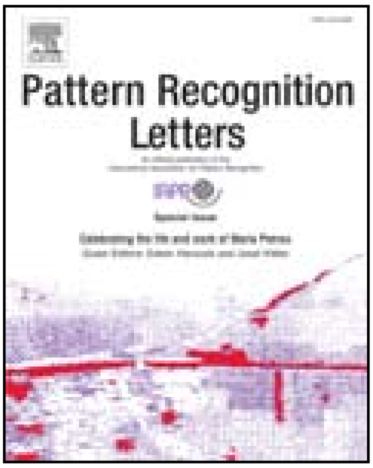


Oversampling imbalanced data in the string space

Francisco J. Castellanos, Jose J. Valero-Mas,
Jorge Calvo-Zaragoza, Juan R. Rico-Juan

PII: S0167-8655(18)30003-5
DOI: 10.1016/j.patrec.2018.01.003
Reference: PATREC 7047

To appear in: *Pattern Recognition Letters*

Received date: 16 June 2017
Revised date: 16 October 2017
Accepted date: 4 January 2018

Please cite this article as: Francisco J. Castellanos, Jose J. Valero-Mas, Jorge Calvo-Zaragoza, Juan R. Rico-Juan, Oversampling imbalanced data in the string space, *Pattern Recognition Letters* (2018), doi: 10.1016/j.patrec.2018.01.003

Highlights

- Oversampling in the string space for addressing imbalanced classification
- Generating new strings between pairs of instances using the Edit distance
- Experimentation with contour representations of handwritten digits and characters
- Statistical performance improvement of the classifier with respect to imbalanced case

Pattern Recognition Letters
journal homepage: www.elsevier.com

Oversampling imbalanced data in the string space

Francisco J. Castellanos[a], Jose J. Valero-Mas[a], Jorge Calvo-Zaragoza[a,**], Juan R. Rico-Juan[a]

[a]*Pattern Recognition and Artificial Intelligence Group, Department of Software and Computing Systems, University of Alicante, 03690 Alicante, Spain*

ABSTRACT

Imbalanced data is a typical problem in the supervised classification field, which occurs when the different classes are not equally represented. This fact typically results in the classifier biasing its performance towards the class representing the majority of the elements. Many methods have been proposed to alleviate this scenario, yet all of them assume that data is represented as feature vectors. In this paper we propose a strategy to balance a dataset whose samples are encoded as strings. Our approach is based on adapting the well-known Synthetic Minority Over-sampling Technique (SMOTE) algorithm to the string space. More precisely, data generation is achieved with an iterative approach to create artificial strings within the segment between two given samples of the training set. Results with several datasets and imbalance ratios show that the proposed strategy properly deals with the problem in all cases considered.

1. Introduction

Within the Pattern Recognition (PR) field, supervised classification aims at categorizing unknown elements considering a set of labeled examples. This particular task has been largely addressed due to its applicability in disparate duties such as text recognition (Plamondon and Srihari, 2000), audio music analysis (McVicar et al., 2014) or image categorization (Ciregan et al., 2012), among many others. As a consequence, many algorithms have been proposed, being some representative examples the *k*-Nearest Neighbor (Cover and Hart, 1967), Support Vector Machines (Platt, 1999) or Random Forest (Breiman, 2001).

In general, classification systems show a great dependency with respect to the representation considered for the data. In this regard, two main families of approaches are typically distinguished (Duda et al., 2001): *statistical representations*, in which the data is encoded as numerical feature vectors, and *structural representations*, which codify data as symbolic data structures, that is strings, trees or graphs. Statistical representations depict a limited flexibility in terms of representation but they are addressable by most PR models (Bunke and Riesen, 2012). On the contrary, structural representations are constrained in their applicability since not all classification schemes can process them, but they constitute very powerful abstractions capable of properly representing high-level relationships. Within the structural representations, the *string space* (samples that consist of sequences of characters) is considered in many PR applications such as chain-code-based classification (Lee and Kim, 2015), signature verification (Fischer and Plamondon, 2017), music genre recognition (Yoon et al., 2016), or time series classification (Marteau and Gibet, 2015), among others. A significant advantage of this representation is the existence of the well-known Edit distance, which represents a suitable distance metric for classification (Gottlieb et al., 2014).

Regardless of the type of representation considered, most classification schemes assume the ideal situation of having an equal distribution of the classes. Nevertheless, real-world situations do not generally show that trend, which typically results in the classifier biasing its performance to the class representing the majority of the elements (He and Garcia, 2009). In many applications, as for instance in the medical field (Yu and Ni, 2014; Tesfahun and Bhaskari, 2013), data collections generally show imbalanced distributions in which the class represented by a scarce amount of examples (the minority class) is indeed the important one to predict. Thus, a large number of paradigms and methods for tackling the class imbalance problem have been studied, many of which will be introduced in next section.

[**]Corresponding author: Tel.: +349-65-903772; Fax: +349-65-909326
e-mail: `jcalvo@dlsi.ua.es` (Jorge Calvo-Zaragoza)

A possible solution to tackle this issue is to artificially create new examples from the minority class to compensate the class distribution. However, the main problem with such approach is that balancing algorithms only work with statistical data, *ie.*, represented by feature vectors. Hence, when addressing cases considering structural representations, the data must be mapped onto a vector space before the balancing stage. While this solution may somehow solve the aforementioned limitation of the data-balancing methods, these mapping processes generally entails a decrease in the classification performance (Bunke and Riesen, 2012). There is thus a need for designing new balancing methods which can be directly applied to structural data.

