. Due to this lack of awareness of monotonicity [4], these preprocessing techniques can severely deteriorate the monotonicity of the data-sets and reduce the performance of the classifiers. For example, the possible noisy instances generated by over-sampling could mean a greater damage in monotonic classification, because they may increase the number of monotonicity violations in the data-sets. The under-sampling techniques could remove important instances that determine the limit of the classes in term of monotonicity.

Therefore, new sampling approaches must be designed considering the monotonicity constraints. We propose new sampling techniques based on monotonic chains. In monotonic classification, a chain [35] is a set of comparable instances, that is, they can be sorted. These are very important assets of the classification carried out relevant methods such as KNN [16] and OSDL [34,35], because they determine the possible classes without monotonic violations for new instances. Our techniques perform the sampling using these chains and preserving, as much as possible, the monotonicity of the data-sets. Additionally, these methods take monotonic noise into consideration, in order to avoid instances that violate monotonicity during the sampling process. These differences with the traditional methods reduce the deterioration of monotonicity of sampled data-sets and maintain the improvement of the accuracy for minority classes.

To do so, we have put together a new scheme for applying both under- and over-sampling to monotonic imbalanced data-sets. This scheme consists of several good practices, related to the influence of monotonic violations and chains on sampling, that can be extended to the almost all the sampling techniques in the literature. This scheme has been implemented in five famous under- and over-sampling approaches of the State-of-the-Art of imbalanced classification: Random Under-Sampling (RUS), Random Over-Sampling (ROS), Synthetic Minority Oversampling TEchnique (SMOTE) [10], ADAptive SYNthetic sampling approach (ADASYN) [27] and Majority Weighted Minority Oversampling TEchnique (MWMOTE) [2].

Throughout this paper, two different empirical studies are carried out with exactly the same experimental framework. The first experiments test the selected sampling techniques in their standard and ordinal versions using 8 monotonic imbalanced sets which are very common in the literature. The majority are multi-class and can be considered highly imbalanced problems. The original and sampled data-sets are classified by five well-known classifiers. Two evaluation metrics are used: Macro Average Arithmetic (MAvA) [47] evaluates the prediction capability in multi-class imbalanced scenarios and Non-Monotonic Index (NMI) [4] determines the monotonicity of data-sets and predictions. The obtained results show empirically the deterioration of the monotonicity degree in data-sets caused by standard and ordinal sampling approaches.

Then, a second experimental study is performed following the same framework to analyze the behavior of new monotonic sampling techniques. The different predictions obtained are compared in terms of multi-class accuracy and monotonicity. The comparison shows the capacity of monotonicity preservation of the monotonic sampling techniques over the standard ones. The outcomes are corroborated by the use of non-parametric statistical tests: Friedman ranking test [20,25] and Bayesian Sign test [5].

This paper is organized as follows. In Section 2, we present the problems approached and their solutions: classification with monotonic constraints and class imbalance problem. Section 3 is devoted to setting up the bases to adapt sampling approaches to monotonic scenarios and explain in detail the chain based sampling techniques. In Section 4, the experimental framework used in the different empirical studies is presented. Section 5 recalls two experimental studies: an analysis on the impact of standard and ordinal sampling on monotonic classification and a comparison of the results achieved by monotonic sampling. Finally, in Section 6, the main conclusions of this study are given.

2. Background

In this section, we introduce the background knowledge of the problems addressed in this paper.

2.1. Monotonic classification

Monotonic classification, just as ordinal regression and/or classification, aims to predict an ordinal class label y for new sample x with ordinal attributes with the help of a labeled set, i.e. f: $x \rightarrow y$. In both problems, the classes $\mathcal{Y}$ are categories $\mathcal{Y} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_C\}$ with a problem imposed arrangement $\mathcal{L}_1 \prec \mathcal{L}_2 \prec \ldots \prec \mathcal{L}_C$. However, there is a big difference between both problems. Ordinal classification just focuses on minimizing the errors of predicted and real labels. Monotonic classification imposes monotonicity constraints between the input variables and predicted labels, that is, every instance x'

dominated by x should have a lower or the same assigned class $f(x')$ than x class label $f(x)$. Formally, $x \succeq x' \rightarrow f(x) \geq f(x')$ [30], where $x \succeq x' \leftrightarrow x_j \geq x'_j$, $j = 1, \ldots, A$.

Recently this problem has drawn the attention of data mining practitioners, who have designed monotonic classifiers based on distinct models, such as, instance-based learning [3,16,24,35], rules-based methods and decision trees [1,4,39,41], support vector machines [13,33], neural networks [19,31,52] and ensemble learning [15,26,45]. These monotonic classifiers avoid monotonicity violations in their predictions. And they can be pure, when their decisions are always monotonic, or partial, if they minimize their violations as much as possible. Some of these monotonic classifiers have to learn from monotonic data-sets in order to properly predict new samples. Only if all the pairs of example i, j of a training set D are monotonic, the data-set D is considered monotonic [3], i.e. $x_i \succeq x_j \rightarrow y_i \geq y_j, \forall x_i, x_j \in D$.

Even though these constraints are defined in the learning and prediction phases of the classification process, they have to be taken into consideration in preprocessing phases. Otherwise the monotonicity of the involved training set can be severely compromised. Few studies have been undertaken in this field [6,7,28]. For example, imbalanced classification in scenarios with monotonic constraints has not been explored, although it has been for ordinal classification [40,42].

2.2. The class imbalance problem

The class imbalance problem refers to a severe loss of classification accuracy of certain classes due to their under-representation in the training data-set [38]. That is, some classes have a lot less instances than others, which affects their identification by standard classifiers. These under-represented classes are known as minority or positives, whilst the rest are referred to as majority or negative classes. In many real-life applications, the most important classes are usually the most imbalanced. Therefore, the misclassification of these classes entails greater costs [12].

The inaccurate prediction of the minority classes mainly results from the generalization of the standard classifiers used to infer models and avoid over-fitting. The use of global performance measures, such as accuracy rate, also negatively affects the learning phase. Therefore, the difference in the number of representatives of each class has a great influence on this issue. However, there are other issues that aggravate this problem [22,38], such as noise, overlapping, lack of density and small disjuncts.

Many data scientists have shown great interest in the field and several approaches have been designed to deal with it. These can be categorized into the following three [38]:

- Data level approaches [10,44]: the affected data-sets are rebalanced by sampling. The equal representation between classes is reached by generating more examples for the minority class (over-sampling) [10], removing examples from the majority class (under-sampling) or both (hybrid methods).
- The cost-sensitive learning [11,32] considers the costs of the errors of the misclassification of the minority class into function minimization.
- The algorithmic approaches [14,46] are modifications of base learning algorithms in order to achieve better performance with imbalanced data-sets.

Data level approaches are the most popular solutions to the problem, since they enable the standard classification methods. Random Over- and Under-Sampling are the most basic procedures that randomly duplicate or remove examples from the minority or majority classes. Other more complex methods have been proposed, such as SMOTE (Synthetic Minority Oversampling TEchnique) [10], ADASYN (ADAptive SYNthetic sampling approach) [27] or MWMOTE (Majority Weighted Minority Oversampling TEchnique) [2]. These five are remarkable approaches among the State-of-the-Art methods.

SMOTE [10] generates synthetic instances x_g through the linear interpolation of randomly selected instances x and their nearest neighbors x_n from the same minority class (see Eq. (1)). Similarly, ADASYN [27] also generates new instances through this interpolation, however, the selection of the examples are not uniformly random. It prioritizes instances near to the borders of the class according to a distribution of weights. These weights are computed per instance as the ratio of neighbors with a different class label of the class of the evaluated instance.

Linear interpolation is carried out using the following formula:

$$x_g = x + (x_n - x) * \alpha \tag{1}$$

where α is a random number chosen in the range [0,1].

MWMOTE [2] is based on clustering to define the group of instances to be interpolated. First, MWMOTE identifies the minority instances at the borders, i.e. hard to learn instances, and removes the possible noisy instances. Then, sample weights are assigned in relation to their closeness to a border and the low density of their clusters. Finally, these instances are randomly selected according to their weights and interpolated with another sample of the same cluster.

Additionally, standard global metrics negatively influence the performance and they have to be replaced by more suitable options: Area Under the ROC Curve [29] or Geometric Mean are commonly used in binary imbalance problems [38,44]. Macro Average Arithmetic (MAvA) [47] has been frequently chosen for multi-class scenarios and is computed as followed:

$$MAvA = \frac{1}{m} \sum_{i=1}^{m} ACC_i \tag{2}$$

where ACC_i is the partial accuracy rate for the class i independently.

Algorithm 1 Example in pseudo-code of the identification of the type of an example x.

```
function SAMPLETYPE(x - evaluated instance, y - instance x class label, D - data-set)
    if ∃(x_i, y_i) ∈D | x_i ⪰ x ∧ y > y_i ∨ x ⪰ x_i ∧ y_i > y then
        Set x as violation of monotonicity
    else                                          ▷ Get set of instances with same class as y
        S_y ← {∀(x_i, y_i) ∈ D | y_i = y}
        if ∄x_i ∈ S_y | x_i ⪰ x ∨ x ⪰ x_i then
            Set x as a instance at the limit of chain
        else
            Set x as a instance inside a chain
        end if
    end if
end function
```

3. Chain based sampling in monotonic imbalanced classification

The intention of this work is to design proper ways of sampling monotonic data-sets without loosing their monotonicity. To do so, we propose a new sampling schema based on chains. The main difference is the maintenance of the essential parts of the chains of each class. That is, the upper- and lower- instances of each class in each chain [35].

A monotonic chain is a set S of comparable instances, i.e. those which can be sorted. The upper-instance $\uparrow x_S$ of the chain S is the one that dominates all the other samples of the chain. The lower-instance $\downarrow x_S$ of the chain is dominated by the rest. The upper- and lower- instances of each class for a whole chain determine the upper and lower bounds of the range of possible labels without monotonic violations for new instances which are in that chain [16,35]. Formally, the upper- and lower- instances are defined as the following:

$$\uparrow x_S = x \in S : \forall x_i \in S \; x \succeq x_i \tag{3}$$

$$\downarrow x_S = x \in S : \forall x_i \in S \; x_i \succeq x \tag{4}$$

Since sampling techniques usually work independently for each class, chains can be segmented in subsets of a certain class. Then, the instances at the limits of these subsets are the ones that must remain untouched.

In contrast to traditional methods, monotonic sampling also takes monotonic noisy instances into consideration, avoiding them so as not to greatly affect the decisions during preprocessing. Even more, our techniques mitigate their impacts when possible.

To achieve these objectives, the monotonic sampling techniques classify the instances of each class in three different groups of relevance. This classification is used to bias the selection of the examples involved in the sampling, promoting instances at the limits and inside chains and avoiding monotonic noise. The three groups of representatives are detailed in the following:

- At the limits of chains: As said before, these are the most important instances of the chains and the classes, because they determine the ordering limit of each class. Therefore, the sampling techniques prioritize them over the rest. This type of example is identified when there is no comparable example of the same class that is greater or smaller, i.e. upper- or lower- limit, respectively. As explained above, chains are treated independently for each class.
- Inside chains: These representatives give consistency to the chains. The more of this type there is in a chain, the stronger and more relevant the chain is. When possible, these examples must be conserved during sampling. These are the counterparts of the previous type, so they are identified by finding greater and smaller comparable samples.
- Violation of monotonicity: These are monotonic discrepancies, that is, instances that are not monotonically consistent with others. They must be avoided, because they are one of the main reasons for monotonic deterioration of the models. They are identified by comparison with comparable instances of other classes. Sampling techniques avoid them when possible. Tolerance to these examples is computed as indirectly proportional to the number of violations of the instances. When a instance has a high number of violations, there will be only a small possibility of being involved in the sampling.

