Applied Soft Computing 22 (2014) 511–517

Contents lists available at ScienceDirect

Applied Soft Computing

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

# Preprocessing noisy imbalanced datasets using SMOTE enhanced with fuzzy rough prototype selection

Nele Verbiest[a,*], Enislay Ramentol[b], Chris Cornelis[a,c], Francisco Herrera[c,d]

[a] Department of Applied Mathematics and Computer Science, Ghent University, Belgium
[b] Department of Computer Science, University of Camagüey, Cuba
[c] Department of Computer Science and AI, University of Granada, Spain
[d] Faculty of Computing and Information Technology, North Jeddah, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

*Article history:*
Received 12 March 2013
Received in revised form
17 December 2013
Accepted 21 May 2014
Available online 2 June 2014

*Keywords:*
Imbalanced classification
SMOTE
Prototype selection
Fuzzy rough set theory

ABSTRACT

The Synthetic Minority Over Sampling TEchnique (SMOTE) is a widely used technique to balance imbalanced data. In this paper we focus on improving SMOTE in the presence of class noise. Many improvements of SMOTE have been proposed, mostly cleaning or improving the data after applying SMOTE. Our approach differs from these approaches by the fact that it cleans the data before applying SMOTE, such that the quality of the generated instances is better. After applying SMOTE we also carry out data cleaning, such that instances (original or introduced by SMOTE) that badly fit in the new dataset are also removed. To this goal we propose two prototype selection techniques both based on fuzzy rough set theory. The first fuzzy rough prototype selection algorithm removes noisy instances from the imbalanced dataset, the second cleans the data generated by SMOTE. An experimental evaluation shows that our method improves existing preprocessing methods for imbalanced classification, especially in the presence of noise.


## 1. Introduction

Imbalanced classification is an important topic in data mining, as data in real-world applications such as fraud detection [1], oil spill detection [2], text classification [3] and medical applications [4] turn out to be imbalanced. In imbalanced problems, at least one class (the minority class) is under-represented. In general imbalanced problems, there can be more than one minority class and many other classes, but these problems can all be reduced to the basic case where there is one minority class and one other class, called the majority class. The Imbalance Ratio (IR) expresses how imbalanced a dataset is, and is defined as the size of the majority class over the size of the minority class. Datasets with IR 1 are perfectly balanced, while datasets with higher IR are more imbalanced.

Using the classification accuracy to measure the performance of data mining techniques for imbalanced classification is not appropriate. E.g., when a dataset has 10% minority examples and 90% majority examples, a classifier that classifies any instance to the majority class will obtain a classification accuracy of 90%, but is clearly not a good classifier for the problem at hand. Instead, one may use the Receiver Operating Characteristic (ROC) curve. It plots the ratio of correctly classified minority instances against the ratio of correctly classified majority instances. The Area Under the ROC Curve (AUC, [5]) then reflects the trade-off between correctly classified minority and majority instances.

Many techniques for imbalanced classification do not focus on the classifier itself, but carry out preprocessing on the data before classification. One of the most well-known techniques is the Synthetic Minority Oversampling TEchnique (SMOTE, [6]), that introduces artificial instances using interpolation. After applying SMOTE, the dataset is balanced and common classifiers can be used.

We focus on improving SMOTE in the presence of class noise. SMOTE cannot handle noise adequately and can even reinforce it, as the introduced instances might be the result of interpolation between noisy instances. As most of the existing improvements of SMOTE [7–11] clean the data after applying SMOTE, they cannot tackle the class noise problem adequately.

We carry out data cleaning before applying SMOTE to improve the quality of the instances generated by SMOTE and apply data cleaning after SMOTE, such that even if some low-quality instances were introduced by SMOTE, they are removed, together with original instances that show a bad cooperation with the artificial ones. The technique is based on two ideas. The first is that when SMOTE is applied using noisy data, it is logical that the data that it generates is of low quality. The second is that, even when SMOTE is applied to

* Corresponding author at: Krijgslaan 281 (S9), 9000 Gent, Belgium. Tel.: +32 92644770; fax: +32 92644995.
*E-mail address:* Nele.Verbiest@UGent.be (N. Verbiest).

http://dx.doi.org/10.1016/j.asoc.2014.05.023
1568-4946/

clean data, it can still generate low-quality data, as SMOTE generates artificial data using a simple interpolation strategy. Therefore, we propose to carry out data cleaning before and after applying SMOTE to eliminate the potentially noisy instances.

As we focus on improving the K Nearest Neighbor (KNN, [23]) classification, we use the term Prototype Selection (PS, [12]) for data cleaning in the remainder of this work. We develop two PS methods based on fuzzy rough set theory: one for imbalanced data, called Fuzzy Rough Imbalanced Prototype Selection (FRIPS) and one for normal (balanced) data, called Fuzzy Rough Balanced Prototype Selection (FRBPS). Both algorithms measure the quality of the instances using fuzzy rough set theory and remove instances for which the quality is lower than a certain threshold. To determine this threshold, FRIPS evaluates candidate thresholds by measuring the training AUC, while FRBPS measures the training accuracy. The final hybrid algorithm subsequently applies FRIPS, SMOTE and FRBPS.

We develop an experimental analysis considering 65 datasets with artificial class noise, in which we analyze the components of the hybrid proposal, analyze the positive synergy among components, and show that our method improves state-of-the-art approaches.