In this paper we address the aforementioned issue of imbalanced classification in the particular context of the string space. For that, we propose a new technique for artificially equilibrating imbalanced data collections encoded as strings without the need of an intermediate embedding stage to map the data onto a statistical representation. Designing balancing techniques which are directly applicable to string data is not trivial due to the limitations found in mathematical operations for strings (e.g., obtaining the mean of two string data). In this context we take as starting point existing algorithms that address such type of operations as, for instance, the method by Abreu and Rico-Juan (2013), which generates an equidistant string between two examples using the Levenshtein distance.

The rest of the paper is structured as follows: Section 2 contextualizes the problem of imbalanced classification highlighting the most known techniques to tackle it; Section 3 explains the new proposed technique to artificially equilibrate imbalanced string-based data collections; Section 4 describes the experimental methodology considered for assessing the proposal; Section 5 presents and analyzes the results obtained; finally, Section 6 summarizes the general conclusions obtained and discusses possible future work.

2. Background

This section describes the issue of classification in imbalanced scenarios and introduces some of the most well-known approaches for tackling it. Finally, this concept is extended and contextualized for the case of structural data, which constitutes the particular case of study of this work.

As discussed above, imbalanced data is commonly found in real-life data collections. Nevertheless, conventional classification models are not generally prepared for such cases, which results in schemes in which the classifier consistently categorizes new instances as belonging to the majority class, thus neglecting the minority ones. To palliate this undesired consequence, different strategies may be found in the literature, being generally grouped in two main categories (He and Garcia, 2009): *algorithmic-level* approaches and *data-level* methods.

Algorithmic-level methods modify the actual behavior of the classifier so that it compensates the asymmetry in the data instead of assuming a class-balanced problem. This is typically done using cost-sensitive training strategies, which consider higher penalties for the misclassifications of the data belonging to the minority class than for the rest. However, this family of methods is totally dependent on the learning algorithms considered, thus being these techniques hardly generalizable.

Data-level approaches compensate the class distribution by directly modifying the dataset at issue. This is generally carried out by reducing the population of the majority class and/or increasing the population of the minority one.

On the one hand, reducing the population of the majority class, also known as *undersampling*, depicts the clear advantage of being independent on the type of data considered as it simply implies the elimination of examples from the dataset. However, this process may entail some loss of critical information for the classification. Classical approaches in this context are the One-Sided Selection (Kubat and Matwin, 1997), the Tomek Links (Tomek, 1976), and the Neighborhood Cleaning Rule (Laurikkala, 2001), among others.

On the other hand, increasing the population of the minority class, which is known as *oversampling*, does not generally imply any loss of information since no elements are discarded. However, this process is more restrictive than the previous one as generating new elements is totally dependent on the representation considered for the data. Also note that including new artificial elements in the data collection may imply an increase in the cost of the classification process, especially for instance-based classifiers as k-Nearest Neighbor (kNN); nevertheless this should not represent a practical problem since there exist successful algorithms for efficiently performing the classification task (Cherian et al., 2014; Almalawi et al., 2016). Among the different existing methods, the Synthetic Minority Over-sampling Technique (SMOTE) algorithm (Chawla et al., 2002) stands as one of the most well-known and successful algorithms. It aims at alleviating the issues with imbalanced data collections by populating the space around the instances of the minority class with new artificial samples from the same class. Due to its reported success, some extensions have been proposed such as the SMOTE-Borderline 1 and SMOTE-Borderline 2 by Han et al. (2005), which focus on performing the population process on the decision boundaries among the different classes. Other existing alternatives are the work by Jo and Japkowicz (2004), which proposes an oversampling method based on clustering, or the Adaptive Synthetic Sampling Approach for Imbalanced Learning (ADASYN) by He et al. (2008), which progressively adapts the level of oversampling of the minority class depending on the difficulty of the classifier to learn the data distribution.

More recently, Das et al. (2015) proposed the RApidy COnverging Gibbs (RACOG) algorithm, which considers a probabilistic framework that oversamples the minority class using a Markov chain Monte Carlo strategy. Abdi and Hashemi (2016) presented a method for oversampling in multi-class imbalanced problems. Their method generates new samples based on the Mahalanobis distance to class means. In the work of Zhu et al. (2017), an oversampling technique for multi-class imbalance tasks is proposed, which weighs the directions of the minority class samples to generate new prototypes accordingly. In addition to these advances, note that oversampling methods may be complemented with classical undersampling techniques (Valero-Mas et al., 2017; Junsomboon and Phienthrakul, 2017).

For all above, oversampling methods stand as an appealing alternative for dealing with the class-imbalance problem without modifying the classification algorithm (Amin et al., 2016). Typically, these approaches have been constrained to statistical data representations because the operations involved (basic arithmetic operations such as mean or median) are not trivial in a structural space. Thus, in this work we address this shortage by proposing a new oversampling approach for tackling data imbalance directly in structural representations, and more precisely string data. We shall consider the SMOTE algorithm as base idea, as it still constitutes one of the most successful methods for oversampling in class-imbalance classification tasks (Bach et al., 2017).

3. Oversampling imbalanced data in the String Space

Oversampling methods for balancing class distributions do not usually imply any loss of classification performance as no information is discarded. Nevertheless, in general these methods are only applicable to statistical data representations. In this regard, this section proposes a new method for equilibrating imbalanced data collections for the particular case in which the elements are encoded as strings.

