G-SOMO
An Oversampling Approach based on Self-Organized Maps and Geometric SMOTE

Georgios Douzas*[1,a], Rene Rauch[1,b], Fernando Bacao[1,c]

Affiliation: [1]NOVA Information Management School, Universidade Nova de Lisboa

Postal Address: NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal

Telephone: +351 21 382 8610

Email: [a]gdouzas@novaims.unl.pt, [b]rene.rauch@outlook.de, [c]bacao@novaims.unl.pt

*Corresponding Author

Traditional supervised machine learning classifiers are challenged to learn highly skewed data distributions as they are designed to expect classes to equally contribute to the minimization of the classifiers cost function. Moreover, the classifiers design expects equal misclassification costs, causing a bias for overrepresented classes. Different strategies have been proposed to correct this issue. The modification of the data set has become a common practice since the procedure is generalizable to all classifiers. Various algorithms to rebalance the data distribution through the creation of synthetic instances were proposed in the past. In this paper, we propose a new oversampling algorithm named G-SOMO. The algorithm identifies optimal areas to create artificial data instances in an informed manner and utilizes a geometric region during the data generation process to increase their variability. Our empirical results on 69 datasets, validated with different classifiers and metrics against a benchmark of commonly used oversampling methods show that G-SOMO consistently outperforms competing oversampling methods. Additionally, the statistical significance of our results is established.

Keywords: Machine Learning, Classification, Imbalanced Learning, Oversampling, G-SMOTE, SOM.

1 Introduction

Learning from imbalanced datasets is a pervasive and challenging problem in supervised machine learning. Imbalanced datasets are common in many application domains such as fraud detection, medical diagnosis, risk management, airborne imagery, face recognition and forecasting of ozone levels and can be seen as a form of data scarcity [Vong et al., 2014], [He and Garcia, 2009]. The recent growth in the number of research papers on the topic [Haixiang et al., 2017], [Fernandez et al., 2018], suggests that it continues to be a relevant and important topic for the research community.

The imbalanced learning problem can be defined as a classification problem where there is a substantial asymmetry between the number of instances of the different classes. The dominant class is called the majority class while the other classes are called the minority classes [Chawla et al., 2003]. The Imbalance Ratio (IR) is the measure commonly used to assess the imbalance severity in datasets. The IR of each

minority class is defined as the ratio between the majority and the minority class number of samples. It is a frequent case to observe IR values between 100 and 100.000 [Chawla et al., 2002], [Barua et al., 2014].

Imbalanced datasets constitute a problem since standard machine learning methods induce a bias in favor of the majority class during training. This happens because the minority classes contribute less to the minimization of accuracy, the commonly used objective function. It is well known that accuracy is heavily biased towards the majority class, and additionally can be a misleading choice when assessing the performance of a classifier. For instance, assuming a dataset with 99% majority class instances and only 1% minority class instances, the model accuracy would still be 99% without classifying a single minority instance correctly.

It is also important to note that most methods assume equal costs in the misclassification of minority and majority instances. However it is often the case, that the misclassification costs associated with the misclassification of minority instances (false negatives) is much higher than the misclassification of majority instances (false positives) [Domingos, 1999], [Ting, 2002]. Diseases screening tests are just an example of the situation in which false negatives involve a much higher cost than the false positives [Wan et al., 2014].

In general, there are three different options to handle imbalanced datasets [Fernández et al., 2013]. One is to adjust the cost function of the algorithm, which means that during the training process, the model will be heavily penalized for misclassifying minority instances (false negatives). This can be a laborious task as it involves the modification of the training process for every algorithm the user applies. Another option is to focus on the modification of algorithms, reinforcing the learning towards the minority class. Again, this can be a complex and lengthy procedure, possibly restricting the algorithm options available to the user. Finally, the last option is to modify the data. This can be done by generating new synthetic instances of the minority classes, known as oversampling, or by removing instances from the majority class, known as undersampling. The objective of both is to balance the dataset, minimizing the skewness of the data distribution.