Algorithm 1 exemplifies with pseudo-code and formal expressions how the types of an instance x are determined.

With these ideas, we settle the bases to extend the use of chains in nearly every sampling technique, both under-sampling and over-sampling approaches.

3.1. Under-sampling for monotonic imbalanced classification

For monotonic versions of under-sampling methods, it is highly important to preserve the upper- and lower-limits of the chains. As many as possible of these instances will be prioritized and maintained in the final sampled set until equal balance is reached.

Algorithm 2 Monotonic Random Under-Sampling.

```
1: function MRUS(N - number of instances to select, S_y - sampled set of class y, D - data-set)
2:     initialize: S_violations ← ∅, S_atLimits ← ∅, S_inChain ← ∅, Prob ← ∅, S_res ← ∅
3:     for (x_i, y_i) ∈ S_y do
4:         if sampleType(x_i, y_i, D) = monotonicViolation then
5:             Prob_i = 1.0/numViolations(x_i, y_i, D)
6:             S_violation ← (x_i, y_i)
7:         else
8:             if sampleType(x_i, y_i, D) = atLimits then
9:                 S_atLimits ← (x_i, y_i)
10:            else
11:                S_inChain ← (x_i, y_i)
12:            end if
13:        end if
14:    end for
15:    if Size(S_atLimits) > N then                                   ▷ Step 1
16:        S_res ← uniformlyRandomSelection(S_atLimits)
17:    else                                                           ▷ Step 2
18:        S_res ← S_atLimits
19:        if Size(S_inChain) > (N − Size(S_atLimits)) then
20:            S_res ← uniformlyRandomSelection(S_inChain)
21:        else                                                       ▷ Step 3
22:            S_res ← S_inChain
23:            Normalize(Prob)
24:            #Samples = N − Size(S_atLimits) − Size(S_atChains)
25:            for j ∈ [1, #Samples] do
26:                S_res ← rouletteSelection(S_violations, Prob)
27:            end for
28:        end if
29:    end if
30:    return S_res
31: end function
```

Additionally, under-sampling methods could help to mitigate the impact of monotonic violations of the data-set by purposely sampling the monotonic noisy instances. This feature is unique in under-sampling methods, since over-sampling lacks any similar characteristics.

Depending on the original under-sampling technique, this knowledge can be introduced in different ways. For weight-based sampling, a monotonic weight can be given to each instance according the three previously mentioned types of instances. The weights given to instances at the limits of chains are recommended to be higher than the ones given to the samples inside chains. Non monotonic instances receive a weight of 0 ensuring their exclusion, but only if there are enough samples of the other two types to achieve class balance. Otherwise, these weights may be indirectly proportional to the number of monotonic violations caused by the instances. These monotonic weights can be included as a factor of the original weighting. See the following subsection for more details.

However, we have extended the famous Random Under-Sampling. The original method randomly selects a subset of the majority classes to rebalance the data-set. RUS does not use any weights or probabilities in its process of selection. It merely chooses the samples in a uniformly random way. Therefore, the monotonic RUS approach (mRUS) is designed as a hierarchical selection according to the mentioned groups of relevance instead of being purely random. This hierarchical selection has the following steps:

- *Step 1:* If the number of at-the-limits samples is big enough to fulfill the rebalance of the data-sets, a subset of this group is uniformly and randomly selected. Otherwise, every instance is selected and Step 2 will be executed.
- *Step 2:* If the number of instances inside the chains is enough to rebalance the data-sets within the already selected instances, the same selection procedure is applied to this type of instances. Otherwise, all are selected and Step 3 will continue with the rebalance.
- *Step 3:* Only if needed, monotonic noisy instances will be used to complete the balanced data-set. The selection is not uniform; it is guided by a probability distribution. The probability to select a certain non monotonic sample is computed as the inverse of the number of monotonic violations. So, instances with more discrepancies have less chance of being selected, while instances with just one inconsistency have the highest probability. This final selection has been implemented following a roulette wheel scheme, as seen in Line 26. This procedure computes a cumulative probability

distribution and generates a random number between 0 and 1. The first cumulative probability greater than this number indicates the selected instance.

These steps are exemplified in the following algorithm:

3.2. Over-sampling for monotonic imbalanced classification

Over-Sampling techniques are more likely to be weight-based methods. Therefore, their monotonic extensions could easily include a monotonic weighting according to the three categories of the previously defined instances. These weights can then be transformed in a probability distribution to guide the selection of the over-sampled instances. The hierarchical approach used for under-sampling lacks common sense for over-sampling techniques, since they cannot easily implement it.

As previously mentioned, monotonic noise or non monotonic instances should be excluded from the sampling decisions. However, they cannot be removed from the final data-set, as in under-sampling. In order to reduce their impact as much as possible, the probability of them being selected is equal to zero. Then, they will not be replicated. But if the sampled class is entirely composed of instances with monotonic violations, they cannot be avoided. Then, they will be selected according to the inverse of their number of violations within the range (0,1].

Since instances at the limit of chains seize more valuable information, they should have a significantly higher chance than those inside chains. At least, the probability of the former instances should double that of the latter. Finally, these weights are normalized to form a probability distribution. Algorithm 3 shows the implementation of this weight scheme in pseudo-code.

According to this weighting approach, we have chosen four well-known over-sampling techniques to design a monotonic version of them. We have detailed the highlights of each of these methods in the following:

- **ROS:** The original Random Over-Sampling technique randomly replicates instances of the minority classes, which could greatly deteriorate the monotonicity of the data-sets if non monotonic instances are frequently selected. The monotonic version of ROS (mROS) does not perform a uniformly random selection anymore. It selects the instances following a roulette wheel selection scheme as in mRUS, but with the previously explained probability distribution computed by Algorithm 3. An explanation in pseudo-code of mROS can be found in Algorithm 4.
  mROS is not so effective for the methods that transform replicas into class membership, such as OSDL [35] or some relabeling techniques [17,43]. These methods intend to mitigate the impact of replicas with different classes by fusing them into probabilities. For data-sets with these inconsistencies, mROS is very effective. On the other hand, mROS achieves nothing. Therefore, we have designed a different method of replication (Line 7). The replications will have a small variation in their values, to avoid attaining exactly the same instances. This is done using jittering, that is, by introducing a small Gaussian noise to the feature values of the selected instance.
  In order to ensure that the derived sample is as close as possible to the selected one, the standard deviation σ of the Gaussian noise generator must be sufficiently small, so that $\sigma = 0.001$. Therefore, this small variation of each feature is

Algorithm 3 Monotonic weights function.

```
1:  function MONOTONICWEIGHTS(S_y - data-set of class, D - data-set)
2:      initialize: Prob[1..size(S_y)] = 0, S_violations ← ∅
3:      for (x_i, y_i) ∈ S_y do
4:          if sampleType(x_i, y_i, D) = monotonicViolation then
5:              Prob_i = 1.0/numViolations(x_i, y_i, D)
6:              S_violations ← i
7:          else
8:              if sampleType(x_i, y_i, D) = atLimits then
9:                  Prob_i = 4.0
10:             else
11:                 Prob_i = 2.0
12:             end if
13:         end if
14:     end for
15:     if Size(S_violations) != Size(S_y) then
16:         for i ∈ S_violations do
17:             Prob_i = 0
18:         end for
19:     end if
20:     Normalize(Prob)
21:     return Prob
22: end function
```

Algorithm 4 Monotonic Random Over-Sampling.

1: **function** MROS(*rep* - option of replication (std or 1-NN) ,*N* - number of instances to replicate, S_y - sampled set of class *y*, *D* - data-set)
2: **initialize:** $Prob \leftarrow monotonicWeights(S_y, D)$, $S_{res} \leftarrow S_y$
3: **for** $i \in [1, N]$ **do**
4: $(x_i, y_i) = rouletteSelection(S_y, Prob)$
5: **if** $rep = true$ **then**
6: $S_{res} \leftarrow (x_i, y_i)$
7: **else**
8: $x_g = gaussianRandom(\mu = 0, \sigma = 0.001) + x_i$
9: $S_{res} \leftarrow (x_g, y_i)$
10: **end if**
11: **end for**
12: **return** S_{res}
13: **end function**

Algorithm 5 Monotonic SMOTE.

1: **function** MSMOTE(*k* - nearest neighbors, $interpolationRatio = 0.5$, *N* - number of instances to replicate, S_y - sampled set of class *y*, *D* - data-set)
2: **initialize:** $Prob \leftarrow monotonicWeights(S_y, D)$, $S_{res} \leftarrow S_y$
3: **for** $i \in [1, N]$ **do**
4: $(x_i, y_i) = rouletteSelection(S_y, Prob)$
5: $S_{nn} \leftarrow kNN(k, S_y).getNeighbors(x_i, y_i)$
6: $Prob_{nn} \leftarrow monotonicWeights(S_{nn}, D)$
7: $(x_{nn}, y_{nn}) = rouletteSelection(S_{nn}, Prob_n n)$
8: $\alpha = random(0, 1)$
9: $x_g = x_i + interpolationRatio * (x_{nn} - x_i) * \alpha$
10: $S_{res} \leftarrow (x_g, y_i)$
11: **end for**
12: **return** S_{res}
13: **end function**

randomly generated following a Gaussian distribution with a mean μ equal to original feature value and a σ of 0.001 (Line 8). These two versions of mROS are implemented as a parameter of the same method, so the best result is always chosen according to the features of the classification method.

- **SMOTE** [10]: This method generates synthetic instances through interpolation of randomly selected instances and their nearest neighbors. Its adaption must avoid the selection of monotonic noisy instances, because their interpolations will probably be noisy. Algorithm 5 represents the method Monotonic SMOTE (mSMOTE).
  mSMOTE follows exactly the same selection procedure as mROS. When a sample is selected (Line 4), its *k* nearest neighbors are computed and just one is chosen to perform the interpolation. As shown in Line 6, the nearest neighbors are selected with the same probabilities scheme mentioned before. That is, neighbors at the limits of chains are prioritized over those that are inside a chain.
  For mSMOTE, the interpolations are limited and ensured to be computed with a maximum distance of a percentage of the total distance to the selected nearest neighbor. This percentage can be considered to be a parameter of the method, *interpolationRatio*. This parameter can be set from 0, meaning no interpolation is performed and thus it works as a mROS with repetitions, to the value of 1, as the standard linear interpolation. Values in this range modulate "room for error" during the interpolation process. If we want to be conservative and try to preserve monotonicity as much as possible, we will try to generate synthetic instances near the one selected, i.e. setting a low value for the parameter. We recommend setting it to 0.5, as a good trade off between being conservative and covering more of the problem space. The expression in Line 9 reflects the modification of the standard expression for linear interpolation.
- **ADASYN** [27]: The original ADASYN already includes a weighting schema to prioritize samples at the borders of the minority class. The weight of a instance is obtained as the ratio of nearest neighbors with a different class value. The instances are selected according to these weights, then, the linear interpolation with their neighbors are performed as SMOTE.
  Monotonic ADASYN (mADASYN) is similar to mSMOTE. However, since it already computes weights for each instance, a combination of both schemes has to be defined. This is done by the product of both normalized weights, then they are normalized again. The following expression shows this transformation:

$$Prob_i = \frac{\Delta_i}{k} * monotonicWeights(x_i, D) \tag{5}$$

Table 1
Summary of the sampling techniques and their suggested parameters.