As the starting point of our proposal we consider the initial SMOTE algorithm. Let us assume a two-class training set $\mathcal{T}$ that may be further divided in two subsets $\mathcal{T}_{\text{MAJ}}$ and $\mathcal{T}_{\text{MIN}}$ representing the instances of the majority and minority classes, respectively. Basically, SMOTE populates the minority class by adding new instances to $\mathcal{T}_{\text{MIN}}$.

More specifically, SMOTE takes as input a parameter N that indicates the number of prototypes to be generated. For each prototype $p \in \mathcal{T}_{\text{MIN}}$, the algorithm randomly selects N of its k-nearest neighbors within $\mathcal{T}_{\text{MIN}}$. Then, new prototypes are artificially generated, which are located at any place of the segment between p and each of the selected neighbors. The new prototypes are finally included in the minority class set $\mathcal{T}_{\text{MIN}}$.

As it is, the SMOTE algorithm has no special requirements, only that a distance between pairs of prototypes of the input space can be defined to be able to retrieve the nearest k-neighbors. However, note that the operation *"generate a prototype that is located in the segment between two prototypes"* is straightforward for feature vectors, but it is not trivial in a structured space like the string one.

In this work, we define the geometric concept of *"point in the segment between two prototypes"* as the point for which the sum of the distances to such prototypes is equal to the distance between those prototypes. Since the distance considered has an important relevance on the development of our strategy, we shall consider the use of the Edit Distance (Wagner and Fischer, 1974). For a given pair of data encoded as strings, this distance is defined as the minimum number of editing operations (insertions, substitutions or deletions) to transform one of the strings into the other.

For the actual data generation, our strategy makes use of the algorithm proposed by Abreu and Rico-Juan (2013). This method is able to retrieve a point in the segment between two string prototypes by means of randomly deciding which of the editing operations needed to transform one string into another is eventually applied.

First, we assume that given two strings v and w that might be of different lengths, we have a function GetSecuenceOfTransformations(v, w) that returns the sequence of edit operations required to transform v into w. This can be efficiently performed with a dynamic programming approach reaching lineal complexity respect of the size of the strings.

Considering ϵ as the empty character, we can define all the mentioned editing operations as substitutions: an insertion equals to a substitution of ϵ by the new character, and a deletion is equivalent to a substitution of the character by ϵ. Let $e(a, b)$ be the operation *"substitute character a by character b"*, which nicely generalizes to *insertion* as $e(\epsilon, b)$ and *deletion* as $e(a, \epsilon)$. Note that a match is produced when $a = b$, which represents an edit operation with no cost.

Considering these operations, a new string located in the segment between v and w can be obtained by following Algorithm 1. The algorithm receives two strings v and w with the goal of generating a new artificial string s located somewhere in the segment between v and w. It also takes a threshold within the range $[0, 1]$, indicating to which of the reference strings the generated one is more likely to be closer. That is, a threshold above 0.5 will generate strings closer to v, being closer to w otherwise.

The complete process consists of the following steps:

(i) Get the sequence of transformations W required to transform v into w. There might be more than one solution, so we assume that any of them is returned (line 2).

(ii) Initialize the sequence of transformations W' that will generate s from v (line 3).

(iii) Iterate over each edit operation of W (line 4). If the operation is a match, it is directly added to W' (lines 5-6). Otherwise, it is randomly decided—according to the threshold (line 8)—if the operation is either substituted by a match (line 9) or maintained (line 11). Note that if the operation is maintained, s gets closer to w.

(iv) When all transformations have been checked, W' is applied to v to get the generated string s.

Let us describe a running example to understand its operation. Let us consider the strings $v = 223697$ and $w = 246985$. Then, the sequence W of edit operations to transform the string v into w (returned by GetSecuenceOfTransformations(v, w)) can be defined by $e(2, \epsilon)$ $e(2, 2)$ $e(3, 4)$ $e(6, 6)$ $e(9, 9)$ $e(7, 8)$ $e(\epsilon, 5)$. The distance is 4, which results from counting the length of the sequence but ignoring matches.

After getting the sequence of transformations, it is easy to obtain a string in the segment between 223697 and 246985 by randomly replacing insertions, deletions or substitutions by matches. For instance, if we skip the first and last edit operation by a match, the result is

Algorithm 1: Generation of a new string prototype in the segment between two given strings.

```
Algorithm CreateNewString($v$, $w$, $t$)
    Data: $v$, $w$: strings
    Data: $t \in [0, 1]$: threshold
    Result: $s$: string in the segment between $v$ and $w$
    $W \leftarrow$ GetSecuenceOfTransformations($v$, $w$)
    $W' \leftarrow ()$
    foreach $e(a, b) \in W$ do
        if $a = b$ then
            Append($W'$, $e(a, b)$)
        else
            if Random() $\leq t$ then
                Append($W'$, $e(a, a)$)
            else
                Append($W'$, $e(a, b)$)
            end
        end
    end
    $s \leftarrow$ ApplyTransformations($v$, $W'$)
    return $s$
end
```