In this paper, we adopt the last option and tackle imbalanced learning problem by rebalancing class distribution through oversampling. This is a more general approach because, once the distribution is balanced, any supervised machine learning algorithm can be used, without any further modification. Additionally, by using oversampling, contrary to undersampling, no data is wasted, as none of the majority instances needs to be removed.

The oversampling process can be divided into two steps: An initial process of choosing where to generate the synthetic minority instances and the actual process of generating them. Both of these steps are essential for an effective oversampling algorithm and should prevent the generation of duplicate as well as noisy instances.

The method presented in this paper, G-SOMO, extends the Self-Organizing Map Oversampling algorithm (SOMO) [Douzas and Bacao, 2017], shown to be particularly efficient in identifying areas where the synthetic minority instances should be generated, by coupling it with the Geometric SMOTE algorithm (G-SMOTE) [Douzas and Bacao, 2019], used as the data generation mechanism of the minority instances.

SOMO is a clustering based oversampling algorithm which uses the Self-Organizing Map (SOM) and has been shown to outperform related algorithms over various datasets. It preserves the topology of the input space, while reducing the dimensionality to a two-dimensional representation. The emerging clusters are then used as a map to guide the generation of minority instances within, but also between neighboring minority clusters. In the original SOMO paper the generation process was done through the application of the SMOTE mechanism [Chawla et al., 2002].

The new algorithm presented in this paper, replaces SMOTE with G-SMOTE as the data generation mechanism. G-SMOTE generates synthetic samples in a geometric region of the input space, around each selected minority instance. It has been shown that G-SMOTE produces significant improvements in the generated data quality. We evaluate G-SOMO performance on binary classification problems using 69 datasets, 4 different classifiers and 3 different performance measures and compare it with Random Oversampling (ROS), SMOTE, K-Means SMOTE [Douzas et al., 2018], SOMO and G-SMOTE.

In the next section we will outline related methods that have been shown to be efficient. In section 3, we explain some of the shortcomings of oversampling methods and explain the motivation for using G-SOMO. Next, we present in detail the G-SOMO algorithm, followed by a description of the experimental setup. In section 6 the results and their statistical analysis is presented. The last section summarizes our findings and provides some ideas for further research.

2 Related work

The section outlines the current state of the art oversampling methods. Oversampling algorithms generate synthetic examples for the minority class that are added to the training set. In contrast to oversampling, undersampling methods remove samples from the majority class to establish a class balance. This implies that information is excluded, which might affect the learning process negatively, especially when the data set is small. Both methods have shown to be effective, depending on the problem addressed [Chawla et al., 2002]. More information on undersampling methods can be found in [Ganganwar, 2012] and [Yen and Lee,]. Synthetic data instances can be created uninformed, by randomly duplicating minority instances or informed, by identifying areas where oversampling is deemed to be most effective [Douzas et al., 2018].

The simplest form of oversampling is ROS, an uninformed approach, which randomly selects minority samples and duplicates them. The method stands out through its simplicity, but it increases the risk of overfitting, since the same information is used multiple times during the training process.

The most popular approach among practitioners in the domain of oversampling is SMOTE [Chawla et al., 2002]. The algorithm chooses a random minority instance and identifies its k nearest neighboring minority instances. The parameter k is chosen beforehand. A synthetic sample is created along the line segment joining the selected minority instance and a randomly selected minority class neighbor. Figure 1 shows the SMOTE data generation mechanism.