The remainder of this work is structured as follows: in Section 2, we recall the SMOTE procedure, its improvements and Ordered Weighted Average (OWA) fuzzy rough set theory. In Section 3, we introduce a measure based on OWA fuzzy rough set theory to measure the quality of instances, introduce the PS methods FRIPS and FRBPS, and show how we use them to improve SMOTE. In Section 4 we experimentally show that our method improves state-of-the-art techniques w.r.t. AUC. We conclude and suggest further research directions in Section 5.

## 2. Preliminaries

In this section we provide preliminaries that make the remainder of the paper self-contained. We first recall the SMOTE procedure and its state-of-the-art improvements. Next, we provide the reader with background on classification of noisy data. Finally, we summarize the most important notions of fuzzy rough set theory and its extension to OWA based fuzzy rough sets.

### 2.1. SMOTE and its improvements

The SMOTE procedure [6] is often used to balance data. First, the $k$ nearest neighbors of all minority instances are determined. Next, artificial minority instances are introduced on the lines between the minority instances and their $k$ nearest neighbors, until the dataset is balanced.

Although SMOTE improves the classification of imbalanced data substantially, it can still been improved. E.g., SMOTE with Tomek Links (SMOTE-TL, [7]) uses SMOTE to over-sample the data and then cleans the data to get better-defined class clusters, by removing adjacent couples of instances from different classes. SMOTE with Edited Nearest Neighbors (SMOTE-ENN, [7]) over-samples the data using SMOTE and then cleans the data with the ENN prototype selection technique. SMOTE with Rough Set theory (SMOTE-RS$B^*$, [8]) cleans the data after applying SMOTE by removing instances that do not belong to the rough lower approximation. In SMOTE with Fuzzy Rough Set Theory (SMOTE-FRST, [9]), this approach is improved by using fuzzy rough set theory and by iteratively applying SMOTE and data cleaning.

Other techniques directly try to improve the quality of the instances generated by SMOTE by adjusting the SMOTE algorithm. Borderline SMOTE (SMOTE-BL1 and SMOTE-BL2, [11]) strengthens the borderline and its nearby points of the minority class by over-sampling instances in the border of the minority class. The difference between the two algorithms is that SMOTE-BL2 generates artificial instances that are closer to the minority class. Safe-level SMOTE (SMOTE-SL, [10]) assigns each minority instance a *safe level* before generating synthetic instances. When generating artificial instances with SMOTE, these instances are positioned closer to the instance with the largest safe level, such that the artificial instances are generated only in safe regions. A more recent approach to improve SMOTE is presented in [22]. Their algorithm, called Majority Weighted Minority Oversampling Technique (MWMOTE), first identifies the most hard-to-learn minority instances, and then assigns weights to them based on the distance between them and the nearest majority instances. The synthetic instances are then interpolated based on the most informative minority instances in a way that the generated instances lie inside a minority class cluster.

### 2.2. Classification of noisy data

When developing classification methods, one should always keep in mind that the data at hand may be noisy. There are various types of noise, which we recall here briefly.

The main distinction that should be made is between attribute noise, where errors are introduced to attribute values, and class noise, where certain instances are mislabeled. In [21], three types of attribute noise are distinguished: erroneous attribute values, missing or *don't know* values and incomplete or *don't care* values. Following [21], class noise can either come from contradictory examples, where two instances have the same attributes but a different class label, and misclassifications, where the instances are incorrectly annotated.

In this work, we focus on class noise. Instances that are generated by the SMOTE algorithm might include contradictory examples, which then should be removed by our FRBPS algorithm. To ensure that our techniques can handle misclassifications, we add artificial class noise to the data in the experimental section.

### 2.3. Fuzzy rough set theory

In this section we review the main concepts of fuzzy rough set theory [13] and its extension to OWA based fuzzy rough set theory [14].

From now on, $(X, \mathcal{A} \cup \{d\})$ is a decision system, consisting of $n$ instances $X = \{x_1, \ldots, x_n\}$, $m$ attributes $\mathcal{A} = \{a_1, \ldots, a_m\}$ and the decision attribute $d \notin \mathcal{A}$. The value of each instance $x \in X$ for an attribute $a \in \mathcal{A}$ is denoted by $a(x)$. We assume that the attributes $\mathcal{A}$ are continuous and normalized, that is, for $x \in X$ and $a \in \mathcal{A}$, $a(x) \in [0, 1]$. The decision attribute $d$ is nominal and assigns a class $d(x)$ to each instance $x \in X$. In our context, $d(x)$ can take two values for each $x \in X$: $d(x) = 1$ if $x$ is in the minority class and $d(x) = 2$ if $x$ is in the majority class.

We associate a fuzzy indiscernibility relation $R : X \times X \to [0, 1]$ with the decision system as follows:

$$\forall x, y \in X : R(x, y) = \underbrace{\mathcal{T}(1 - (a(x) - a(y))^2)}_{a \in \mathcal{A}}, \tag{1}$$

where $\mathcal{T}$ is a triangular norm[1] (t-norm), which is an associative and commutative mapping $\mathcal{T} : [0, 1]^2 \to [0, 1]$, increasing in both arguments and such that $\forall a \in [0, 1] : \mathcal{T}(a, 1) = a$. In this paper we use the Lukasiewicz t-norm given by

$$\forall a, b \in [0, 1] : \mathcal{T}(a, b) = \max(0, a + b - 1)$$

[1] Note that, as t-norms are commutative and associative, they can unambiguously be extended from $[0, 1]^2 \to [0, 1]$ to $[0, 1]^m \to [0, 1]$ operators.