Algorithm 2: Scheme of the SMOTE algorithm over the string space.

```
1  Algorithm SMOTE-String($\mathcal{T}_{MIN}$, $\mathcal{T}_{MAJ}$, $N$, $t$)
       Data: $\mathcal{T}_{\text{MIN}}$: prototypes from the minority class
       Data: $\mathcal{T}_{\text{MAJ}}$: prototypes from the majority class
       Data: $k$: neighborhood considered for generating new
             prototypes
       Data: $N$: number of new instances per prototype
       Data: $t \in [0, 1]$: threshold
       Result: $\mathcal{T}_{\text{SMOTE}}$: balanced training set
2      foreach $p \in \mathcal{T}_{MIN}$ do
3          $\mathcal{T}_{\text{kNN}} \leftarrow$ kNN($p$,$\mathcal{T}_{\text{MIN}}$, $k$)
4          for $i$=1 to $N$ do
5              $v \leftarrow$ RandomSelect ($\mathcal{T}_{\text{kNN}}$)
6              $s \leftarrow$CreateNewString($p$,$v$,$t$)
7              $\mathcal{T}_{\text{SMOTE}} \leftarrow \mathcal{T}_{\text{SMOTE}} \cup \{s\}$
8          end
9          $\mathcal{T}_{\text{SMOTE}} \leftarrow \mathcal{T}_{\text{SMOTE}} \cup \{p\}$
10     end
11     $\mathcal{T}_{\text{SMOTE}} \leftarrow \mathcal{T}_{\text{SMOTE}} \cup \mathcal{T}_{\text{MAJ}}$
12     return $\mathcal{T}_{SMOTE}$
13 end
```

$e(2,2)$ $e(2,2)$ $e(3,4)$ $e(6,6)$ $e(9,9)$ $e(7,8)$ $e(2,5)$, which applied to the first string gives `2246985`. Note that this resulting string is of distance 3 to `223697` and of distance 1 to `2246985`.

Finally, if we combine everything explained before, we can balance datasets in the string space following Algorithm 2, which considers separately the initial prototypes of the training set belonging to the majority class and the minority class. It iterates over the latter set, and for each of these prototypes, the k-nearest neighbors are retrieved. Then, it generates N new string prototypes located between the considered prototype and one of its nearest neighbors (selected randomly) using the `CreateNewString()` method explained above. The new generated prototype is added to the final training set. Once N prototypes have been generated, the reference prototype is also included in the final training set. Eventually, when all prototypes from the minority class have been consulted, the process returns the balanced training set which comprises the prototypes from the minority class (both original and generated) and those belonging to the majority class.

3.1. *Extension to Borderline 1 and Borderline 2*

In addition to the classic SMOTE approach, we are also considering its two extensions Borderline 1 (B1) and Borderline 2 (B2). Instead of a general populating approach of the minority class, these variants focus on detecting and oversampling regions close to the boundaries between classes. For that, these methods initially obtain a set $\mathcal{T}_{\text{DANGER}}$ that represents the information located in the frontier between $\mathcal{T}_{\text{MIN}}$ and $\mathcal{T}_{\text{MAJ}}$. The initialization of $\mathcal{T}_{\text{DANGER}}$ follows the next process:

(i) A prototype p is extracted from minority class set $\mathcal{T}_{\text{MIN}}$.

(ii) The kNN rule is applied to p in the entire train set $\mathcal{T}$.

(iii) If more than a half of the neighbors belong to the set of the majority class $\mathcal{T}_{\text{MAJ}}$, instance p is included in $\mathcal{T}_{\text{DANGER}}$.

(iv) The process is repeated until all prototypes in $\mathcal{T}_{\text{MIN}}$ have been queried.

Once $\mathcal{T}_{\text{DANGER}}$ has been obtained, B1 and B2 consider different ways of generating prototypes. On the one hand, B1 populates the minority class using the same process as SMOTE but iterating over the set $\mathcal{T}_{\text{DANGER}}$ instead of $\mathcal{T}_{\text{MIN}}$ (line 2 of Algorithm 2). B2, on the other hand, maintains the idea of B1 but additionally generates instances between pairs of prototypes from sets $\mathcal{T}_{\text{DANGER}}$ and $\mathcal{T}_{\text{MAJ}}$.

4. Evaluation methodology

Recalling from above, the proposed algorithm aims at creating new string prototypes to compensate the imbalance class distribution and thus improve the performance of the classifier. This sections describes the evaluation methodology considered for assessing the goodness of the approach, which is graphically shown in Fig. 1.

In order to have a higher control over the experiments, we considered a set of balanced data collections. By means of an external parameter, we controlled the imbalance degree by randomly removing prototypes from one of the classes (the one chosen as minority). Having the initial balanced dataset allowed us to obtain a *glass ceiling* in terms of the classification performance which might be achieved when artificially compensating the class distribution. We considered several minority class ratios with respect to the majority class size, namely 20 %, 40 %, 60 % and 80 %. Once the data was artificially imbalanced, we applied the proposed string-based SMOTE algorithm, as well as the B1 and B2 extensions. In our experiments

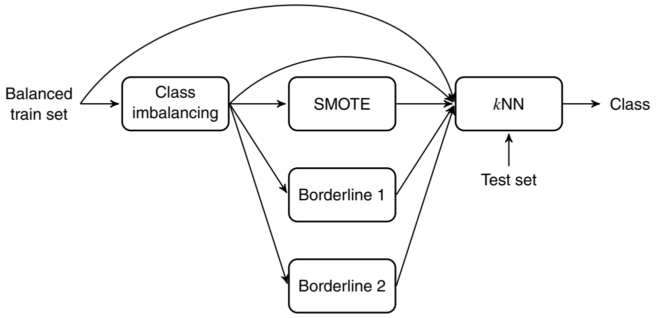


Figure 1. Experimental scheme considered. The initial data collection, which may be imbalanced, is equilibrated using one of the proposed string-based oversampling strategies. Finally, classification is performed using the *k*NN classifier with $k = 1$.

we considered a threshold value of $t = 0.5$ for the SMOTE and Borderline 1 algorithms so that the new string prototypes were generated at the same distance of the initial two string elements; for the Borderline 2 method we considered a threshold value of $t = 0.75$ to generate the new prototypes closer to the string element belonging to the minority class. For all cases we considered a neighborhood of $k = 1$ elements (Algorithm 2).