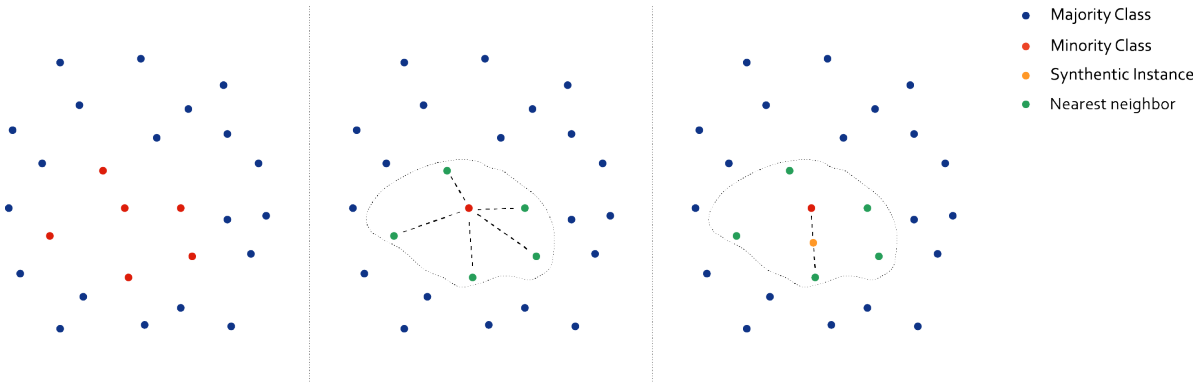

Figure 1: SMOTE data generation machanism, adopted from [Schubach et al., 2017]

.

Compared to ROS, the synthetic samples are not copies of the original samples and therefore the risk

of overfitting is reduced. However, SMOTE has several drawbacks. First, the algorithm randomly selects a minority instance for oversampling with uniform probability. Therefore it tackles between-class imbalance, while within-class imbalance is ignored [Nekooeimehr and Lai-Yuen, 2016]. This is because areas that were highly populated by minority samples will expand, while smaller areas of minority samples will remain sparse [Prati et al., 2004b]. Second, the algorithm is prone to the generation of noisy samples. An example of it is when initially a noisy minority class sample is selected. This scenario frequently results in the generation of additional noisy samples.

To tackle the problems of SMOTE several modifications were created. SMOTE + Edited Nearest Neighbor [Batista et al., 2004] removes any misclassified instances after the creation of synthetic samples, by applying the edited nearest neighbor rule. Safe-Level SMOTE [Bunkhumpornpat et al., 2009] applies a weight degree to differentiate between noisy and safe instances. Furthermore, G-SMOTE [Douzas and Bacao, 2019] extends the linear interpolation mechanism of SMOTE by introducing a geometric region where the data generation process occurs. Thereby, the algorithm increases the data variability and prevents additional correlation between the created samples. Further algorithms like Borderline-SMOTE [Han et al., 2005], MWMOTE (Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning) [Barua et al., 2014], ADASYN and its variation KernelADASYN [Tang and He, 2015] are trying to avoid noisy samples and focus on hard to learn instances. Hereby, the borderline samples of the classes are identified to use the informative minority class instances.

The prior algorithms focus exclusively on between-class imbalance. On the other hand, within-class-imbalance refers to identifying sparse or dense clusters of minority or majority instances. To tackle this challenge, different clustering based oversampling methods have been proposed. These methods are segmenting the instances and then apply traditional oversampling algorithms to each segment.

Cluster SMOTE [Cieslak et al., 2006] utilizes the K-means clustering algorithm, to identify clusters with a specific threshold of minority instances, before applying SMOTE within these clusters. In addition, K-means SMOTE [Douzas et al., 2018] method applies a similar approach, but also considers the density within the clusters. Then sparse clusters receive a higher number of generated samples compared to dense clusters. DBSMOTE [Bunkhumpornpat et al., 2012] provides another approach to cluster based oversampling, by applying the density-based DBSCAN algorithm to discover arbitrarily shaped clusters and create artificial data instances along the shortest path from each minority class instance to a pseudo-centroid of the cluster. A-SUWO [Nekooeimehr and Lai-Yuen, 2016] uses a hierarchical clustering approach and adaptively determines the size to oversample each sub-cluster using its classification complexity and cross-validation procedure.