As all attributes are normalized, $R(x, y)$ returns a value between 0 and 1 that expresses to what extent the instances $x$ and $y$ are similar to each other w.r.t. the attribute set $\mathcal{A}$.

Fuzzy rough set theory can be used to approximate concepts (fuzzy sets) by means of the indiscernibility relation. Given a fuzzy set $S$, the lower approximation of $S$ is a fuzzy set $X \to [0, 1]$ given by:

$$\forall x \in X : (R \downarrow S)(x) = \min_{y \in X} \mathcal{I}(R(x, y), S(y)) \tag{2}$$

It expresses to what extent instances similar to $x$ belong to $S$. Here, $\mathcal{I}$ is a fuzzy implication, which is a mapping $\mathcal{I} : [0, 1]^2 \to [0, 1]$, decreasing in the first and increasing in the second argument, and for which $\mathcal{I}(0, 0) = 1, \mathcal{I}(1, 1) = 1, \mathcal{I}(0, 1) = 1$ and $\mathcal{I}(1, 0) = 0$. In this work we use the Lukasiewicz implication, given by

$$\forall a, b \in [0, 1] : \mathcal{I}(a, b) = \min(1, 1 - a + b)$$

The second part of the fuzzy rough approximation of the fuzzy set $S$ is given by the upper approximation as follows:

$$\forall x \in X : (R \uparrow S)(x) = \max_{y \in X} \mathcal{T}(R(x, y), S(y)), \tag{3}$$

where $\mathcal{T}$ is a t-norm, we use the Lukasiewicz t-norm in this work. This lower approximation expresses to what extent there exist instances that are similar to $x$ and that belong to $S$.

In [14], it was noted that this classical definition of fuzzy rough sets is highly susceptible to noise: the lower and upper approximation only depend on one instance, as the strict minimum and maximum operators are used. Therefore, it was proposed to soften these minimum and maximum operators by means of OWA operators [15]. Recall that, given a weight vector $W = \langle w_1, \ldots, w_p \rangle$ for which $\sum_{i=1}^{p} w_i = 1$ and $\forall i \in \{1, \ldots, p\}, w_i \in [0, 1]$, the OWA aggregation of $p$ values $s_1, \ldots, s_p$ is given by:

$$OWA_W(s_1, \ldots, s_p) = \sum_{i=1}^{p} w_i t_i, \tag{4}$$

where $t_i = s_j$ if $s_j$ is the $i$th largest value in $s_1, \ldots, s_p$. This operator resembles the weighted average, but the difference is that the weights are associated with positions rather than with values.

The OWA operator can be used to soften the minimum operator. Consider a weight vector $W_{\min} = \langle w^1_{\min}, \ldots, w^p_{\min} \rangle$ such that weights are increasing. Then small values will be associated with large weights, while high values will be associated with low weights. As a result, the $OWA_{W_{\min}}$ operator behaves like the minimum operator, but does take into account more than one value. In this work we will use the slowly increasing additive weights $\langle 1, 2, \ldots, p-1, p \rangle$, after normalization this results in the following weights:

$$\forall i \in \{1, \ldots, p\} : w^i_{\min} = \frac{i}{p(p+1)/2}. \tag{5}$$

Completely analogously, we can define the $OWA_{W_{\max}}$ operator that softens the maximum operator. Its normalized weights $W_{\max} = \langle w^1_{\max}, \ldots, w^p_{\max} \rangle$ are given as follows:

$$\forall i \in \{1, \ldots, p\} : w^i_{\max} = \frac{p - i + 1}{p(p+1)/2}. \tag{6}$$

In [15] it was suggested to replace the maximum and minimum operators in the lower and upper fuzzy approximation by OWA operators, leading to the following definitions of OWA fuzzy rough sets:

$$(R{\downarrow}^{OWA} S)(x) = OWA_{W_{\min}} \mathcal{I}(R(x, y), S(y)) \tag{7}$$

$$(R{\uparrow}^{OWA} S)(x) = OWA_{W_{\max}} \mathcal{T}(R(x, y), S(y)) \tag{8}$$

## 3. Improving SMOTE using fuzzy rough prototype selection

In this section we describe the procedures we use to improve the SMOTE algorithm. First, we discuss how we can use fuzzy rough set theory to measure the quality of instances. Next, we use this measure to improve SMOTE by means of two fuzzy rough prototype selection techniques.

### 3.1. Measuring the quality of instances using OWA fuzzy rough sets

Fuzzy rough set theory can be used to measure the quality of instances. We first define the following fuzzy set for each $x \in X$:

$$\forall y \in X : [x]_d(y) = \begin{cases} 1 & \text{if } d(x) = d(y) \\ 0 & \text{else,} \end{cases}$$

which can be interpreted as the set containing all instances with the same class as $x$.

The quality of an instance $x \in X$ can now be measured by $\gamma : X \to [0, 2]$, defined as follows:

$$\gamma(x) = (R{\downarrow}^{OWA}[x]_d)(x) + (R{\uparrow}^{OWA}[x]_d)(x). \tag{9}$$

That is, we consider the lower and upper approximation of the class of $x$ by means of $R$, and then measure how well $x$ belongs to this upper and lower approximation. The lower approximation part expresses to what extent instances similar to $x$ also belong to the same class as $x$. This means that if many instances of a different class are close to $x$, the lower approximation membership is low. If the instances from different classes are further away, the lower approximation membership is higher. The upper approximation of $[x]_d$ by means of $R$ expresses to what extent there are instances close to $x$ from the same class.