Note that, since the gist of the work is related to the use of string data, many classification algorithms such as Support Vector Machines, Decision Trees or Artificial Neural Networks cannot be used. Thus, we restricted ourselves to the use of the kNN with $k = 1$ for the classification stage, with the Edit Distance as dissimilarity measure.

Regarding the precise data collections, we considered three different datasets: the *United States Postal Service* (USPS) digit dataset (Hull, 1994) which comprises images of 16×16 pixels of single handwritten digits, the *NIST SPECIAL DATABASE* (NIST) of handwritten characters (Wilkinson et al., 1992), and the MNIST collection of isolated handwritten digits in images of 28x28 pixels (LeCun et al., 1998). All these datasets comprise images of isolated shapes, hence contour descriptions with Freeman Chain Codes (Freeman, 1961) were extracted. Note that in no case we are claiming that this representation is the most suitable for these datasets but it allowed us to perform the desired experimentation in the string space.

Furthermore, these datasets entail a multi-class scenario. However, SMOTE is designed for two-class classification scenarios. Thus, we chose pairs of classes from the aforementioned datasets which depicted a higher level of confusion. That is, those that imply a more challenging recognition problem. For each dataset, this confusion level was determined by creating subsets containing all the pairs of classes of the collection and classifying each subset using kNN with $k = 1$. Eventually, we selected the subsets which depicted the lowest classification scores: for the MNIST and USPS collections we selected the pairs of classes $(3, 5)$ and $(7, 9)$; for the NIST collection, the highest confusion level was found in the pairs (C, G) and (V, Y). Table 1 details the number of instances of the data collections in terms of the imbalance level considered.

For the quantitative evaluation of the strategies we considered the F-measure (F_1) as it constitutes a typical figure of merit in the context of imbalanced classification. Focusing on the minority class, this measure is obtained as

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \tag{1}$$

where TP stands for the True Positives or correctly classified elements, FP represents the False Positives or misclassified elements from the majority class as minority ones, and FN stands for the False Negatives or the misclassified elements from the minority class as of the majority one.

Finally, it should be noted that all methods in this paper have been implemented in Java language and the results shown in Section 5 have been obtained with a general purpose computer with the following technical specifications: Intel(R) Core(TM) i7-4700HQ CPU @2.40GHz, 12GB RAM, and Linux Mint (64 bits) operating system.

5. Results

This section presents the results obtained for the assessment strategy proposed. The results provided correspond to the average of the individual values obtained with a 5-fold cross-validation scheme for each data collection.

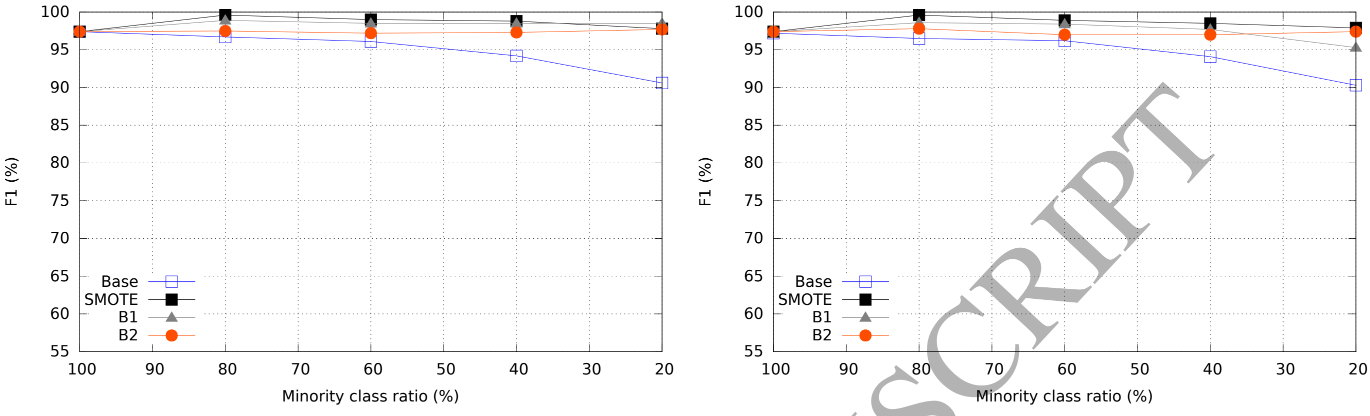
For a better comprehension, these results are provided graphically in Figure 2. For all cases, the x-axis depicts the percentage of minority class after the initial class-imbalance process, whereas the y-axis represents the classification performance in terms of the F_1 measure.