The SOMO algorithm [Douzas and Bacao, 2017] applies a self-organizing map to create a two-dimensional representation of the multidimensional input space and creates inter-cluster and intra-cluster synthetic samples based on the underlying manifold structure. Specifically, the algorithm maps the input space in a two-dimensional space where clusters are identified. The next step is to apply SMOTE in order to generate synthetic instances within selected clusters as well as between them. Besides clustering based algorithms, further approaches have been proposed, that are based on ensemble methods like SMOTEBoost [Chawla et al., 2003] and DataBoost-IM [Guo and Viktor, 2004].

3 Motivation

The section briefly outlines the challenges and shortcomings of competitive oversampling methods and clarifies the motivation for our proposed algorithm G-SOMO.

A classifiers performance is significantly influenced by the positioning of the instances in the dataset as presented in Figure 2. A common challenge in real-world problems, is that the instances of a class can

be spread over the boundaries of the majority class [Tang and Gao, 2007]. Furthermore, the different types of instances can be labeled as safe and unsafe, while the latter are further divided as borderline and outlier instances [Sáez et al., 2016].

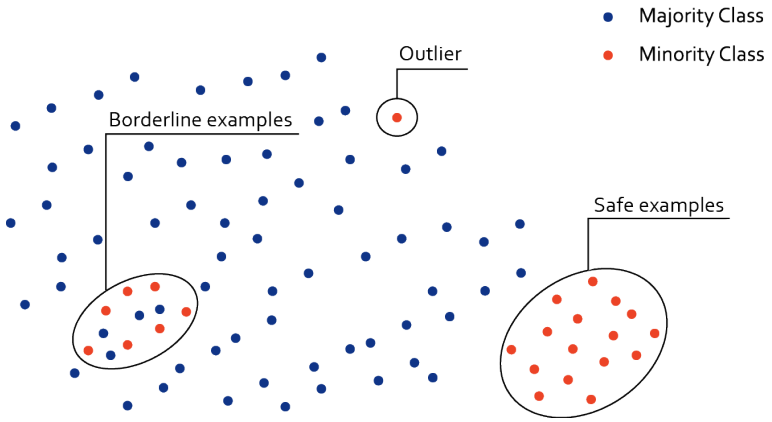

Figure 2: Challenging data formations, adopted from [Sáez et al., 2016].

Safe instances are located in the homogeneous regions and populated by the examples from one class only [Rodriguez et al., 2012]. They are clearly separated from examples of other classes. Most classifiers are able to correctly identify those instances [Prati et al., 2004a]. Instances that are not clearly separated are considered unsafe ones, like borderline instances and outliers [Kubat and Matwin, 1997]. Borderline instances are placed in the boundary region between classes, where instances of multiple classes overlap [Sáez et al., 2016]. Outliers are isolated distinct instances, also termed as noise.

Learning algorithms are challenged when they are confronted with overlapping areas between classes, especially unsafe instances. Therefore, oversampling algorithms should be able to carefully consider in which areas to create artificial instances and which areas to ignore. Inefficiencies that were observed with SMOTE and similar algorithms are the generation of noise penetrating the area of majority class, as well as the generation of nearly duplicated examples. Noisy examples can be created when applying k-nearest neighbor approaches, since they can either choose a noisy instance as a starting point or select a noisy instance as nearest neighbor, as seen in Figure 3a and 3b. Similar or nearly duplicated instances are created while generating new artificial instances within the borders of a minority cluster since they do not help to strengthen the decision boundary and might lead to overfitting, as it can be seen in Figure 3c. Figure 3d outlines the case, in which the parameter k is too high and observations from another cluster are selected.