Sampling techniques	Parameters
RUS & ROS	No parameters
SMOTE & ADASYN & ADASYNOrd	$k = 5$, $distance = Euclidean$
MWMOTE & MWMOTEOrd	$k_1 = 5, k_2 = 3, k_3 = 3, C_p = 3,$ $C_f = 5, C_{max} = 2, distance = Euclidean$
CWOSOrd	$NN = 5, NS = 5, \alpha = 1,$ $C_{thres} = 2, distance = Euclidean$
mRUS & mROS	no parameters
mSMOTE & mADASYN & mMWMOTE	Same recommended parameters, $interpolationRatio = 0.5$

where Δ_i is the number of neighbors with a different class than x_i in the set of k nearest neighbors and $monotonicWeight(x_i, D)$ is the monotonic weight of x_i defined in Algorithm 3 and normalized in the range (0,1]

- **MWMOTE** [2]: The original MWMOTE is a very different approach compared to SMOTE and ADASYN. It uses clustering as well as several nearest neighbor rules to define the instances S_{imin} from the minority class that are really hard to learn. To do so, the set S_{minf} is obtained with the minority instances which are not considered to be noise, that is, instances that have at least one minority instance among their k_1 nearest neighbors. Then, S_{bmaj} is computed as the union of all k_2 nearest enemies, instances from other classes, of the instances in S_{minf}. Finally, S_{imin} is the result of all k_3 nearest neighbors belonging to the minority class of S_{bmaj}. Instances in S_{imin} are the only samples that can be selected according to weight distribution S_w.
  The weight S_w for given sample x_i is computed with the sum of the information weights I_w contributed by all the instances in S_{bmaj}:

$$S_w(x_i) = \sum_{x_j \in S_{bmaj}} I_w(x_j, x_i) \tag{6}$$

  The information weight is composed of a closeness factor $C_f(x_j, x_i)$ and a density factor $D_f(x_j, x_i)$. The former determines the closeness to the decision boundaries, while the latter quantifies the density of the cluster to which the instance x_i belongs.

$$I_w(x_j, x_i) = C_f(x_j, x_i) * D_f(x_j, x_i) \tag{7}$$

  Monotonic MWMOTE (mMWMOTE) maintains this algorithmic structure untouched. Similarly as mADASYN, the monotonic weight distribution is merged with the original distribution S_w. This is done by multiplying both factors:

$$Prob_i = S_w(x_i) * monotonicWeights(x_i, D) \tag{8}$$

  As a reminder, the selection of the neighbor or corresponding cluster instance to be interpolated with the previously selected sample x_i is also guided using the monotonic weighting in the monotonic methods SMOTE, ADASYN and MWMOTE.

4. Experimental framework

This section introduces the experimental framework followed in all the experiments conducted in this paper. In order to study the viability of sampling in monotonic environments, we select five well-known sampling techniques from the-State-of-the-Art: RUS, ROS, SMOTE [10], ADASYN [27] and MWMOTE [2]. These were explained in previous sections.

They were chosen following the study [40] that develops a version of these methods for ordinal regression. Then, we can also test them within monotonic scenarios. Ordinal versions of ROS and SMOTE do not change much: they are just applied to all the classes to balance them. For this reason, we have only included one version of RUS, ROS and SMOTE. However, ADASYN and MWMOTE ordinal extensions include a factor to consider the ordinal rank difference between the evaluated instances and their neighbors or closest samples. For ADASYN, this factor is introduced in the weights calculation, while for MWMOTE, it appears in the closeness factor.

Additionally, in the experiments we have included the new over-sampling technique developed in [40]. CWOSOrd is a cluster-based oversampling technique that is focused on the more complex and smaller clusters, where it generates more synthetic samples. The ordering relationship and the distances to other classes samples are used to compute a probability distribution to guide the random selection of samples for synthetic generation. For more specific details, we refer the readers to original paper [40]. Table 1 recalls the sampling techniques used and their parameters, including our proposals.

Table 2
Description of the data-sets used.

Data-set	Ins.	At.	Cl.	At. directions	%Ins. per Class
balance	625	4	3	{−, −, +, +}	46.1/**7.8**/46.1
car	1728	6	4	All direct	70.1/22.2/**4.0**/**3.8**
ERA	1000	4	9	All direct	**9.1**/**14.2**/18.1/17.2/15.8/**11.8**/**8.8**/**3.1**/**1.9**
ESL	488	4	9	All direct	**0.2**/**2.5**/**7.7**/20.5/23.9/27.6/**12.8**/**3.9**/**0.9**
LEV	1000	4	5	All direct	**9.2**/28.0/40.3/19.7/**2.8**
SWD	1000	10	4	All direct	**3.1**/35.2/39.9/21.8
windsorhousing	546	11	2	All direct	76.6/**23.4**
wisconsin	699	9	2	All direct	65.3/**34.7**

Table 3
Parameters considered for the algorithms compared.

Algorithm	Parameters
M*k*NN [16]	k = 5, distance = euclidean, neighborsType = inRange
OSDL [35]	balanced = No, classificationType = median, lowerBound = 0, upperBound = 1 tuneInterpolationParameter = No, weighted = No, interpolationStepSize = 10, interpolationParameter = 0.5
OLM [3]	modeResolution = conservative modeClassification = conservative
C4.5-MID [4]	R = 1, confidence = 0.25, items per leaf = 2
MonMLP [31]	default parameters, hidden1 = 8 iter.max = 1000, monotone = all att

These methods are applied to 8 different monotonic data-sets that have been selected as they are commonly used in the literature of classification with monotonic constraints. Table 2 shows the characteristics of each data-set: number of instances (Ins.), attributes (At.) and classes (Cl.), monotone direction between each attribute and the class, and the percentage of representation of each class label in the data-set. As can be seen in the table, the majority of these data-sets are multi-class and highly imbalanced problems. However, the classification cannot be carried out by class decomposition schemes [18,21], such as One-vs-One [23,51] or One-vs-All [9]. Since these algorithms perform the classification in decomposed binary independent problems, the order and monotonic relations between classes are distorted.

Therefore, the sampling techniques are applied to the classes depending on whether they are majority or minority classes for under- or over-sampling, respectively. The sampling is stopped when equal balance is reached. The minority classes are represented with bold-face font in the column %Ins. per Class at Table 2, the rest are considered majority classes.

Then, the resulting data-sets are classified by 5 different multi-class monotonic classifiers. Table 3 recalls the monotonic classifiers used during these experiments and their parameters. All this process is carried out following a 10-fold stratified cross validation scheme (10-fcv) with the partitions found in the software KEEL [49].

As performance measures, we have selected MAvA and Non-Monotonic Index (NMI). The above mentioned MAvA evaluates the prediction capability of the classifiers for multi-class imbalance problems. MAvA results should improve after applying sampling. Non-Monotonic Index is used to determine the impact of the sampling techniques on the monotonicity of the monotonic classifier predictions. Non-Monotonic Index (NMI) [4] measures the ratio of instance pairs *NMP* that violate monotonicity, with respect to the total number of pairs of instances in a predicted set. To compute it, the sets of test predictions resulting from the 10-fcv classification are merged into only one set. Then, the NMI is computed over this set following the expression:

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

where N stands for the total number of instances.

In order to corroborate the different outcomes of the experiments, we use non-parametric statistical tests: the well-known Friedman rank test [20,25] and Bayesian Sign test [5].

Bayesian Sign test is a pairwise Bayesian non-parametric sign test based on the Dirichlet Process [5]. This test computes a distribution with the difference of the results obtained by the two compared algorithms (A vs B). Then, a decision is made according to the position of the majority of the distributions in one of the three regions: left (superiority of method B), rope (statistical equivalence) and right (superiority of method A)[5].

We have used the R package, named rNPBST [8], to perform the different tests and to present the graphics in the following section.

Table 4
MAvA average results obtained by the selected classifiers with standard and ordinal sampling.

	Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
MkNN	0.5727	0.5827	0.5742	0.6106	*0.6143*	0.5828	*0.6128*	0.6105	0.6137
OSDL	0.5909	0.4587	0.4502	0.4351	0.4344	*0.4373*	0.4459	*0.4472*	0.4429
OLM	0.5468	0.5273	0.5656	0.5762	*0.5771*	*0.5771*	0.5810	*0.5815*	0.5746
MID	0.5434	0.5693	0.6272	0.6064	*0.6104*	0.5955	0.5929	*0.5987*	0.5912
MonMLP	0.5582	0.6034	0.6300	0.5902	0.5865	*0.5924*	0.5689	*0.5771*	0.5709

Table 5
Friedman ranking for MAvA results obtained by the selected classifiers with standard and ordinal sampling.

	MAvA				
Rank	M*k*NN	OLM	MID	OSDL	MonMLP
1	ADASYN (3.50)	MWMOTEOrd (3.31)	ROS (2.50)	Original (1.50)	SMOTE (3.75)
2	MWMOTE (3.75)	MWMOTE (3.93)	ADASYN (3.56)	RUS (3.62)	ADASYN (3.93)
3	CWOSOrd (3.75)	ADASYNOrd (4.06)	SMOTE (3.62)	SMOTE (4.93)	ADASYNOrd (4.18)
4	SMOTE (4.62)	SMOTE (4.37)	MWMOTEOrd (4.06)	ADASYNOrd (5.00)	ROS (4.37)
5	MWMOTEOrd (4.75)	ADASYN (4.50)	MWMOTE (5.56)	ROS (5.12)	MWMOTEOrd (4.68)
6	RUS (5.25)	CWOSOrd (4.56)	ADASYNOrd (5.81)	ADASYN (5.31)	RUS (5.25)
7	ADASYNOrd (5.37)	ROS (5.62)	Original (6.00)	MWMOTEOrd (5.5)	MWMOTE (5.31)
8	ROS (7.00)	Original (7.12)	CWOSOrd (6.25)	MWMOTE (6.81)	CWOSOrd (6.50)
9	Original (7.00)	RUS (7.50)	RUS (7.62)	CWOSOrd (7.18)	Original (7.00)

5. Experimental studies

This section is devoted to experimentally showing the influence on monotonicity of standard, ordinal and monotonic chain based sampling techniques. First, the performance results of several standard and ordinal samplings are analyzed, paying special attention to their impact on the monotonicity of the prediction of different monotonic classifiers. The following subsection presents the results obtained by our new monotonic sampling scheme. An extensive comparison is made with the original results of the classifiers and the standard sampling.