An initial point to comment is the performance of the baseline configuration considered. As the imbalance ratio of the distribution was progressively increased, the performance of the classifier remarkably decreased for all data collections assessed. For instance, for the MNIST set, the performance decreased from an $F_1 = 97.5$ % with the balanced data distribution to a value of $F_1 = 90.5$ % when the minority class was reduced to a 20 % of its initial size. This decrease was also observed for the USPS dataset, in which the performance lowered from the a value of $F_1 = 92.5$ % to an $F_1 = 75$ %, the NIST (C, G) case for which the classifier showed an initial correctness ratio of $F_1 = 88$ % which deteriorated to an $F_1 = 55$ %, and in the NIST (V, Y) case in which this decrease went from an $F_1 = 88$ % to an $F_1 = 68$ %. All these values confirmed the initial suspicion of that an imbalanced data distribution could imply a worse performance of the classifier.

Focusing now on the results retrieved by the balancing strategies, it can be observed that the proposed methods were able to cope with the aforementioned issues of imbalanced scenarios. For all studied imbalance ratios, the different string-based oversampling methods were not only capable of recovering the performance loss due to the imbalance, but also they improved over the baseline results. Some particular examples in which this improvement was remarkably noticeable are all the NIST cases, for which the classification performance improved from a value of $F_1 = 88$ % to correctness rates above $F_1 = 95$ %, and the USPS case, in which the classification rates improved from an $F_1 = 92$ % to performances above $F_1 = 95$ %. For the particular case of the MNIST set, the proposed string-based oversampling techniques were able to recover the performance

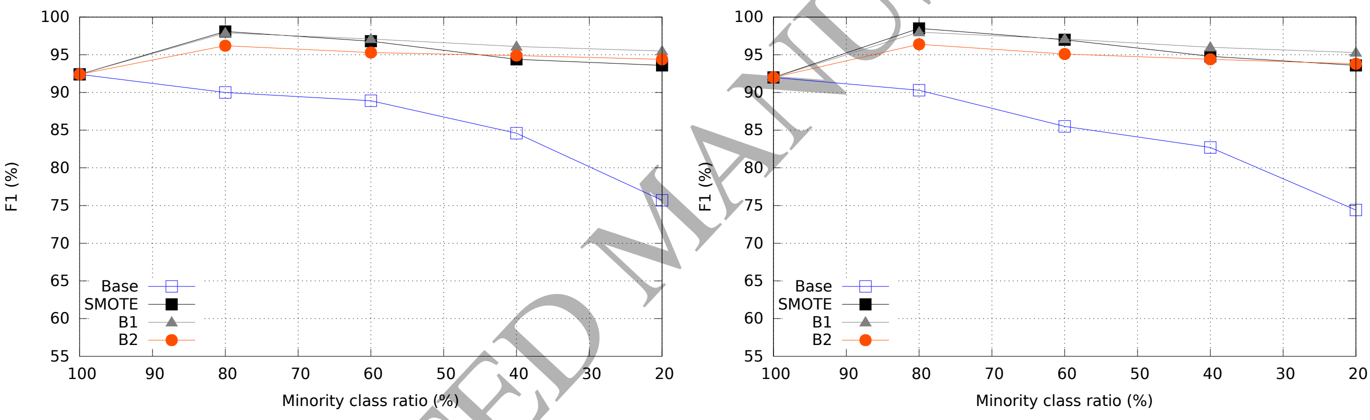
Table 1. Description of the datasets in terms of the number of instances.

Database	Majority class size	Minority class ratio				
		100 %	80 %	60 %	40 %	20 %
NIST	520	520	416	312	208	104
MNIST	1000	1000	800	600	400	200
USPS	928	928	742	556	372	186



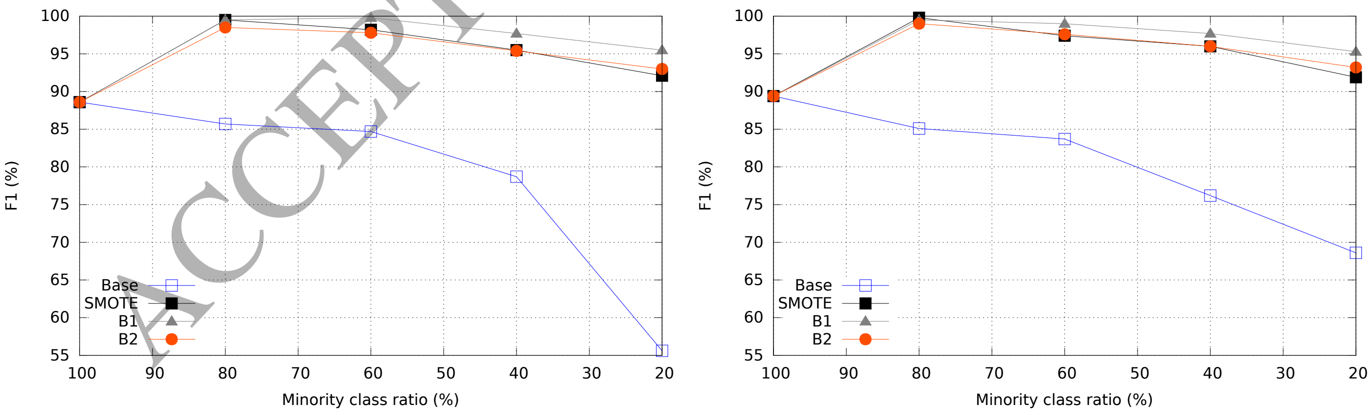
(a) MNIST (3 and 5)

(b) MNIST (7 and 9)



(c) USPS (3 and 5)

(d) USPS (7 and 9)



(e) NIST (`C` and `G`)

(f) NIST (`V` and `Y`)

Figure 2. Results obtained for the MNIST (2a, 2b), NIST (2e, 2f), and USPS (2c, 2d) datasets for the selected class pairs. The graphs assess the different string-based oversampling strategies proposed considering a kNN classifier (with $k = 1$) and different initial imbalance ratios.

decrease of the imbalance distribution, but did not show the performance boost over the baseline observed in the rest of the collections. This fact was possibly due to the high baseline performance of the MNIST collection obtained in our experiments ($F_1 = 97.5$ %), which constitutes a result considerably difficult to surpass.