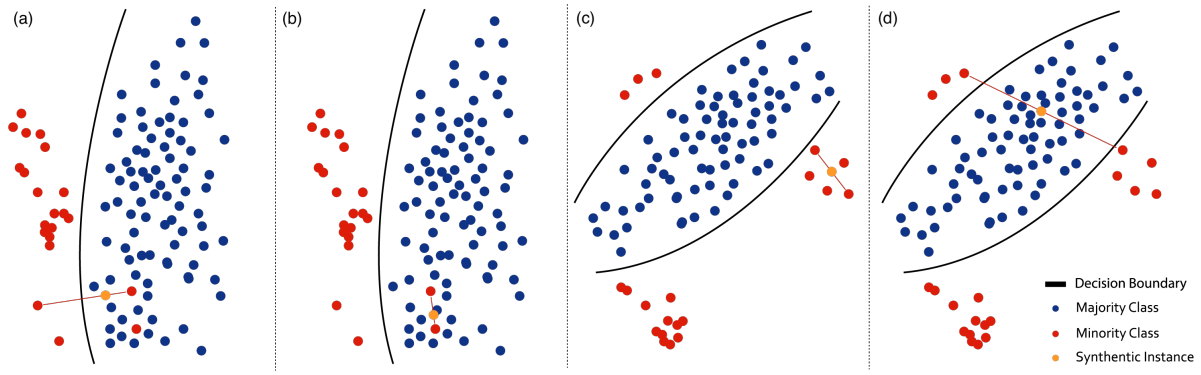

Figure 3: Shortcomings of SMOTE, adopted from [Douzas and Bacao, 2017]

SOMO algorithm improves the selection criteria of the minority class samples, which are used to generate

synthetic examples. Through the synthetic data generation in and between neighboring minority clusters, as well as the consideration of the density, the algorithm also manages to generate the synthetic instances in more productive areas of the data space. Therefore, the generation of noisy examples is avoided. Although SOMO manages to identify efficient oversampling areas, the generation of synthetic instances handled by the SMOTE algorithm still provides several drawbacks. Challenges, as outlined in Figure 3d, are prevented through the above identification of clusters, nevertheless, the SMOTE algorithm introduces a high correlation between samples, since it only creates synthetic instances that are on the line segment between two minority class instances.

G-SMOTE extends the linear interpolation mechanism by introducing a geometric region, in which the data generation process occurs. Figure 4 illustrates this idea.

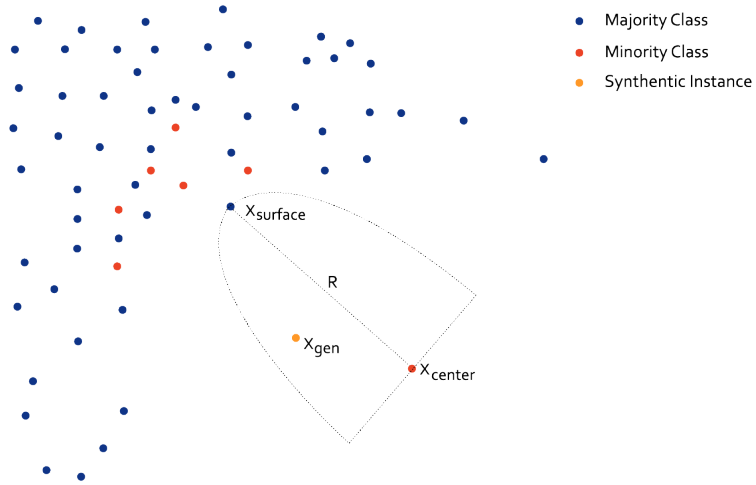

Figure 4: Synthetic data generation based on a geometric region, adopted from [Douzas and Bacao, 2019]

Besides utilizing a geometric region for the data generation mechanism, G-SMOTE also introduced the *majority* and *combined* selection strategies, which avoid the scenarios shown in Figures 3a, 3b and 3d. More specifically, the synthetic sample is created between the initially selected minority instance and the closest majority instance, as it can be seen in Figure 4. It has been empirically shown that G-SMOTE [Douzas and Bacao, 2019] significantly increased the performance of SMOTE.

SOMO provides an efficient process to identify areas that should be populated with minority instances, but lacks the improvements provided by G-SMOTE. On the other hand, G-SMOTE exploits an intelligent and efficient approach to generate synthetic instances, but lacks the ability to identify attractive regions for their generation. In this paper we propose a combination of both techniques, leveraging the benefits of both algorithms.