To understand the intuition behind the $\gamma$ measure, we consider the case without OWA generalization and rewrite it as follows:

$$\begin{aligned}
\gamma'(x) &= (R \downarrow [x]_d)(x) + (R \uparrow [x]_d)(x) \\
&= \min_{y \in X} \mathcal{I}(R(x, y), [x]_d(y)) + \max_{y \in X} \mathcal{T}(R(x, y), [x]_d(y)) \\
&= \min_{y \in X}(\min(1, 1 - R(x, y) + [x]_d(y))) + \max_{y \in X}(\max(0, R(x, y) + [x]_d(y) - 1)) \\
&= \min_{y \in \{z \in X | d(z) \neq d(y)\}} (1 - R(x, y)) + \max_{y \in \{z \in X | d(z) = d(y)\}} R(x, y) \\
&= 1 - \max_{y \in \{z \in X | d(z) \neq d(y)\}} R(x, y) + \max_{y \in \{z \in X | d(z) = d(y)\}} R(x, y).
\end{aligned}$$

Here, we clearly see that an instance is penalized if there is an instance close to it belonging to a different class, and that it is rewarded if there is an instance from the same class close to it. By using OWA fuzzy rough sets, we soften this idea: the lower approximation penalizes the instance $x$ for all instances from different classes close to it, and the worse instances (instances from different classes closest to $x$) are penalized more. On the other hand, the upper approximation rewards the instance $x$ for all instances from the same class close to it, and the best instances (instances from the same class closest to it) are rewarded most.

### 3.2. Improving SMOTE using fuzzy rough prototype selection

The measure defined in the previous subsection can be used to measure the quality of the instances. We want to remove the low-quality instances, while maintaining the high-quality instances. The main question is now what threshold to use to decide if instances should be removed or retained. We proceed as follows: we calculate $\gamma(x)$ for all instances $x$ and use each of these values as a threshold. Next, we consider the corresponding instance subsets and select the one with the highest training AUC or training accuracy, depending on the problem (imbalanced or balanced) at hand.

First consider the quality of all instances: $\gamma(x_1), \ldots, \gamma(x_n)$. After removing duplicates from these values, we obtain the set of thresholds $T_\gamma$:

$$T_\gamma = \{\tau_1, \ldots, \tau_k\},$$

where $1 \leq k \leq n$. We now associate a subset of instances with each threshold $\tau \in T_\gamma$ as follows:

$$S_\tau = \{x \in X | \gamma(x) \geq \tau\}.$$

Next, we want to measure the quality of these subsets. We consider two cases:

- In case of normal (balanced) problems, we can use the training classification error using a leave-one-out procedure, as described in Algorithm 1. In this algorithm, each instance in the training set $X$ is classified using the candidate subset of prototypes $S$ as a pool for possible nearest neighbors. If the instance $x \in X$ to be classified is included in $S$, we temporarily omit it from $S$. As we use the 1NN classifier for the final classification, we also use it to calculate the training accuracy, that is, an instance is classified to the class of its nearest neighbor. The value returned by Algorithm 1 is the training accuracy, which can be used to assess the predictive quality of the candidate subset $S$.
- In case of an imbalanced dataset, it is better to use the training AUC instead of the training accuracy. For the 1NN classifier, its formula is given by:

$$1 + \frac{\text{TP}}{\text{TP} + \text{FN}} - \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{10}$$

with TP the number of correctly classified minority instances, TN the number of correctly classified majority instances, FP the number of falsely classified majority instances and FN the number of falsely classified minority instances. In Algorithm 2 it is described how to calculate the training AUC, based on the confusion matrix $C = \begin{pmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{pmatrix}$ that is updated for each classified instance.

**Algorithm 1.** trainACC, procedure to measure the training accuracy of a subset of instances using a leave-one-out approach on balanced datasets.

```
input: Reduced decision system (S, A ∪ {d}) (S ⊆ X).
acc ← 0
for x ∈ X do
  if x ∈ S then
    Find the nearest neighbor nn of x in S \ {x}
  else
    Find the nearest neighbor nn of x in S
  end if
  if d(nn) = d(x) then
    acc ← acc + 1
  end if
end for
Output acc
```

**Algorithm 2.** trainAUC, procedure to measure the AUC of a subset of instances using a leave-one-out approach on imbalanced datasets.

```
input: Reduced decision system (S, A ∪ {d}) (S ⊆ X).
Initialize confusion matrix C = ( 0 0 ; 0 0 )
for x ∈ X do
  if x ∈ S then
    Find the nearest neighbor nn of x in S \ {x}
    C(d(x), d(nn)) ← C(d(x), d(nn)) + 1
  else
    Find the nearest neighbor nn of x in S
    C(d(x), d(nn)) ← C(d(x), d(nn)) + 1
  end if
end for
Output AUC based on C.
```

Finally, the subsets $S_{\tau i_1}, \ldots, S_{\tau i_s}$ with the highest training accuracy (or training AUC in the case of imbalanced datasets) are considered. Notice that more than one subset can have the maximum training accuracy or AUC. The subset that is finally returned is $S_{median(\tau_{i_1}, \ldots, \tau_{i_s})}$. We opt to take the median because it is a compromise between removing possibly useful instances and retaining too many instances. Moreover, experimental evaluation in [16] showed that using the median resulted in the best accuracies.