Regarding the individual performance of each of the oversampling strategies, it can be observed that the results achieved by the three proposed balancing methods did not show remarkable differences among them. For instance, for all the data collections considered, the proposed strategies retrieved quite similar performances with subtle differences among them (the different curves practically overlap).

To facilitate the overall analysis, Table 2 details the average classification performance results obtained in terms of the F_1 measure for each imbalance ratio and balancing method proposed. These average results support the commented conclusion that the proposed strategies palliate the accuracy loss caused by the class imbalance. Specifically, the B1 strategy seems to report the best overall performance. As a representative case, it is observed that the use of B1 exhibits improvement from $F_1 = 75.2$ % to $F_1 = 94.0$ % when the considering highest initial imbalance, i.e. 20 % of the minority class.

Table 2. Average results in terms of F_1(%) obtained with the considered datasets for the different string-based oversampling strategies with respect to the minority class ratio. Bold figures highlight the best result for each initial class imbalance situation.

Minority class ratio	Base	SMOTE	B1	B2
80 %	90.0	**98.8**	98.7	97.1
60 %	88.1	97.1	**98.1**	96.2
40 %	83.5	95.9	**96.8**	95.5
20 %	75.2	94.0	**96.0**	94.7

Finally, in order to minimize the possibility that the differences in the performance observed were due to chance variation, we performed a pairwise, non-parametric Wilcoxon signed rank test (Demsar, 2006) to statistically assess their performance. We used the 30 independent results (one per corpus and cross-validation fold) to perform these statistical tests. The assumed hypothesis is that the use of our strategy significantly improves the accuracy obtained if classification is performed directly with the imbalanced dataset (ie., the baseline configuration considered). The results can be checked in Table 3.

Attending to the results in this statistical analysis it can be observed that, for all imbalance cases, the proposed string-based oversampling strategies statistically improved over the baseline situation. Furthermore, note that the statistical significance level of 99 % established in this evaluation constitutes a very restrictive threshold. Thus, the fact that the proposed modification on the SMOTE, B1, and B2 oversampling methods were able to surpass this acceptance level accounts for their goodness and usefulness in the context of imbalanced classification problems with string data.

Table 3. Results of the Wilcoxon test comparing the performance of the baseline classification (no oversampling) against our oversampling strategies with SMOTE, B1, or B2. Symbol ✓represents that the oversampling significantly improves the baseline. A significance level $p < 0.01$ has been considered for the analysis.

Minority class ratio	Oversampling strategy		
	SMOTE	B1	B2
80 %	✓	✓	✓
60 %	✓	✓	✓
40 %	✓	✓	✓
20 %	✓	✓	✓

6. Conclusions

In this paper we proposed a new approach to solve the problem of imbalanced data in the string space. The idea was to use the well-known Synthetic Minority Oversampling Technique (SMOTE) and its Bordeline 1 and 2 extensions. Since these techniques were designed for feature vectors—for which it is easy to generate new prototypes at specific points of the space— we used a strategy to be able to directly generate new string prototypes. This strategy is capable of generating a new prototype in the segment between two given strings, and so it allows the adaptation of SMOTE to this scenario, which could be applied with datasets generated with Freeman Code as HOMUS (Calvo-Zaragoza and Oncina, 2014).

To corroborate the goodness of our approach, we performed experiments with datasets of isolated symbols (digits and characters). These data can be represented as strings by encoding their contours with Freeman Chain Codes, being therefore suitable for our case. To control the experiments and make the appropriate comparisons, the datasets were artificially imbalanced at different levels (20 %, 40 %, 60 %, and 80 %). In all the experiments performed, our oversampling strategy on the string space significantly improved the classification results compared to directly considering the initial imbalanced data collection.

Given the successful results obtained, we intend to follow a similar approach for developing Prototype Generation (PG) algorithms in the string space. Unlike oversampling, which allows the dataset to be balanced, PG algorithms are designed to have a more compact representation of the initial collection. For that, they create prototypes located in key places of the space so that many of the original prototypes can be discarded with the ultimate goal of speeding up the k-nearest neighbor search (Triguero et al., 2012). As a more ambitious project, the idea is also to extend this approach to other structured spaces such as trees or graphs, since there also exist satisfactory (yet approximate in most cases) algorithms to compute an edit distance in these spaces.

Acknowledgments

This work was partially supported by the Spanish Ministerio de Economía y Competitividad through Project TIMuL (No. TIN2013-48152-C2-1-R supported by EU FEDER funds), the Universidad de Alicante through the FPU program (UAFPU2014–5883) and grant GRE-16-04.

References

Abdi, L., Hashemi, S., 2016. To combat multi-class imbalanced problems by means of over-sampling techniques. IEEE Transactions on Knowledge and Data Engineering 28, 238–251.

Abreu, J., Rico-Juan, J.R., 2013. An improved fast edit approach for two-string approximated mean computation applied to ocr. Pattern Recognition Letters 34, 496–504.