4 Proposed method

The previous section outlined the inefficiencies of the SMOTE algorithm, as well as of SOMO and G-SMOTE. The proposed method, G-SOMO, manages to tackle them, by utilizing the best characteristics of both algorithms. The method considers the density, based on the average Euclidean distances, in and between clusters when creating artificial data instances. Also, G-SOMO creates synthetic instances within a geometric region, increasing their variety. Finally, through the *majority* or *combined* selection

strategies, the creation of noisy examples is avoided. This should already be prevented through the identification of clusters, but still provides an advantage, in the case where the number of clusters were not correctly selected.

The proposed method consists of three stages:

1. Initially, the SOM algorithm is applied to the normalized input data set. Consequently, a mapping of the high dimensional data space into a low-dimensional space is created in such a way that the topological relations of the input patterns are preserved [Köküer et al., 2007]. To train the SOM, the Euclidean distance between the input vector and all neural weights has to be calculated. The neuron that has the shortest distance to the input vector (the winner) is chosen and its weights are slightly modified to the direction represented by the input vector. Afterward, the neighboring neurons weights are modified in the same direction [Brocki and Koržinek,]. Due to SOMs ability to preserve topological relations from high dimensional input spaces, insights in the underlying data structure can be retrieved by analyzing the two-dimensional output map.

2. In the second stage of the algorithm, filtered clusters are identified, i.e. clusters where the minority class dominates the majority class on the number of samples in the cluster. A weighted approach is used to create synthetic instances in the filtered clusters and also between neighboring filtered clusters.

3. The last stage of the algorithm is marked by the creation of synthetic instances within the identified filtered clusters as well as between them. The synthetic data generation is based on a geometric region, that is formed between the current minority instance and its selected neighbor. The shape of the geometric region varies, depending on various hyper-parameters.

4.1 G-SOMO Algorithm

The complete G-SOMO algorithm is listed below:

Algorithm 1: Pseudo code for G-SOMO implementation

Input : S_{maj}, S_{min}, N_{intra}, N_{inter}, IR_f, $SOM_{parameters}$, α_{trunc}, α_{def}, α_{sel}

Algorithm

1. Train a SOM with $SOM_{parameters}$ on the normalized input data set. Define as cl the collection of randomly assigned cluster labels on the SOM nodes that contain at least one minority class instance.

2. Define $cl_f = \{i \in cl : IR^i < IR_f\}$ and $cl_{f,n} = \{(i,j) \in cl_f \times cl_f : i,j \text{ topological neighbors }\}$ where IR^i is the IR of i cluster.

3. Calculate the density factor for each $i \in cl_f$ as $den^i = \dfrac{n_+^i}{(dist^i)^2}$ and for each $(i,j) \in cl_{f,n}$ as $den^{i,j} = den^i + den^j$ where n_+^i is the number of minority class samples in i cluster and $dist^i$ is the sum of Euclidean distances between all pairs of minority class samples in i cluster.

4. Calculate the sampling weights for the intra-cluster case as $w_{intra}^i = \dfrac{1/den^i}{\sum_{i \in cl_f} 1/den^i}$ and for the inter-cluster case as $w_{inter}^{(i,j)} = \dfrac{1/den^{(i,j)}}{\sum_{i,j \in cl_f} 1/den^{(i,j)}}$.

5. Calculate the number of artificial instances to be created for each $i \in cl_f$ from $[w_{intra}^i \cdot N_{intra}]$ and for each $(i,j) \in cl_{f,n}$ from $[w_{inter}^{(i,j)} \cdot N_{inter}]$.

6. Define as $S_{min,i}$ and $S_{maj,i}$ the sets of minority and majority class instances for i cluster, respectively. Set $S_{synthetic} = \emptyset$.