5.1. On the viability of standard sampling techniques in monotonic imbalanced scenarios

In order to test the feasibility of sampling for monotonic imbalanced data-sets, we must first assure that the prediction capability has increased after applying them. Table 4 recalls the average MAvA results achieved by each classifier with and without the resulting set of each preprocessing method. The table cells colored in gray indicate an improvement compared to the original results obtained by the classifier. Italics font is used to highlight the best performance for each sampling group; in this case, the comparison of the standard and ordinal version of ADASYN and MWMOTE.

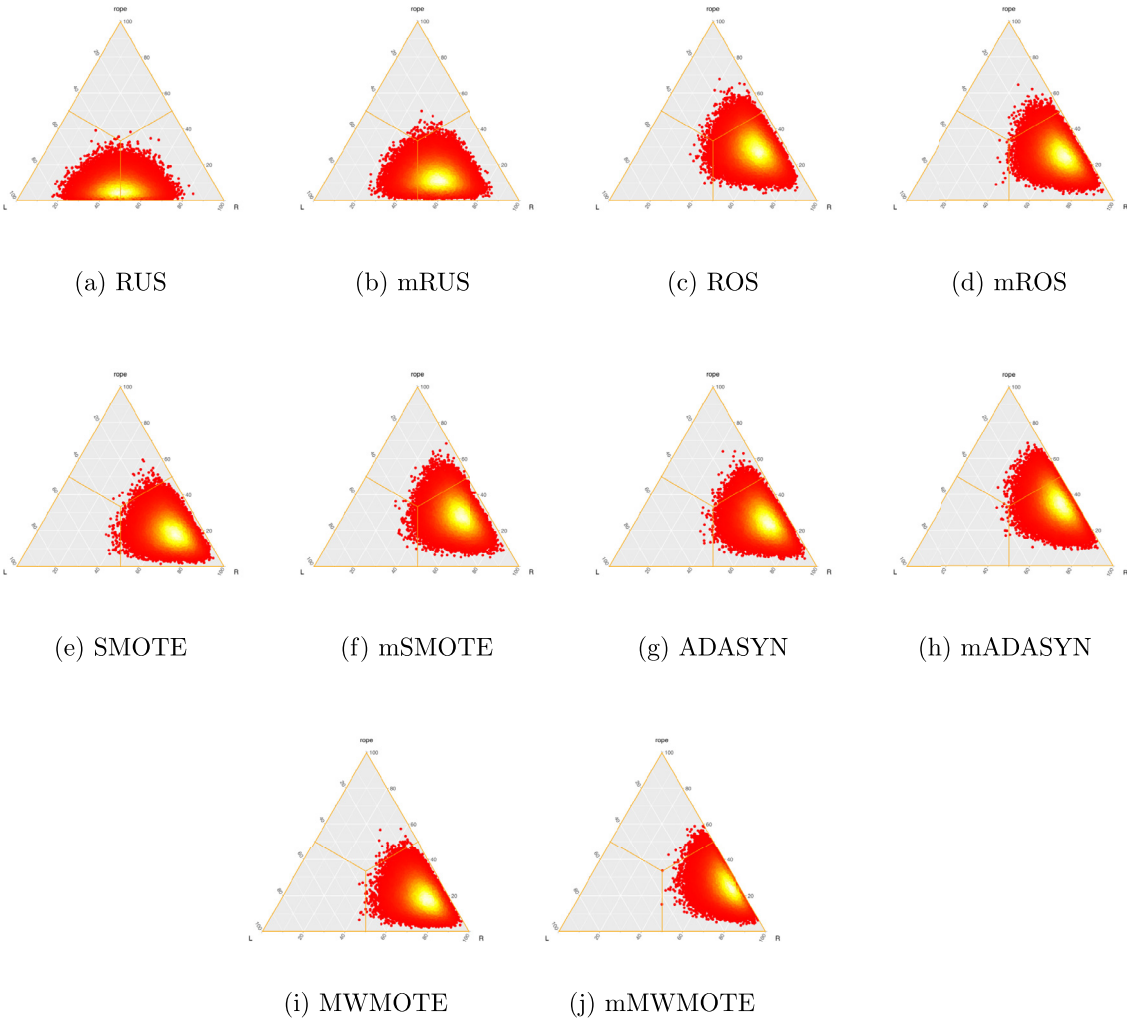
As shown in Table 4, nearly all results achieved with balanced sets outperform the classification with skewed data-sets. Even though it was expected, these improvements are remarkably significant, especially the one obtained with the combination of ADASYN+M*k*NN, ROS+MID and ROS+MonMLP. However, OSDL always achieves better results with the original data-sets. With this outcome, we can deduce that sampling techniques will not work properly for OSDL. Probably, this is related to its particular way of classification based on stochastic dominance and probability distribution functions extracted from repeated instances with different class labels [35]. Therefore, we exclude OSDL from the rest of the experiments.

In addition, Table 5 shows a Friedman ranking per selected classifier of the sampling techniques sorted by their performance in terms of MAvA. Here, the MAvA advantage of sampling can be also appreciated, since the original results are usually ranked last, with the exception of OSDL.

There is not one sampling method which is superior to the rest. For some classifiers, certain sampling methods are well ranked, while for others, they are not. Their good or bad performance strongly depends on the base learner. For ordinal sampling, there is no significant improvement over their standard versions. ADASYN is ranked higher more times than ADASYNOrd, while, for MWMOTE, the ordinal version is usually ranked better. CWOSOrd is not well ranked for any of the selected classifiers.

A better prediction ability thanks to a preprocessing technique is not enough to prove its viability for classification with monotonic constraints. Its impact on monotonicity must be studied. Table 6 gathers the average NMI retrieved from the selected classifiers with the different sampling methods. As in the previous Table 4, the results in gray improve those recalled purely by the classifier. With exception of SMOTE and ADASYN for OLM, all NMI measured with sampling are much higher. Therefore, these sampling techniques really deteriorate the monotonicity of the data-sets and, as consequence, the monotonicity of the monotonic learner.

This fact can be also observed in the Friedman ranking presented in the Table 7 that composed of all the NMI results. The original classifiers are nearly always ranked first with a small ranking value compared to the second best.



(a) RUS (b) mRUS (c) ROS (d) mROS

(e) SMOTE (f) mSMOTE (g) ADASYN (h) mADASYN

(i) MWMOTE (j) mMWMOTE

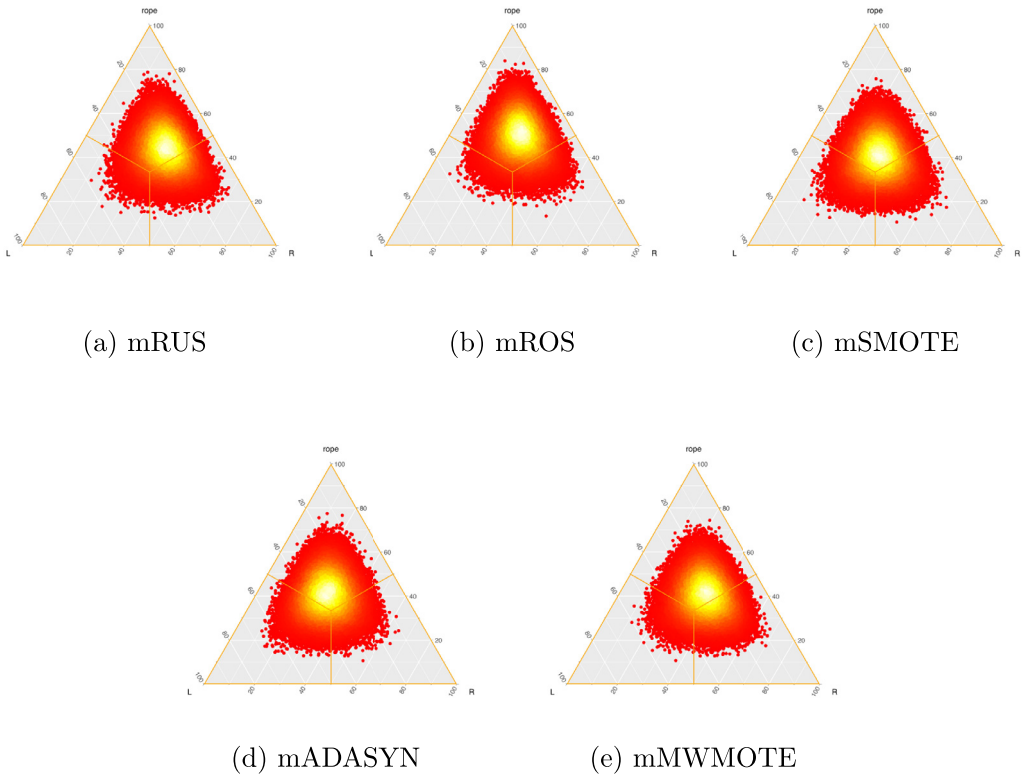
Fig. 1. Bayesian Sign Test heatmap for the sampling MAvA results vs the original MAvA results.

Table 6
NMI average results obtained by the selected classifiers with standard and ordinal sampling.

	Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
MkNN	0.0011	0.0034	0.0014	0.0020	*0.0024*	0.0053	*0.0020*	0.0022	0.0026
OSDL	0.0009	0.0020	0.0164	0.0052	0.0051	*0.0048*	*0.0044*	0.0057	0.0069
OLM	0.0018	0.0028	0.0021	0.0017	0.0016	*0.0015*	*0.0019*	*0.0019*	0.0020
MID	0.0030	0.0090	0.0050	0.0038	*0.0045*	0.0051	*0.0042*	0.0044	0.0037
MonMLP	0.0005	0.0021	0.0013	0.0013	*0.0015*	0.0016	*0.0036*	0.0037	0.0076

Table 7
Friedman ranking for NMI results obtained by the selected classifiers with standard and ordinal sampling.

	NMI				
Rank	MkNN	OLM	MID	OSDL	MonMLP
1	Original (2.43)	ADASYNOrd (3.31)	Original (3.37)	Original (3.18)	Original (2.50)
2	ROS (3.81)	ADASYN (3.68)	MWMOTEOrd (4.00)	RUS (4.43)	ADASYN (4.37)
3	ADASYN (4.68)	ROS (4.68)	SMOTE (4.12)	SMOTE (4.81)	ADASYNOrd (4.62)
4	SMOTE (4.87)	SMOTE (4.75)	ADASYN (4.56)	MWMOTE (4.81)	SMOTE (4.87)
5	MWMOTE (5.12)	Original (5.12)	CWOSOrd (4.87)	ADASYN (5.31)	MWMOTE (4.87)
6	CWOSOrd (5.25)	RUS (5.5)	MWMOTE (5.12)	ADASYNOrd (5.37)	ROS (5.00)
7	MWMOTEOrd (5.50)	MWMOTEOrd (5.62)	ADASYNOrd (5.68)	ROS (5.62)	MWMOTEOrd (5.37)
8	ADASYNOrd (5.93)	CWOSOrd (6.06)	ROS (5.87)	MWMOTEOrd (5.62)	CWOSOrd (6.00)
9	RUS (7.38)	MWMOTE (6.25)	RUS (7.37)	CWOSOrd (5.81)	RUS (7.37)



(a) mRUS (b) mROS (c) mSMOTE

(d) mADASYN (e) mMWMOTE

Fig. 2. Bayesian Sign Test heatmap for the monotonic sampling MAvA results vs the standard sampling MAvA results.