As already mentioned in Section 1, we will use the term FRIPS in case the training AUC is used, and FRBPS in case the training accuracy is used.[2] Summarizing, these algorithms work as follows:

1. Calculate $\gamma(x_1), \ldots, \gamma(x_n)$
2. Remove duplicates, this results in $T_\gamma = \{1, \ldots, k\}$, where $1 \leq k \leq n$
3. Calculate $S_{\tau_1}, \ldots, S_{\tau_k}$
4. Calculate quality$(S_{\tau_1}), \ldots,$ quality$(S_{\tau_k})$ (quality = trainACC for FRBPS and quality = trainAUC for FRIPS)
5. Select the subsets $S_{\tau i_1}, \ldots, S_{\tau i_s}$ with the highest quality
6. Return $S_{median(\tau_{i_1}, \ldots, \tau_{i_s})}$.

These two algorithms can now be used to improve the SMOTE algorithm, as illustrated in Fig. 1. First, the entire imbalanced dataset $X$ is cleaned using FRIPS, resulting in $S_{FRIPS} \subseteq X$. Next, SMOTE is applied to balance this cleaned dataset and returns $S_{SMOTE}$. Finally, FRBPS is applied to remove noise in the balanced dataset and returns $S_{FRBPS} \subseteq S_{SMOTE}$.

## 4. Experimental evaluation

In this section we evaluate the performance of the designed procedure. We use 65 datasets from the Keel dataset repository[3] that are described in Table 1. The datasets have relatively many instances such that prototype selection makes sense, but the number of attributes is limited, to keep the running time of the experimentation under control. The IR of the datasets varies over the datasets: the glass1 dataset is almost balanced, while the abalone19 dataset is highly imbalanced.

As FRIPS is designed to remove noise, we add artificial class noise to the data. We consider six noise levels, ranging from 0% class noise to 50% class noise in steps of 10%. For instance, when the noise level is 30%, the class labels of 30% randomly picked instances of the training data are swapped. Both the class labels of minority and majority instances can be swapped.

We follow a 5-fold cross validation procedure to evaluate the performance of the algorithms. We divide each dataset in 5 folds, of which each fold is once used as test data and the remaining folds

[2] Note that FRBPS is similar to the Fuzzy Rough Prototype Selection (FRPS) method introduced in [16]. The difference is that we improved the measure that expresses the quality of instances by using both the OWA lower and upper approximation, whereas in [16] only the fuzzy rough positive region was considered.

[3] www.keel.es.

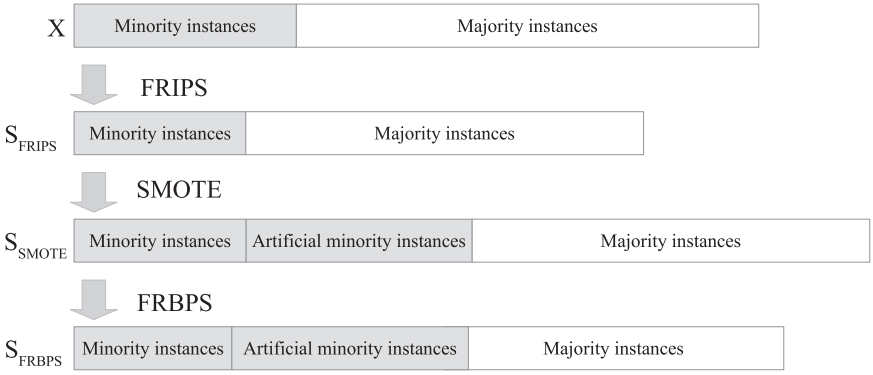


**Fig. 1.** Improving SMOTE with fuzzy rough prototype selection: the original data is cleaned using FRIPS, whereafter SMOTE is applied to balance the data. Finally, the balanced data is cleaned using FRBPS.

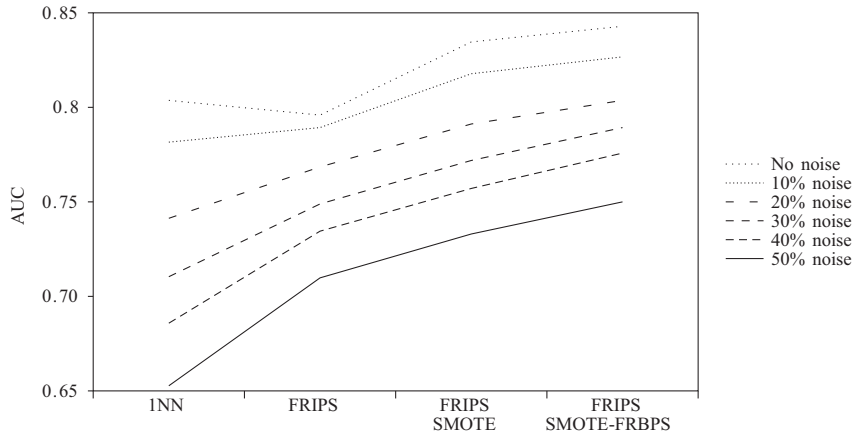


**Fig. 2.** Average AUC over all 65 datasets for different noise levels.

**Table 1**
Properties of the 60 datasets used in the experimental evaluation. The name, IR, number of attributes (# att) and number of instances (# inst) are given.