7. For each $i \in cl_f$ and each $(i,j) \in cl_{f,n}$ repeat the following:

   7.1. Shuffle $S_{min,i}$.

   7.2. Repeat until the calculated number of instances from step 6 is created:

      7.2.1. Select x_{center} from $S_{min,i}$.

      7.2.2. Apply the α_{sel} selection strategy and calculate a radius R. For the case of neighboring filtered clusters, use $S_{min,j}$ and $S_{maj,j}$ to search for the nearest neighbors.

      7.2.3. Generate a random point inside the unit-hypersphere centered at the origin of the input space.

      7.2.4. Truncate the unit hyper sphere using α_{trunc} hyper-parameter.

      7.2.5. Deform the unit hyper sphere to a hyper spheroid using α_{def} hyper-parameter.

      7.2.6. Rescale the created sample x_{gen} using the radius R.

      7.2.7. Add the sample x_{gen} to the set $S_{synthetic}$ of generated instances.

Output: $S' = S_{maj} \cup S_{min} \cup S_{synthetic}$

4.2 Explanation of G-SOMO

The G-SOMO algorithm relies on filtered clusters, which are areas where the number of minority class instances dominates over the number of majority class instances. These regions can be considered as safe areas for the generation of minority samples. G-SOMO assumes that noisy examples belong to non-filtered clusters and are ignored. The density and weights of each filtered cluster are calculated to create new instances in each filtered cluster (intra-cluster) and between filtered clusters that are neighbors on the topological output map of the SOM algorithm (inter-cluster). The geometric region used for the data generation process increases the variety of the generated instances.

Input

The G-SOMO algorithm requires the following input parameters:

- S_{maj} and S_{min} represent the set of instances of the majority and minority classes, respectively.
- N_{intra}, N_{inter} are the number of artificial instances that will be created within and between neighboring filtered clusters.
- IR_f represents the maximum imbalance ratio of any filtered cluster and takes positive values. Higher values allow more clusters to be included in the set of filtered clusters.
- $SOM_{parameters}$ represents the set of all SOM hyper-parameters. One notable SOM hyper-parameter is the size of the grid N_{grid}, which determines the number of nodes and therefore clusters.
- α_{trunc}, α_{def} and α_{sel}, known as geometric hyper-parameters, control the geometric characteristics of the G-SMOTE data generation mechanism.

Steps 1-5

These steps correspond to the SOMO algorithm minus the data generation phase. Initially the input data are normalized and SOM algorithm is applied with the provided hyper-parameters $SOM_{parameters}$. The set of filtered clusters cl_f is identified by comparing the imbalance ratio $IR_i = \frac{n_{-i}}{n_{+i}}$ of each cluster $i \in cl_f$, where n_{-i} and n_{+i} are the number of instances belonging to the minority and majority classes respectively, to a threshold called IR_f. Filtered clusters are areas of the input space where the number of minority instances dominate over the number of majority instances and therefore can be considered as safe areas to apply any local data generation mechanism. An important point is that the G-SOMO algorithm will not create any artificial instances if no filtered clusters are found. Additionally the set $cl_{f,n}$ of neighboring filtered clusters pairs is identified by including only pairs of filtered clusters that are topological neighbors.

For each identified filtered cluster i, the average Euclidean distance $dist_i$ is calculated between the minority instances. Using the average Euclidean distance, we define a density factor to each filtered cluster as $den_i = \frac{n_{+i}}{dist_i^2}$. This measure provides information about the distribution of the instances within each cluster and is used to assign the correct amount of artificial samples to each cluster. Also for each topological neighboring combination of filtered clusters (i, j), the density $den^{i,j}$ is defined as the sum of both individual density factors. Figure 5a illustrates the SOM Grid with all identified filtered clusters and Figure 5b outlines the topological neighbor structure.