Additionally, thanks this study of monotonicity, ordinal sampling methods become impractical, since ADASYN is still usually better than ADASYNOrd, MWMOTE is now ranked higher than its ordinal version in most cases and once again, CWOSOrd ranks last in the Friedman ranking.

With this first experimental study, we can conclude that standard sampling can improve the prediction of the classifiers, but they impair the monotonicity to a large extent. And the existing ordinal sampling methods do not solve the problem.

The complete results achieved by these methods in terms of MAvA and NMI can be found in the following experimental study, for the standard sampling methods, and in Appendix A. They have been removed from this section for the sake of clarity.

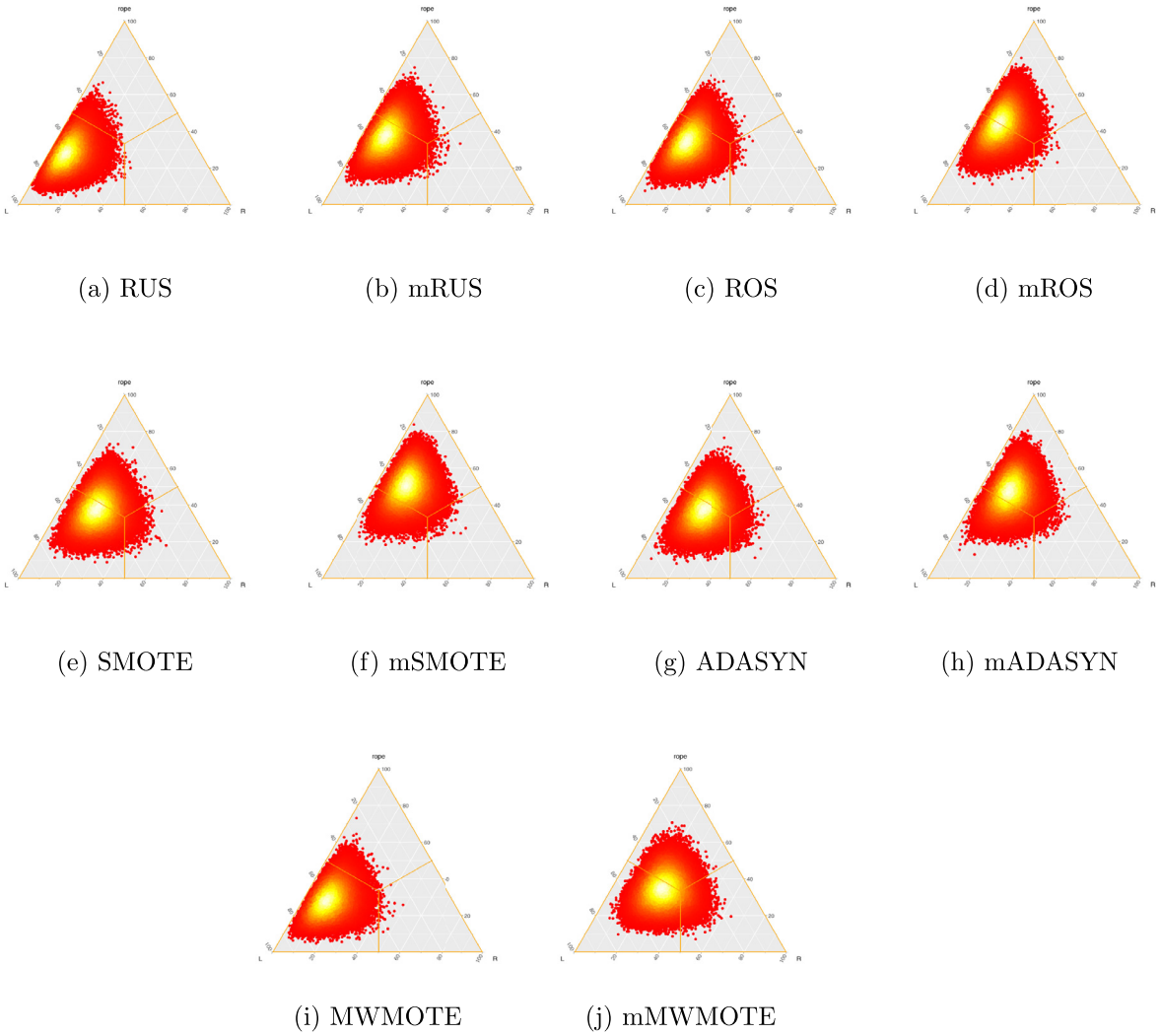
5.2. Standard and monotonic sampling: results and analysis

Throughout this empirical study, we analyze the behavior of our chain based sampling techniques. Their performance in terms of MAvA and NMI are compared with the classifiers with and without the standard sampling methods. Given the bad performance shown in the previous experimental study using OSDL with sampled data-sets in comparison to the one with original sets, OSDL has been excluded from this experiment.

Table 8 recalls the MAvA achieved per data-set with the different configurations of base learners and standard or monotonic preprocessing. The best results between the standard and monotonic versions of each sampling are highlighted in italics. The gray cells represent an improvement when compared to the classifiers without sampling.

All the classifiers improve their performance in terms of MAvA by using sampling, both standard and monotonic, for most of the data-sets. The best sampling methods on average are the standard ROS sampling and Monotonic ADASYN (mADASYN). RUS and mRUS are the methods which show the least improvement, probably because the under-sampling performed to balance the class distribution is too aggressive. The monotonic version is slightly better. At first sight, it is difficult to guess which is the best set of methods in terms of predictability: standard or monotonic sampling.

Bayesian Sign test has been used to thoroughly analyze the results and to clearly present the significance of the differences between the evaluated methods with the help of heatmap plots. Fig. 1 represents the probability distributions of the differences between the sampling methods and the original results of the classifiers. As we can observe, the majority of the points in nearly all the plots are on the right side of the triangle. This means that the results obtained with sampling are significantly better than those obtained without sampling. With the exception of RUS (Fig. 1a) and mRUS (Fig. 1a), the test assigns an almost 0 probability in favor of the original classifiers. Monotonic RUS (Fig. 1a) is still valued, since most of the distribution is in the right region.



(a) RUS (b) mRUS (c) ROS (d) mROS

(e) SMOTE (f) mSMOTE (g) ADASYN (h) mADASYN

(i) MWMOTE (j) mMWMOTE

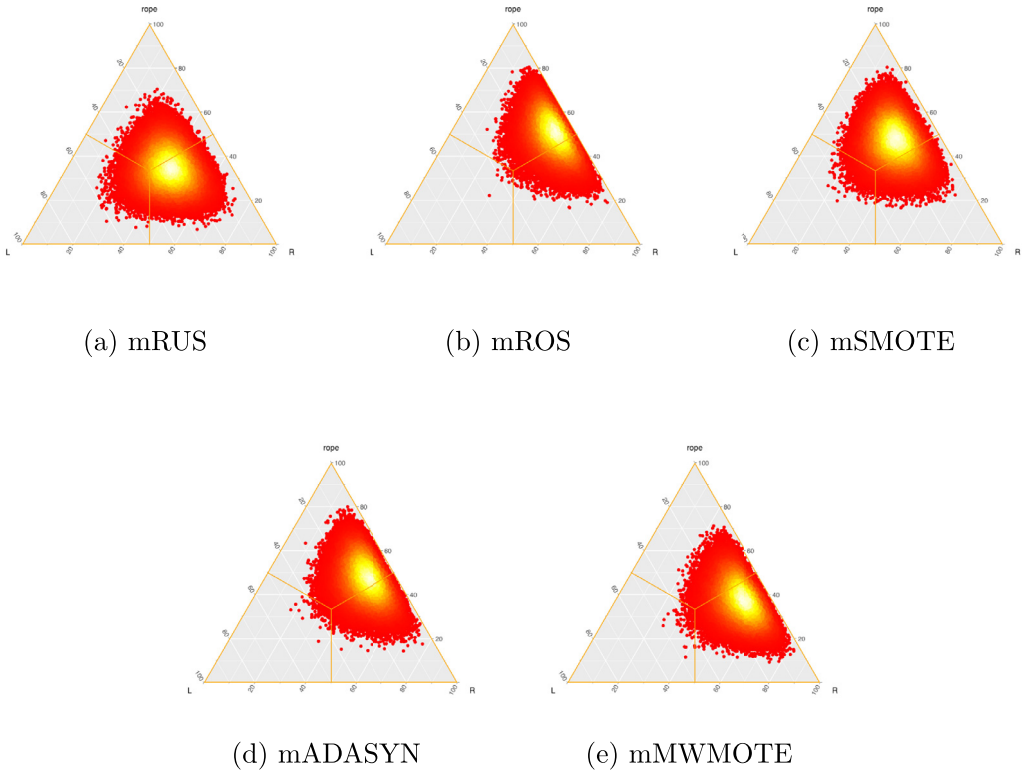
Fig. 3. Bayesian Sign Test heatmap for the sampling NMI results vs the original NMI results.

Fig. 2 graphically compares the MAvA results of the monotonic versus standard sampling techniques. With a large part of distributions in the rope regions, there is no significant difference in terms of predicative capability between these two sampling groups. Monotonic RUS, ROS and MWMOTE (Fig. 2a, b and e, respectively) are slightly better than their standard versions, since their distributions have shifted a bit to the right.

Table 9 gathers the NMI results reached using the different standard and monotonic sampling methods for every classifier per data-set. Cells in gray indicate the preservation of monotonicity. Even though there are not a huge amount of cells in gray, monotonic sampling results are quite similar to the original results when they do not achieve an improvement. It is worth mentioning that monotonicity is preserved on average for OLM and MID for nearly all monotonic sampling, except mROS and mRUS, respectively. On average, every monotonic sampling outperforms its counterpart for each classifier and the mean of all the results. The majority of best results per data-set and type of sampling, marked in italics, are located in the monotonic columns. These results represent a better preservation of monotonicity using the monotonic sampling compared to the traditional methods.

Fig. 3 represents the Bayesian Sign test results for the difference of each of the sampling techniques and the NMI results obtained without them. At first glance, there is a change in the tendency of the distributions of the standard and the monotonic methods. Even though all of them have shifted to the left, towards the original results of the classifiers, the distributions of the monotonic sampling techniques are largely located in the rope region, i.e. the practical equivalence region. Monotonic RUS is excluded from this good behavior. Additionally, for monotonic MWMOTE (Fig. 3j), a good amount of points are located in the right section, indicating some monotonic improvements as compared to the original results.

Fig. 4 shows the result of the Bayesian Sign test for the NMI comparison of monotonic sampling vs standard sampling methods. As we can observe, all the distributions are heavily shifted to the right, emphasizing the superiority of monotonic



(a) mRUS (b) mROS (c) mSMOTE

(d) mADASYN (e) mMWMOTE

Fig. 4. Bayesian Sign Test heatmap for the monotonic sampling NMI results vs the standard sampling NMI results.

Table 8
MAvA results obtained by the selected classifiers with standard and monotonic sampling.