Almalawi, A.M., Fahad, A., Tari, Z., Cheema, M.A., Khalil, I., 2016. k-nnvwc: An efficient k-nearest neighbors approach based on various-widths clustering. IEEE Transactions on Knowledge and Data Engineering 28, 68–81.

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., Hawalah, A., Hussain, A., 2016. Comparing oversampling techniques to handle the class imbalance problem: a customer churn prediction case study. IEEE Access 4, 7940–7957.

Bach, M., Werner, A., Żywiec, J., Pluskiewicz, W., 2017. The study of under- and over-sampling methods utility in analysis of highly imbalanced data on osteoporosis. Information Sciences 384, 174–190.

Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

Bunke, H., Riesen, K., 2012. Towards the unification of structural and statistical pattern recognition. Pattern Recognition Letters 33, 811–825.

Calvo-Zaragoza, J., Oncina, J., 2014. Recognition of pen-based music notation: the homus dataset, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE. pp. 3038–3043.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research , 321–357.

Cherian, A., Sra, S., Morellas, V., Papanikolopoulos, N., 2014. Efficient nearest neighbors via robust sparse hashing. IEEE Transactions on Image Processing 23, 3646–3655.

Ciregan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE. pp. 3642–3649.

Cover, T.M., Hart, P.E., 1967. Nearest neighbor pattern classification. Information Theory, IEEE Transactions on 13, 21–27.

Das, B., Krishnan, N.C., Cook, D.J., 2015. RACOG and wracog: Two probabilistic oversampling techniques. IEEE Transactions on Knowledge and Data Engineering 27, 222–234.

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

Duda, R.O., Hart, P.E., Stork, D.G., 2001. Pattern Classification. John Wiley & Sons.

Fischer, A., Plamondon, R., 2017. Signature verification based on the kinematic theory of rapid human movements. IEEE Transactions on Human-Machine Systems 47, 169–180.

Freeman, H., 1961. On the encoding of arbitrary geometric configurations. IRE Transactions on Electronic Computers , 260–268.

Gottlieb, L.A., Kontorovich, A., Krauthgamer, R., 2014. Efficient classification for metric data. IEEE Transactions on Information Theory 60, 5750–5759.

Han, H., Wang, W.Y., Mao, B.H., 2005. Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing, Springer. pp. 878–887.

He, H., Bai, Y., Garcia, E.A., Li, S., 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: IEEE International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1322–1328.

He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering 21, 1263–1284.

Hull, J., 1994. A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence 16, 550–554.

Jo, T., Japkowicz, N., 2004. Class imbalances versus small disjuncts. ACM Sigkdd Explorations Newsletter 6, 40–49.

Junsomboon, N., Phienthrakul, T., 2017. Combining over-sampling and under-sampling techniques for imbalance dataset, in: Proceedings of the 9th International Conference on Machine Learning and Computing, ACM, New York, NY, USA. pp. 243–247.

Kubat, M., Matwin, S., 1997. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection, in: Proceedings of the Fourteenth International Conference on Machine Learning (ICML), pp. 179–186.

Laurikkala, J., 2001. Improving identification of difficult small classes by balancing class distribution, in: Proceedings of the Conference on Artificial Intelligence in Medicine in Europe, Springer. pp. 63–66.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

Lee, D., Kim, S.J., 2015. Modified chain-code-based object recognition. Electronics Letters 51, 1996–1997.

Marteau, P., Gibet, S., 2015. On recursive edit distance kernels with application to time series classification. IEEE Transactions on Neural Networks and Learning Systems 26, 1121–1133.

McVicar, M., Santos-Rodríguez, R., Ni, Y., De Bie, T., 2014. Automatic chord estimation from audio: A review of the state of the art. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 22, 556–575.

Plamondon, R., Srihari, S.N., 2000. Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on pattern analysis and machine intelligence 22, 63–84.

Platt, J.C., 1999. Advances in kernel methods, MIT Press, Cambridge, MA, USA. chapter Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

Tesfahun, A., Bhaskari, D.L., 2013. Intrusion detection using random forests classifier with smote and feature reduction, in: Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on, IEEE. pp. 127–132.

Tomek, I., 1976. Two Modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics SMC-6, 769–772.

Triguero, I., Derrac, J., Garcia, S., Herrera, F., 2012. A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42, 86–100.

Valero-Mas, J.J., Calvo-Zaragoza, J., Rico-Juan, J.R., Quereda, J.M.I., 2017. A study of prototype selection algorithms for nearest neighbour in class-imbalanced problems, in: Pattern Recognition and Image Analysis - 8th Iberian Conference, IbPRIA 2017, Faro, Portugal, June 20-23, 2017, Proceedings, pp. 335–343.

Wagner, R.A., Fischer, M.J., 1974. The string-to-string correction problem. Journal of the ACM (JACM) 21, 168–173.

Wilkinson, R.A., Geist, J., Janet, S., Grother, P.J., et al., 1992. The first census optical character recognition system conference. Technical Report. US Department of Commerce.

Yoon, J., Lim, H., Kim, D.W., 2016. Music genre classification using feature subset search. International Journal of Machine Learning and Computing 6, 134.

Yu, H., Ni, J., 2014. An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 11, 657–666.

Zhu, T., Lin, Y., Liu, Y., 2017. Synthetic minority oversampling technique for multiclass imbalance problems. Pattern Recognition 72, 327–340.