| Name | IR | # att | # inst | Name | IR | # att | # inst |
|---|---|---|---|---|---|---|---|
| glass1 | 1.82 | 9 | 214 | ecoli-0-3-4-6-vs-5 | 9.25 | 7 | 205 |
| ecoli-0-vs-1 | 1.86 | 7 | 220 | ecoli-0-3-4-7-vs-5-6 | 9.28 | 7 | 257 |
| pima | 1.87 | 8 | 768 | yeast-0-5-6-7-9-vs-4 | 9.35 | 8 | 528 |
| iris0 | 2 | 4 | 150 | vowel0 | 9.98 | 13 | 988 |
| glass0 | 2.06 | 9 | 214 | ecoli-0-6-7-vs-5 | 10 | 6 | 220 |
| yeast1 | 2.46 | 8 | 1484 | glass-0-1-6-vs-2 | 10.29 | 9 | 192 |
| haberman | 2.78 | 3 | 306 | ecoli-0-1-4-7-vs-2-3-5-6 | 10.59 | 7 | 336 |
| vehicle2 | 2.88 | 18 | 846 | led7digit-0-2-4-5-6-7-8-9-vs-1 | 10.97 | 7 | 443 |
| vehicle1 | 2.9 | 18 | 846 | glass-0-6-vs-5 | 11 | 9 | 108 |
| vehicle3 | 2.99 | 18 | 846 | ecoli-0-1-vs-5 | 11 | 6 | 240 |
| glass-0-1-2-3-vs-4-5-6 | 3.2 | 9 | 214 | glass-0-1-4-6-vs-2 | 11.06 | 9 | 205 |
| vehicle0 | 3.25 | 18 | 846 | glass2 | 11.59 | 9 | 214 |
| ecoli1 | 3.36 | 7 | 336 | ecoli-0-1-4-7-vs-5-6 | 12.28 | 6 | 332 |
| new-thyroid1 | 5.14 | 5 | 215 | cleveland-0-vs-4 | 12.62 | 13 | 177 |
| new-thyroid2 | 5.14 | 5 | 215 | ecoli-0-1-4-6-vs-5 | 13 | 6 | 280 |
| ecoli2 | 5.46 | 7 | 336 | shuttle-c0-vs-c4 | 13.87 | 9 | 1829 |
| segment0 | 6.02 | 19 | 2308 | yeast-1-vs-7 | 14.3 | 7 | 459 |
| glass6 | 6.38 | 9 | 214 | glass4 | 15.47 | 9 | 214 |
| yeast3 | 8.1 | 8 | 1484 | ecoli4 | 15.8 | 7 | 336 |
| ecoli3 | 8.6 | 7 | 336 | page-blocks-1-3-vs-4 | 15.86 | 10 | 472 |
| page-blocks0 | 8.79 | 10 | 5472 | abalone9-18 | 16.4 | 8 | 731 |
| ecoli-0-3-4-vs-5 | 9 | 7 | 200 | glass-0-1-6-vs-5 | 19.44 | 9 | 184 |
| yeast-2-vs-4 | 9.08 | 8 | 514 | shuttle-c2-vs-c4 | 20.5 | 9 | 129 |
| ecoli-0-6-7-vs-3-5 | 9.09 | 7 | 222 | yeast-1-4-5-8-vs-7 | 22.1 | 8 | 693 |
| ecoli-0-2-3-4-vs-5 | 9.1 | 7 | 202 | glass5 | 22.78 | 9 | 214 |
| glass-0-1-5-vs-2 | 9.12 | 9 | 172 | yeast-2-vs-8 | 23.1 | 8 | 482 |
| yeast-0-3-5-9-vs-7-8 | 9.12 | 8 | 506 | yeast4 | 28.1 | 8 | 1484 |
| yeast-0-2-5-7-9-vs-3-6-8 | 9.14 | 8 | 1004 | yeast-1-2-8-9-vs-7 | 30.57 | 8 | 947 |
| yeast-0-2-5-6-vs-3-7-8-9 | 9.14 | 8 | 1004 | yeast5 | 32.73 | 8 | 1484 |
| ecoli-0-4-6-vs-5 | 9.15 | 6 | 203 | ecoli-0-1-3-7-vs-2-6 | 39.14 | 7 | 281 |
| ecoli-0-1-vs-2-3-5 | 9.17 | 7 | 244 | yeast6 | 41.4 | 8 | 1484 |
| ecoli-0-2-6-7-vs-3-5 | 9.18 | 7 | 224 | abalone19 | 129.44 | 8 | 4174 |
| glass-0-4-vs-5 | 9.22 | 9 | 92 | | | | |

**Table 2**
Average AUC over all datasets for different noise levels.

| Datasets | 1NN | SMOTE | SMOTE TL | SMOTE ENN | SMOTE BL1 | SMOTE BL2 | SMOTE SL | SPIDER | SPIDER-2 | MWMOTE | SMOTE FRST | SMOTE RSB* | FRIPS SMOTE FRBPS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No noise | 0.8037 | 0.8315 | 0.8438 | **0.8483** | 0.8222 | 0.8207 | 8038 | 0.8198 | 0.8135 | 0.8194 | 0.8235 | 0.8394 | 0.8429 |
| 10% noise | 0.7816 | 0.7894 | 0.7911 | 0.7984 | 0.7961 | 0.7999 | 0.7819 | 0.8017 | 0.7989 | 0.7316 | 0.7974 | 0.7888 | **0.8267** |
| 20% noise | 0.7413 | 0.7418 | 0.7434 | 0.7588 | 0.7528 | 0.7566 | 0.7413 | 0.7723 | 0.7670 | 0.6677 | 0.7621 | 0.7357 | **0.8037** |
| 30% noise | 0.7104 | 0.7048 | 0.7105 | 0.7350 | 0.7144 | 0.7196 | 0.7104 | 0.7476 | 0.7368 | 0.6108 | 0.7334 | 0.7047 | **0.7893** |
| 40% noise | 0.6858 | 0.6808 | 0.6808 | 0.7165 | 0.6885 | 0.6934 | 0.6858 | 0.7222 | 0.7106 | 0.5610 | 0.7085 | 0.6798 | **0.7758** |
| 50% noise | 0.6527 | 0.6508 | 0.6539 | 0.6924 | 0.6514 | 0.6564 | 0.6526 | 0.6852 | 0.6707 | 0.4951 | 0.6772 | 0.6409 | **0.7500** |

The values indicated in bold are the best results obtained per noise level.