		Original	RUS	mRUS	ROS	mROS	SMOTE	mSMOTE	ADASYN	mADASYN	MWMOTE	mMWMOTE
MkNN	*balance*	0.6343	*0.7241*	0.6575	*0.6100*	*0.6100*	*0.6436*	0.5982	*0.6376*	0.5974	0.6637	*0.6969*
	car	0.9134	0.8824	*0.9759*	0.9553	*0.9673*	*0.9696*	0.9673	*0.9709*	0.9677	*0.9660*	0.9620
	ERA	0.1357	*0.1480*	0.1450	*0.1338*	*0.1338*	0.2501	*0.2878*	0.2794	*0.2949*	*0.2845*	0.2717
	ESL	0.4674	*0.3986*	0.3573	0.4621	*0.4834*	0.4660	*0.4764*	0.4711	*0.4754*	*0.4945*	0.4843
	LEV	0.3924	0.4182	*0.4372*	0.3834	*0.4783*	0.4682	*0.5097*	0.4737	*0.5089*	0.4467	*0.4680*
	SWD	0.4004	0.4692	*0.4908*	*0.4111*	*0.4111*	0.4608	*0.4703*	*0.4616*	0.4605	*0.4034*	0.3989
	windsorhousing	0.6755	*0.6467*	0.6318	*0.6755*	*0.6755*	0.6562	*0.6755*	0.6477	*0.6755*	0.6726	*0.6755*
	wisconsin	0.9625	*0.9745*	0.9716	0.9625	*0.9662*	*0.9699*	*0.9669*	*0.9721*	0.9709	*0.9709*	0.9706
	Avg:	0.5727	0.5827	*0.5834*	0.5742	*0.5907*	0.6106	*0.6190*	0.6143	*0.6189*	0.6128	*0.6160*
OLM	*balance*	0.6019	0.5243	*0.5961*	*0.6019*	*0.6019*	*0.6268*	0.6019	*0.6200*	0.6019	*0.6632*	0.6480
	car	0.8704	0.8424	*0.9231*	*0.9028*	0.8849	*0.8885*	0.8740	*0.8885*	0.8740	*0.8885*	0.8801
	ERA	0.2423	0.2276	*0.2289*	*0.2674*	0.2665	*0.2962*	0.2900	*0.3012*	0.2954	0.2716	*0.2825*
	ESL	0.3981	0.3242	*0.3588*	0.4070	*0.4250*	0.3974	*0.4358*	0.4019	*0.4342*	*0.4225*	0.4176
	LEV	0.4044	0.3801	*0.4188*	0.3729	*0.4225*	0.4165	*0.4530*	0.4075	*0.4560*	0.4155	*0.4315*
	SWD	0.4834	0.4024	*0.5004*	*0.4707*	0.4692	*0.4834*	*0.4834*	*0.4834*	*0.4834*	0.4830	*0.4834*
	windsorhousing	0.5320	*0.6358*	*0.6358*	*0.6352*	0.5599	*0.6444*	0.5442	*0.6324*	0.5410	*0.6487*	0.5553
	wisconsin	0.8418	*0.8814*	*0.8814*	*0.8668*	0.8446	*0.8567*	0.8432	*0.8821*	0.8425	*0.8551*	0.8467
	Avg:	0.5468	0.5273	*0.5679*	*0.5656*	0.5593	*0.5762*	0.5657	*0.5771*	0.5660	*0.5810*	0.5681
MID	*balance*	0.5705	*0.5531*	*0.5531*	0.5941	*0.6499*	*0.5849*	0.5848	0.5674	*0.5826*	0.5755	*0.5849*
	car	0.4603	*0.8191*	0.7840	*0.8810*	0.6499	0.8374	*0.8808*	*0.8786*	0.8640	0.8222	*0.8588*
	ERA	0.3182	*0.2907*	0.2872	0.3040	*0.3131*	*0.3059*	0.3026	*0.2926*	0.2916	0.2867	*0.3102*
	ESL	0.4254	*0.3213*	0.2767	0.4384	*0.4644*	0.4411	*0.4432*	*0.4608*	0.4404	0.4372	*0.4633*
	LEV	0.4731	*0.4592*	0.4517	*0.5732*	0.5641	*0.5367*	0.5048	*0.5101*	0.5073	*0.5157*	0.4958
	SWD	0.4470	0.4676	*0.4916*	0.5542	*0.5592*	*0.5192*	0.4565	*0.4993*	0.4940	0.4542	*0.4888*
	windsorhousing	0.7000	*0.6912*	0.6684	*0.7165*	0.6873	0.6642	*0.6764*	*0.7064*	0.7056	0.6868	*0.7128*
	wisconsin	0.9524	*0.9522*	0.9509	*0.9566*	0.9548	0.9619	*0.9628*	*0.9678*	0.9577	*0.9651*	0.9607
	Avg:	0.5434	*0.5693*	0.5580	*0.6272*	0.6053	*0.6064*	0.6015	*0.6104*	0.6054	0.5929	*0.6094*
MonMLP	*balance*	0.9051	0.8523	*0.8799*	0.9114	*0.9138*	0.8837	*0.9057*	0.8918	*0.9080*	0.8291	*0.8591*
	car	0.6487	*0.7788*	0.7260	0.7326	*0.7769*	*0.7754*	0.7532	0.7564	*0.8088*	0.6941	*0.7552*
	ERA	0.1723	0.1957	*0.2017*	0.2887	*0.2894*	*0.3084*	0.3027	*0.3108*	0.2913	*0.2886*	0.2795
	ESL	0.5370	*0.4503*	0.4435	*0.4941*	0.1111	*0.1111*	*0.1111*	*0.1111*	*0.1111*	*0.1325*	0.1111
	LEV	0.4839	*0.5077*	0.5023	*0.5259*	0.5000	*0.5561*	0.4697	*0.5367*	0.5018	*0.5451*	0.5450
	SWD	0.3497	0.4247	*0.4433*	0.4740	*0.4974*	0.4679	*0.4875*	0.4842	*0.4877*	*0.4415*	0.4014
	windsorhousing	0.5566	*0.6637*	0.6609	0.6544	*0.6581*	*0.6562*	0.6478	0.6359	*0.6478*	*0.6595*	0.6463
	wisconsin	0.8118	0.9537	*0.9564*	*0.9588*	0.9548	*0.9628*	0.9570	0.9648	*0.9688*	0.9611	*0.9665*
	Avg:	0.5582	*0.6034*	0.6017	*0.6300*	0.5877	*0.5902*	0.5794	0.5865	*0.5907*	0.5689	*0.5705*
	Complete Avg:	0.5553	0.5707	*0.5778*	*0.5993*	0.5857	*0.5959*	0.5914	*0.5971*	0.5953	0.5889	*0.5910*

Table 9
NMI results obtained by the selected classifiers with standard and monotonic sampling.

		Original	RUS	mRUS	ROS	mROS	SMOTE	mSMOTE	ADASYN	mADASYN	MWMOTE	mMWMOTE
MkKNN	*balance*	0.0001	0.0006	*0.0003*	*0.0003*	*0.0003*	*0.0004*	0.0005	*0.0004*	0.0005	0.0004	*0.0003*
	car	0.0000	0.0004	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	ERA	0.0056	*0.0081*	0.0084	*0.0071*	*0.0071*	0.0113	*0.0112*	0.0146	*0.0114*	0.0118	*0.0113*
	ESL	0.0012	0.0072	*0.0030*	0.0014	*0.0012*	0.0013	*0.0012*	0.0015	*0.0011*	0.0011	*0.0010*
	LEV	0.0010	0.0066	*0.0044*	0.0016	*0.0012*	0.0019	*0.0012*	0.0018	*0.0012*	0.0019	*0.0016*
	SWD	0.0005	0.0043	*0.0026*	*0.0008*	*0.0008*	*0.0007*	0.0011	*0.0007*	0.0011	0.0011	*0.0009*
	windsorhousing	0.0000	0.0002	*0.0000*	*0.0000*	*0.0000*	0.0001	*0.0000*	0.0001	*0.0000*	*0.0000*	*0.0000*
	wisconsin	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0011	0.0034	*0.0023*	0.0014	*0.0013*	0.0020	*0.0019*	0.0024	*0.0019*	0.0020	*0.0019*
OLM	*balance*	0.0000	0.0001	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0003	*0.0000*
	car	0.0000	0.0001	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	ERA	0.0063	*0.0040*	0.0041	0.0100	*0.0081*	0.0056	*0.0052*	0.0058	*0.0055*	0.0064	*0.0054*
	ESL	0.0025	0.0084	*0.0025*	*0.0023*	0.0032	*0.0023*	0.0025	*0.0023*	0.0025	*0.0018*	0.0027
	LEV	0.0043	0.0038	*0.0025*	*0.0028*	0.0043	*0.0039*	0.0043	*0.0028*	0.0044	*0.0051*	*0.0051*
	SWD	0.0015	0.0062	*0.0046*	*0.0016*	*0.0016*	*0.0015*	*0.0015*	*0.0015*	*0.0015*	0.0016	*0.0015*
	windsorhousing	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0002	*0.0000*	0.0001	*0.0000*	0.0002	*0.0000*
	wisconsin	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0018	0.0028	*0.0017*	*0.0021*	*0.0021*	0.0017	*0.0017*	*0.0016*	0.0017	0.0019	*0.0018*
MID	*balance*	0.0017	*0.0056*	*0.0056*	0.0015	*0.0012*	*0.0012*	0.0017	0.0016	*0.0014*	0.0025	*0.0014*
	car	0.0034	*0.0050*	0.0060	0.0014	*0.0012*	0.0013	*0.0009*	0.0012	*0.0011*	0.0014	*0.0013*
	ERA	0.0086	*0.0064*	0.0071	0.0115	*0.0085*	*0.0087*	0.0090	0.0101	*0.0087*	*0.0106*	0.0109
	ESL	0.0023	0.0205	*0.0146*	0.0064	*0.0031*	0.0041	*0.0021*	0.0038	*0.0033*	0.0044	*0.0025*
	LEV	0.0024	*0.0116*	0.0139	0.0037	*0.0021*	*0.0027*	0.0028	0.0031	*0.0023*	0.0025	*0.0022*
	SWD	0.0020	0.0089	0.0107	0.0026	*0.0025*	0.0027	*0.0025*	0.0029	*0.0027*	*0.0024*	0.0025
	windsorhousing	0.0035	0.0135	*0.0014*	0.0127	*0.0040*	0.0094	*0.0024*	0.0135	*0.0043*	0.0094	*0.0034*
	wisconsin	0.0001	*0.0001*	*0.0001*	*0.0000*	*0.0000*	*0.0001*	*0.0001*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0030	0.0090	*0.0074*	0.0050	*0.0028*	0.0038	*0.0027*	0.0045	*0.0030*	0.0042	*0.0030*
MonMLP	*balance*	0.0000	*0.0001*	*0.0001*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0003	*0.0000*
	car	0.0001	*0.0006*	0.0010	0.0006	*0.0003*	*0.0003*	*0.0003*	*0.0003*	0.0009	*0.0002*	0.0005
	ERA	0.0026	*0.0050*	0.0054	0.0048	*0.0043*	0.0063	*0.0056*	0.0069	*0.0055*	*0.0056*	*0.0056*
	ESL	0.0003	0.0030	*0.0025*	0.0016	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0196	*0.0000*
	LEV	0.0008	*0.0041*	0.0049	0.0027	*0.0018*	0.0023	*0.0018*	0.0023	*0.0015*	0.0020	*0.0016*
	SWD	0.0004	0.0044	*0.0024*	*0.0011*	0.0012	*0.0013*	0.0019	0.0021	*0.0015*	*0.0011*	0.0028
	windsorhousing	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	wisconsin	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0005	0.0021	*0.0020*	0.0013	*0.0010*	0.0013	*0.0012*	0.0015	*0.0012*	0.0036	*0.0013*
	Complete Avg:	0.0016	0.0043	*0.0034*	0.0025	*0.0018*	0.0022	*0.0019*	0.0025	*0.0020*	0.0029	*0.0020*

sampling over standard ones using NMI as measure. For the monotonic RUS and MWMOTE comparisons (Fig. 4a and e), the majority of the distribution is located in the right part. For the rest, the distributions are more or less shared by the rope and right sections. However, for monotonic ROS (Fig. 4b), ADASYN (Fig. 4d) and MWMOTE (Fig. 4e), there are barely no points on the left side, therefore their superiority is significant.