**Table 3**
Observed values of the non-parametric Wilcoxon test for the different noise levels, comparing FRIPS-SMOTE-FRBPS to state of the art algorithms.

| FRPIS SMOTE FRBPS vs. | | 1NN | SMOTE | SMOTE TL | SMOTE ENN | SMOTE BL1 | SMOTE BL2 |
|---|---|---|---|---|---|---|---|
| No noise | *R*+ | 1549.5 | 1226 | 834.5 | 780.5 | 1339.0 | 1337.5 |
| | *R*− | 161.5 | 544 | 876.5 | 930.5 | 372 | 373.5 |
| | *p*-Value | 0 | 0.0097 | 1 | 1 | 0.0002 | 0.0002 |
| 10%cn | *R*+ | 1615 | 1573.5 | 1518 | 1397 | 1567 | 1464 |
| | *R*− | 155 | 196.5 | 252 | 373 | 203 | 247 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | 0.0001 | ≤0.00001 | ≤0.00001 |
| 20%cn | *R*+ | 1637 | 1634 | 1581 | 1521 | 1536.5 | 1532 |
| | *R*− | 133 | 136 | 189 | 249 | 233.5 | 238 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 30%cn | *R*+ | 1590 | 1594 | 1641 | 1573.5 | 1589.5 | 1593 |
| | *R*− | 180 | 176 | 129 | 196.5 | 180.5 | 177 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 40%cn | *R*+ | 1608 | 1620 | 1591 | 1526 | 1607 | 1597.5 |
| | *R*− | 162 | 150 | 120 | 244 | 163 | 172.5 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 50%cn | *R*+ | 1675 | 1662 | 1709.5 | 1497 | 1671 | 1662 |
| | *R*− | 95 | 108 | 60.5 | 273 | 99 | 108 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |

| FRPIS SMOTE | | SMOTE SL | SPIDER | SPIDER-2 | MWMOTE | SMOTE FRST | SMOTE RSB* |
|---|---|---|---|---|---|---|---|
| No noise | *R*+ | 1551 | 1419.5 | 1533.5 | 1813.5 | 1402 | 966 |
| | *R*− | 160 | 291.5 | 236.5 | 331.5 | 368 | 804 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | 0.0001 | 0.5369 |
| 10%cn | *R*+ | 1616.5 | 1491.5 | 1469 | 2071 | 1521.5 | 1559 |
| | *R*− | 153.5 | 278.5 | 301 | 74 | 248.5 | 211 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 20%cn | *R*+ | 1637 | 1391 | 1460.5 | 2071 | 1515 | 1605 |
| | *R*− | 133 | 320 | 309.5 | 74 | 255 | 165 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 30%cn | *R*+ | 1593 | 1457 | 1512 | 2094 | 1459.5 | 1592 |
| | *R*− | 177 | 313 | 258 | 51 | 310.5 | 178 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 40%cn | *R*+ | 1609 | 1456 | 1483 | 2122 | 1511 | 1647 |
| | *R*− | 161 | 314 | 287 | 23 | 259 | 123 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |
| 50%cn | *R*+ | 1675 | 1552 | 1601 | 2129 | 1598 | 1692 |
| | *R*− | 95 | 218 | 169 | 16 | 172 | 78 |
| | *p*-Value | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 | ≤0.00001 |

as train data, and report the average AUC over all folds. For the final classification, we always use the 1NN classifier. The parameter $k$ in the SMOTE algorithm is fixed to 5 as in the original proposal.

The full results are available on our website.[4] First, we analyze the influence of FRIPS and FRBPS on the performance of 1NN and SMOTE. In Fig. 2, we show the average AUC of 4 methods for the six different noise levels. On the left hand side, we show the performance of the 1NN classifier without any preprocessing. It can be seen that the noise highly influences the AUC: there is a drop of 0.15 AUC if 50% class noise is added. The second method applies FRIPS before classifying the data. In case noise was added to the data, this improves the AUC. The more class noise is added, the better the improvement by FRIPS. The third method balances the data selected by FRIPS, which improves the classification in all cases. The last method applies FRBPS after balancing, which improves the AUC

[4] http://users.ugent.be/ nverbies/.

even more. Fig. 2 shows that all steps we proposed (FRIPS, SMOTE and FRBPS) do improve the results and are necessary.

Next, we compare the hybrid algorithm called FRIPS-SMOTE-FRBPS with state-of-the-art preprocessing techniques for imbalanced classification. Besides the improvements of SMOTE described in Section 2.1, we also consider the SPIDER algorithm which removes majority instances that result in misclassifying instances from the minority class and over-samples minority instances that are surrounded by majority instances [17]. The last algorithm we use is SPIDER2: a two-phase version of the SPIDER algorithm presented in [18]. In the first phase noisy majority instances are removed or relabeled, in the second phase noisy minority examples are amplified.