Recapitulating, the five monotonic sampling techniques based on monotonic chains increase the predictability of the classifier and, at the same time, to a large extent, improve the standard sampling approaches in terms of monotonicity. mRUS and mMWMOTE show a marked improvement as compared to the standard versions.

Among the monotonic sampling methods analyzed, it is difficult to choose the best technique, since the best results strongly depend on the characteristics of the problem and classifier used. On average, mADASYN results in the best techniques in terms of MAvA, while mROS achieves the best monotonic results. mRUS is the worst on average, but it obtains fairly good results for those datasets where there is a good amount of representatives of the minority classes, such as *car* and *wisconsin*, especially when combined with OLM and MonMLP. mROS is very interesting for problems with repeated samples of different classes. Comparing the methods based on linear interpolation, mSMOTE is the best at maintaining monotonicity and a good trade-off between predictability improvement and monotonicity maintenance.

6. Concluding remarks

The imbalanced class problem is a real issue for many real-life applications and commonly used data-sets in classification with monotonic constraints. However, as shown empirically in this paper, traditional solutions are not valid for those scenarios in which monotonic constraints are assumed, since these methods heavily degrade the monotonicity of the data-sets.

In this paper, a new sampling scheme based on monotonic chains was designed to consider the constraints of monotonicity and to be able to be applied to monotonic data-sets. The main objective was to improve the accuracy of those minority classes, while preserving the monotonicity of the models. We have developed this scheme within 5 different well-known sampling techniques, one under-sampling and four over-sampling algorithms. These new methods have been empirically tested with their standard versions, statistically resulting in the same results in terms of prediction enhancement and in much better preservation of the monotonicity of the predictions. In some cases, they have even improved the monotonicity of the models compared to the original without sampling. Among the chain based sampling methods, mROS is the best in terms of monotonicity and mSMOTE is the best in predictability improvement and monotonicity preservation.

Acknowledgements

This work was supported by the Spanish National Research Project TIN2017-89517-P and the Project BigDaP-TOOLS - Ayudas Fundación BBVA a Equipos de Investigación Científica 2016 and by a research scholarship (FPU) given to the author Sergio González by the Spanish Ministry of Education, Culture and Sports. Additionally, this paper is the result of a collaboration with the Prof. Sheng-Tun Li during the international stay made by Sergio González at National Cheng Kung University. This stay was partially supported by the 2017 Summer Program in Taiwan for Spanish Graduate Students held by the Ministry of Science and Technology of Taiwan (R.O.C.).

Appendix A

Table A.10
MAvA results obtained by the selected classifiers with standard and ordinal sampling.

		Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
	balance	0.6343	0.7241	0.6100	0.6436	0.6376	0.6376	0.6637	0.6637	0.6657
	car	0.9134	0.8824	0.9553	0.9696	0.9709	0.9709	0.9660	0.9636	0.9645
	ERA	0.1357	0.1480	0.1338	0.2501	0.2794	0.2588	0.2845	0.2769	0.2911
	ESL	0.4674	0.3986	0.4621	0.4660	0.4711	0.3228	0.4945	0.4915	0.5123
MKNN	*LEV*	0.3924	0.4182	0.3834	0.4682	0.4737	0.3925	0.4467	0.4414	0.4491
	SWD	0.4004	0.4692	0.4111	0.4608	0.4616	0.4599	0.4034	0.4031	0.4044
	windsorhousing	0.6755	0.6467	0.6755	0.6562	0.6477	0.6477	0.6726	0.6726	0.6526
	wisconsin	0.9625	0.9745	0.9625	0.9699	0.9721	0.9721	0.9709	0.9709	0.9702
	Avg:	0.5727	0.5827	0.5742	0.6106	0.6143	0.5828	0.6128	0.6105	0.6137
	balance	0.6019	0.5243	0.6019	0.6268	0.6200	0.6200	0.6632	0.6632	0.6577
	car	0.8704	0.8424	0.9028	0.8885	0.8885	0.8885	0.8885	0.8909	0.8885
	ERA	0.2423	0.2276	0.2674	0.2962	0.3012	0.2983	0.2716	0.2794	0.2550
	ESL	0.3981	0.3242	0.4070	0.3974	0.4019	0.4069	0.4225	0.4203	0.4253
OLM	*LEV*	0.4044	0.3801	0.3729	0.4165	0.4075	0.4054	0.4155	0.4109	0.4077
	SWD	0.4834	0.4024	0.4707	0.4834	0.4834	0.4834	0.4830	0.4834	0.4834
	windsorhousing	0.5320	0.6358	0.6352	0.6444	0.6324	0.6324	0.6487	0.6487	0.6293
	wisconsin	0.8418	0.8814	0.8668	0.8567	0.8821	0.8821	0.8551	0.8551	0.8502
	Avg:	0.5468	0.5273	0.5656	0.5762	0.5771	0.5771	0.5810	0.5815	0.5746
	balance	0.5705	0.5531	0.5941	0.5849	0.5674	0.5674	0.5755	0.5755	0.5621
	car	0.4603	0.8191	0.8810	0.8374	0.8786	0.8683	0.8222	0.8422	0.8723
	ERA	0.3182	0.2907	0.3040	0.3059	0.2926	0.2997	0.2867	0.3044	0.2929
MID	*ESL*	0.4254	0.3213	0.4384	0.4411	0.4608	0.4638	0.4372	0.4430	0.4179
	LEV	0.4731	0.4592	0.5732	0.5367	0.5101	0.5226	0.5157	0.4981	0.4947
	SWD	0.4470	0.4676	0.5542	0.5192	0.4993	0.4441	0.4542	0.4689	0.4566
	windsorhousing	0.7000	0.6912	0.7165	0.6642	0.7064	0.6489	0.6868	0.6966	0.6741
	wisconsin	0.9524	0.9522	0.9566	0.9619	0.9678	0.9490	0.9651	0.9607	0.9593
	Avg:	0.5434	0.5693	0.6272	0.6064	0.6104	0.5955	0.5929	0.5987	0.5912
	balance	0.5159	0.5130	0.5299	0.3980	0.3965	0.3965	0.5106	0.5106	0.5181
	car	0.9179	0.4976	0.4594	0.4648	0.4638	0.4621	0.4562	0.4615	0.4496
	ERA	0.2525	0.2005	0.1990	0.1988	0.2000	0.1987	0.1878	0.1923	0.1967
	ESL	0.4934	0.1470	0.1429	0.1618	0.1612	0.1618	0.1560	0.1560	0.1560
OSDL	*LEV*	0.4936	0.3532	0.3597	0.3538	0.3545	0.3791	0.3594	0.3606	0.3318
	SWD	0.4588	0.4515	0.4497	0.4344	0.4359	0.4368	0.4334	0.4334	0.4311
	windsorhousing	0.6334	0.5854	0.5400	0.5480	0.5424	0.5424	0.5424	0.5424	0.5392
	wisconsin	0.9617	0.9211	0.9211	0.9211	0.9211	0.9211	0.9211	0.9211	0.9211
	Avg:	0.5909	0.4587	0.4502	0.4351	0.4344	0.4373	0.4459	0.4472	0.4429
	balance	0.9051	0.8523	0.9114	0.8837	0.8918	0.8918	0.8291	0.8291	0.8258
	car	0.6487	0.7788	0.7326	0.7754	0.7564	0.8133	0.6941	0.7508	0.7552
	ERA	0.1723	0.1957	0.2887	0.3084	0.3108	0.2927	0.2886	0.3023	0.2867
	ESL	0.5370	0.4503	0.4941	0.1111	0.1111	0.1111	0.1325	0.1305	0.2985
MonMLP	*LEV*	0.4839	0.5077	0.5259	0.5561	0.5367	0.5029	0.5451	0.5403	0.5370
	SWD	0.3497	0.4247	0.4740	0.4679	0.4842	0.5270	0.4415	0.4431	0.4419
	windsorhousing	0.5566	0.6637	0.6544	0.6562	0.6359	0.6359	0.6595	0.6595	0.5000
	wisconsin	0.8118	0.9537	0.9588	0.9628	0.9648	0.9648	0.9611	0.9611	0.9221
	Avg:	0.5582	0.6034	0.6300	0.5902	0.5865	0.5924	0.5689	0.5771	0.5709
	Complete Avg:	0.5624	0.5620	0.5694	0.5637	0.5645	0.5570	0.5603	0.5630	0.5587

Table A.11
NMI results obtained by the selected classifiers with standard and ordinal sampling.

		Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
	balance	0.0001	0.0006	0.0003	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
	car	0.0000	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	ERA	0.0056	0.0081	0.0071	0.0113	0.0146	0.0172	0.0118	0.0130	0.0106
	ESL	0.0012	0.0072	0.0014	0.0013	0.0015	0.0125	0.0011	0.0011	0.0012
MKNN	*LEV*	0.0010	0.0066	0.0016	0.0019	0.0018	0.0116	0.0019	0.0019	0.0024
	SWD	0.0005	0.0043	0.0008	0.0007	0.0007	0.0007	0.0011	0.0011	0.0010
	windsorhousing	0.0000	0.0002	0.0000	0.0001	0.0001	0.0001	0.0000	0.0000	0.0053
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg:	0.0011	0.0034	0.0014	0.0020	0.0024	0.0053	0.0020	0.0022	0.0026
	balance	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0003	0.0003	0.0003
	car	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	ERA	0.0063	0.0040	0.0100	0.0056	0.0058	0.0051	0.0064	0.0057	0.0059
	ESL	0.0025	0.0084	0.0023	0.0023	0.0023	0.0023	0.0018	0.0018	0.0023
OLM	*LEV*	0.0043	0.0038	0.0028	0.0039	0.0028	0.0027	0.0051	0.0053	0.0053
	SWD	0.0015	0.0062	0.0016	0.0015	0.0015	0.0015	0.0016	0.0015	0.0015
	windsorhousing	0.0000	0.0000	0.0000	0.0002	0.0001	0.0001	0.0002	0.0002	0.0004
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg:	0.0018	0.0028	0.0021	0.0017	0.0016	0.0015	0.0019	0.0019	0.0020
	balance	0.0017	0.0056	0.0015	0.0012	0.0016	0.0016	0.0025	0.0025	0.0028
	car	0.0034	0.0050	0.0014	0.0013	0.0012	0.0008	0.0014	0.0012	0.0014
	ERA	0.0086	0.0064	0.0115	0.0087	0.0101	0.0103	0.0106	0.0102	0.0103
MID	*ESL*	0.0023	0.0205	0.0064	0.0041	0.0038	0.0030	0.0044	0.0033	0.0038
	LEV	0.0024	0.0116	0.0037	0.0027	0.0031	0.0031	0.0025	0.0025	0.0030
	SWD	0.0020	0.0089	0.0026	0.0027	0.0029	0.0027	0.0024	0.0020	0.0022
	windsorhousing	0.0035	0.0135	0.0127	0.0094	0.0135	0.0188	0.0094	0.0138	0.0057
	wisconsin	0.0001	0.0001	0.0000	0.0001	0.0000	0.0002	0.0000	0.0000	0.0001
	Avg:	0.0030	0.0090	0.0050	0.0038	0.0045	0.0051	0.0042	0.0044	0.0037
	balance	0.0006	0.0016	0.0000	0.0058	0.0060	0.0060	0.0022	0.0022	0.0015
	car	0.0000	0.0019	0.0025	0.0034	0.0034	0.0033	0.0057	0.0052	0.0035
	ERA	0.0049	0.0009	0.0301	0.0310	0.0299	0.0274	0.0251	0.0302	0.0326
	ESL	0.0006	0.0000	0.0963	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
OSDL	*LEV*	0.0004	0.0051	0.0010	0.0005	0.0006	0.0009	0.0011	0.0068	0.0163
	SWD	0.0009	0.0030	0.0009	0.0009	0.0009	0.0009	0.0008	0.0008	0.0008
	windsorhousing	0.0036	0.0000	0.0002	0.0001	0.0002	0.0002	0.0002	0.0002	0.0002
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg:	0.0009	0.0020	0.0164	0.0052	0.0051	0.0048	0.0044	0.0057	0.0069
	balance	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0003	0.0003	0.0002
	car	0.0001	0.0006	0.0006	0.0003	0.0003	0.0002	0.0002	0.0002	0.0001
	ERA	0.0026	0.0050	0.0048	0.0063	0.0069	0.0069	0.0056	0.0071	0.0111
	ESL	0.0003	0.0030	0.0016	0.0000	0.0000	0.0000	0.0196	0.0190	0.0426
MonMLP	*LEV*	0.0008	0.0041	0.0027	0.0023	0.0023	0.0035	0.0020	0.0019	0.0035
	SWD	0.0004	0.0044	0.0011	0.0013	0.0021	0.0022	0.0011	0.0011	0.0008
	windsorhousing	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0022
	Avg:	0.0005	0.0021	0.0013	0.0013	0.0015	0.0016	0.0036	0.0037	0.0076
	Complete Avg:	0.0015	0.0040	0.0052	0.0028	0.0030	0.0037	0.0032	0.0036	0.0045

References

[1] J. Alcalá-Fdez, R. Alcalá, S. González, Y. Nojima, S. García, Evolutionary fuzzy rule-based methods for monotonic classification, IEEE Trans. Fuzzy Syst. 25 (6) (2017) 1376–1390.
[2] S. Barua, M.M. Islam, X. Yao, K. Murase, Mwmote–majority weighted minority oversampling technique for imbalanced data set learning, IEEE Trans. Knowl. Data Eng. 26 (2) (2014) 405–425.
[3] A. Ben-David, Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications, Decis. Sci. 23 (1992) 1357–1372.
[4] A. Ben-David, Monotonicity maintenance in information-theoretic machine learning algorithms, Mach. Learn. 19 (1) (1995) 29–43.
[5] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, J. Mach. Learn. Res. 18 (1) (2017) 2653–2688.
[6] J.-R. Cano, N.R. Aljohani, R.A. Abbasi, J.S. Alowidbi, S. Garcia, Prototype selection to improve monotonic nearest neighbor, Eng. Appl. Artif. Intell. 60 (2017) 128–135.
[7] J.-R. Cano, S. García, Training set selection for monotonic ordinal classification, Data Knowl. Eng. 112 (2017) 94–105.
[8] J. Carrasco, S. García, M. del Mar Rueda, F. Herrera, rNPBST: an R package covering non-parametric and bayesian statistical tests, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 281–292.
[9] L. Cerf, D. Gay, N. Selmaoui-Folcher, B. Crémilleux, J.-F. Boulicaut, Parameter-free classification in multi-class imbalanced data sets, Data Knowl. Eng. 87 (2013) 109–129.
[10] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. (2002) 321–357.

[11] N.V. Chawla, D.A. Cieslak, L.O. Hall, A. Joshi, Automatically countering imbalance and its empirical relationship to cost, Data Min. Knowl. Discov. 17 (2) (2008) 225–252.
[12] N.V. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, ACM SIGKDD Explor. Newslett. 6 (1) (2004) 1–6.
[13] C.-C. Chen, S.-T. Li, Credit rating with a monotonicity-constrained support vector machine model, Expert Syst. Appl. 41 (16) (2014) 7235–7247.
[14] S. Datta, S. Das, Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs, Neural Netw. 70 (2015) 39–52.
[15] K. Dembczyński, W. Kotłowski, R. Słowiński, Learning rule ensembles for ordinal classification with monotonicity constraints, Fundam. Inform. 94 (2) (2009) 163–178.
[16] W. Duivesteijn, A. Feelders, Nearest neighbour classification with monotonicity constraints, in: ECML/PKDD (1), 2008, pp. 301–316.
[17] A. Feelders, Monotone relabeling in ordinal classification, in: ICDM, IEEE Computer Society, 2010, pp. 803–808.
[18] A. Fernández, V. López, M. Galar, M.J. del Jesus, F. Herrera, Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches, Knowl. Based Syst. 42 (2013) 97–110.
[19] F. Fernández-Navarro, A. Riccardi, S. Carloni, Ordinal neural networks without iterative tuning, IEEE Trans. Neural Netw. Learn. Syst. 25 (11) (2014) 2075–2085.
[20] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701.
[21] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) 1761–1776.
[22] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst., Man, Cybern., Part C 42 (4) (2012) 463–484.
[23] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, NMC: nearest matrix classification–a new combination model for pruning one-vs-one ensembles by transforming the aggregation problem, Inf. Fusion 36 (2017) 26–51.
[24] J. García, A.M. AlBar, N.R. Aljohani, J.-R. Cano, S. García, Hyperrectangles selection for monotonic classification by using evolutionary algorithms, Int. J. Comput. Intell. Syst. 9 (1) (2016) 184–202.
[25] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[26] S. González, F. Herrera, S. García, Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity, New Gener. Comput. 33 (4) (2015) 367–388.
[27] H. He, Y. Bai, E.A. Garcia, S. Li, Adasyn: adaptive synthetic sampling approach for imbalanced learning, in: Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, IEEE, 2008, pp. 1322–1328.
[28] Q. Hu, W. Pan, Y. Song, D. Yu, Large-margin feature selection for monotonic classification, Knowl. Based Syst. 31 (2012) 8–18.
[29] J. Huang, C.X. Ling, Using AUC and accuracy in evaluating learning algorithms, IEEE Trans. Knowl. Data Eng. 17 (3) (2005) 299–310.
[30] W. Kotłowski, R. Słowiński, On nonparametric ordinal classification with monotonicity constraints, IEEE Trans. Knowl. Data Eng. 25 (11) (2013) 2576–2589.
[31] B. Lang, Monotonic multi-layer perceptron networks as universal approximators, in: International Conference on Artificial Neural Networks, Springer, 2005, pp. 31–37.
[32] F. Li, X. Zhang, X. Zhang, C. Du, Y. Xu, Y.-C. Tian, Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets, Inf. Sci. 422 (2018) 242–256.
[33] S.-T. Li, C.-C. Chen, A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1713–1727.
[34] S. Lievens, B. De Baets, Supervised ranking in the weka environment, Inf. Sci. 180 (24) (2010) 4763–4771.
[35] S. Lievens, B. De Baets, K. Cao-Van, A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting, Ann. Oper. Res. 163 (1) (2008) 115–142.
[36] Y. Liu, L. Nie, L. Han, L. Zhang, D.S. Rosenblum, Action2activity: Recognizing complex activities from sensor data., in: IJCAI, 2015, 2015, pp. 1617–1623.
[37] Y. Liu, L. Zhang, L. Nie, Y. Yan, D.S. Rosenblum, Fortune teller: predicting your career path., in: AAAI, 2016, 2016, pp. 201–207.
[38] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141.
[39] C. Marsala, D. Petturiti, Rank discrimination measures for enforcing monotonicity in decision tree induction, Inf. Sci. 291 (2015) 143–171.
[40] I. Nekooeimehr, S.K. Lai-Yuen, Cluster-based weighted oversampling for ordinal regression (cwos-ord), Neurocomputing 218 (2016) 51–60.
[41] S. Pei, Q. Hu, Partially monotonic decision trees, Inf. Sci. 424 (2018) 104–117.
[42] M. Pérez-Ortiz, P.A. Gutierrez, C. Hervás-Martínez, X. Yao, Graph-based approaches for over-sampling in the context of ordinal regression, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1233–1245.
[43] R. Potharst, A. Ben-David, M.C. van Wezel, Two algorithms for generating structured and unstructured monotone ordinal data sets, Eng. Appl. Artif. Intell. 22 (4–5) (2009) 491–496.
[44] R.C. Prati, G.E.A.P.A. Batista, D.F. Silva, Class imbalance revisited: a new experimental setup to assess the performance of treatment methods, Knowl. Inf. Syst. 45 (1) (2015) 247–270.
[45] Y. Qian, H. Xu, J. Liang, B. Liu, J. Wang, Fusing monotonic decision trees, IEEE Trans. Knowl. Data Eng. 27 (10) (2015) 2717–2728.
[46] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, F. Herrera, Ifrowann: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1622–1637.
[47] J.P. Sánchez-Crisostomo, R. Alejo, E. López-González, R.M. Valdovinos, J.H. Pacheco-Sánchez, Empirical analysis of assessments metrics for multi-class imbalance learning on the back-propagation context, in: Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Springer, 2014, pp. 17–23.
[48] J. Sun, J. Lang, H. Fujita, H. Li, Imbalanced enterprise credit evaluation with dte-sbd: decision tree ensemble based on smote and bagging with differentiated sampling rates, Inf. Sci. 425 (2018) 76–91.
[49] I. Triguero, S. González, J.M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M.J. del Jesús, L. Sánchez, F. Herrera, Keel 3.0: an open source software for multi-stage analysis in data mining, Int. J. Comput. Intell. Syst. 10 (1) (2017) 1238–1249.
[50] M. Velikova, H. Daniels, Decision trees for monotone price models, Comput. Manag. Sci. 1 (3) (2004) 231–244.
[51] Z.-L. Zhang, X.-G. Luo, S. González, S. García, F. Herrera, DRCW-ASEG: One-versus-one distance-based relative competence weighting with adaptive synthetic example generation for multi-class imbalanced datasets, Neurocomputing 285 (2018) 176–187.
[52] H. Zhu, E.C. Tsang, X.-Z. Wang, R.A.R. Ashfaq, Monotonic classification extreme learning machine, Neurocomputing 225 (2017) 205–213.