In Table 2, we show the average AUCs over all datasets for all methods. If no noise is added, the performance of FRIPS-SMOTE-FRBPS is equal to that of the best-performing methods SMOTE-TL and SMOTE-ENN. It can be derived from the full results that this result is mainly due to the worse performance of FRIPS-SMOTE-FRBPS for highly imbalanced problems. For instance, when we only consider datasets with $IR \leq 10$, FRIPS-SMOTE-FRBPS is the best performing algorithm (AUC is 0.8610 whereas the AUC of SMOTE-ENN and SMOTE-TL are 0.8583 and 0.8560, respectively).

When class noise is added, FRIPS-SMOTE-FRBPS performs better than all the other algorithms. The more noise added, the larger the differences. To see if FRIPS-SMOTE-FRBPS significantly outperforms the state-of-the-art results, we perform the statistical Wilcoxon test [19,24–26]. This is a non-parametric pairwise test that aims to detect significant differences between two sample means; that is, the behavior of the two implicated algorithms in the comparison. For each comparison we compute $R+$, the sum of ranks of the Wilcoxon test in favor of FRIPS-SMOTE-FRBPS, $R$, the sum of ranks in favor of the other methods, and also the $p$-value obtained for the comparison. The observed values of the statistics are listed in Table 3, clearly we found very low $p$-values associated to the comparative analysis for high noise levels.

## 5. Conclusion and further work

In this paper we proposed a technique to improve SMOTE using fuzzy rough set theory, the hybrid algorithm FRIPS-SMOTE-FRBPS. First, we clean data such that SMOTE generates better artificial data using FRIPS, a fuzzy rough prototype selection method that takes into account the imbalanced character of the data. Next, we apply SMOTE to balance the data, and clean the data again, now using FRBPS, a fuzzy rough prototype selection method for balanced data.

An experimental evaluation on real datasets with artificial class noise shows that our method improves state-of-the-art approaches.

In the future we would like to make a more general study on how non-fuzzy rough prototype selection techniques could improve SMOTE. That is, in this work we only considered fuzzy rough approaches to prototype selection, but it might be interesting to replace these by, e.g., evolutionary approaches as it was proposed in [20].

## Acknowledgments

This work was partially supported by the Spanish Ministry of Science and Technology under project TIN2011-28488 and the Andalusian Research Plans P11-TIC-7765, P10-TIC-6858.

## References

[1] T. Fawcett, F. Provost, Adaptive fraud detection, Data Min. Knowl. Discov. 1 (3) (1997) 291–316.
[2] M. Kubat, R.C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Mach. Learn. 30 (2–3) (1998) 195–215.
[3] Y. Liu, H.T. Loh, A. Sun, Imbalanced text classification: a term weighting approach, Expert Syst. Appl. 36 (1) (2009) 690–701.
[4] L. Mena, J.A. Gonzalez, Machine learning for imbalanced datasets: application in medical diagnostic, in: Proceedings of the FLAIRS Conference, 2006, pp. 574–579.
[5] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (1997) 1145–1159.
[6] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, SMOTE: synthetic minority oversampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[7] G. Batista, R. Prati, M. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explor. 6 (1) (2004) 20–29.
[8] E. Ramentol, Y. Caballero, R. Bello, F. Herrera, SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory, Knowl. Inf. Syst. 33 (2) (2011) 245–265.
[9] E. Ramentol, N. Verbiest, R. Bello, Y. Caballero, C. Cornelis, F. Herrera, SMOTE-FRST: a new resampling method using fuzzy rough set theory, in: Proceedings of the 10th International FLINS Conference on Uncertainty Modeling in Knowledge Engineering and Decision Making, 2012, pp. 800–805.
[10] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-SMOTE: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2009, pp. 475–482.
[11] H. Han, W. Wang, B. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Proceedings of the International Conference on Intelligent Computing, 2005, pp. 878–887.
[12] S. García, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.
[13] D. Dubois, H. Prade, Rough fuzzy sets and fuzzy rough sets, Int. J. Gen. Syst. 17 (1990) 191–209.
[14] C. Cornelis, N. Verbiest, R. Jensen, Ordered weighted average based fuzzy rough sets, in: Proceedings of the 5th International Conference on Rough Set and Knowledge Technology, 2010, pp. 78–85.
[15] R.R. Yager, On ordered weighted averaging aggregation operators in multicriteria decisionmaking, IEEE Trans. Syst. Man Cybern. 18 (1988) 183–190.
[16] N. Verbiest, C. Cornelis, F. Herrera, FRPS: a fuzzy rough prototype selection method, Pattern Recognit. 46 (10) (2013) 2770–2782.
[17] J. Stefanowski, S. Wilk, Selective pre-processing of imbalanced data for improving classification performance, in: Proceedings of the 10th International Conference in Data Warehousing and Knowledge Discovery, 2008, pp. 283–292.
[18] K. Napierala, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing, 2010, pp. 158–167.
[19] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 6 (1945) 80–83.
[20] S. García, F. Herrera, Evolutionary under-sampling for classification with imbalanced data sets: proposals and taxonomy, Evol. Comput. 17 (3) (2009) 275–306.
[21] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study of their impacts, Artif. Intell. Rev. 22 (3–4) (2004) 177–210.
[22] 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. 99 (2012) (preprints).
[23] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[24] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining, Exp. Anal. Power Inf. Sci. 180 (2010) 2044–2064.
[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] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.