Utilizing the density information, a relative weight w_{intra}^i is calculated for each filtered cluster, that represents the normalized inverse density. N_{intra} and N_{inter} provide the guideline of the total synthetic instances to be created in the intra-cluster and inter-cluster processes, respectively. N_{intra} times the

weight for each specific filtered cluster results in the amount of artificial data that is generated for the intra-cluster case. Similarly, the weights $w_{inter}^{(i,j)}$ of two neighboring filtered clusters (i, j) are calculated. Once each neighboring pair of filtered clusters has a relative weight assigned, N_{inter} times the relative weight results in the number of synthetic instances that are generated for the inter-cluster case. If no neighborhood relations exist then the artificial samples are created through the intra-cluster process.

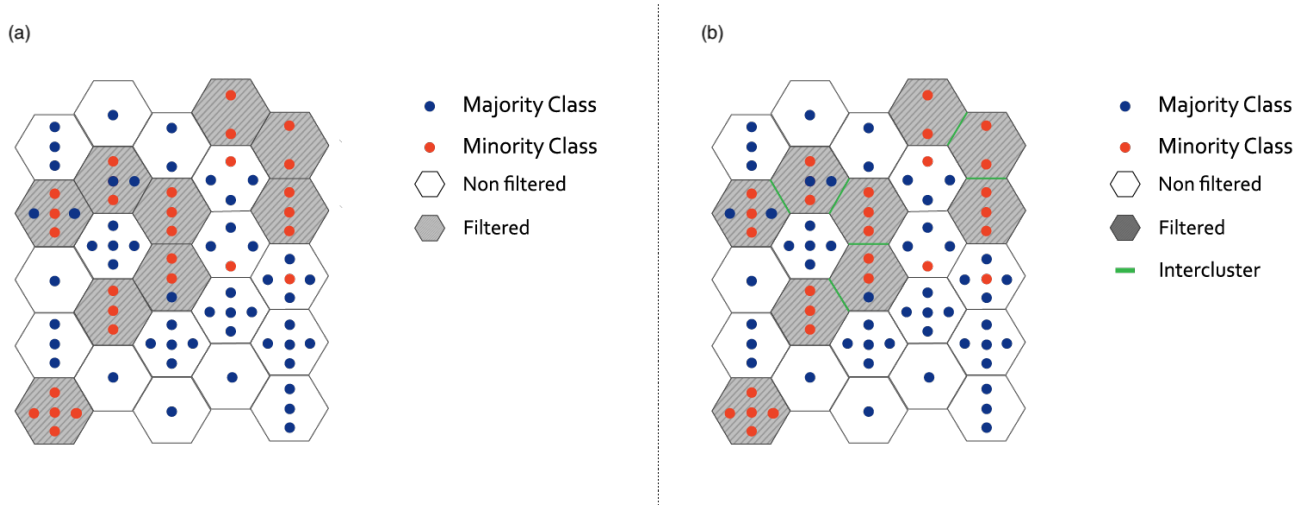

Figure 5: Overview of identified filtered clusters and neighboring relations, adopted from [Douzas and Bacao, 2017]

Step 7:

In previous steps, the numbers of required synthetic instances for each filtered cluster and each neighboring pair of filtered clusters were calculated. The following process, that corresponds to the G-SMOTE data generation mechanism, is applied to each one of them individually.

The set of minority instances is shuffled, minority instances are repetitively selected, where each minority instance can be selected multiple times if the number of required synthetic instances is higher than the number of minority instances. The current selected minority sample is named x_{center}. Based on x_{center} a selection strategy given by a_{sel} is applied and identifies $x_{surface}$, the final neighbor of x_{center}. Both are used to define the geometric area where new instances are generated and a radius R is defined as their Euclidean distance.

The data generation process starts by creating a unit hypersphere around the origin, as it can be seen in Figure 6a. This hypersphere is going to be modified within the next steps. Initially, a random point e_{sphere} is created on the surface of the unit hypersphere. Subsequent, e_{sphere} is transformed to x_{gen} a random point along the radius, uniformly distributed. Next, a truncation is applied, a partition of the hypersphere, as illustrated in 6b with α_{trunc} determining the degree of the truncation. Another transformation deforms the truncated hypersphere to a hyper spheroid, as it can be seen in 6c. The parameter α_{def} controls the degree of the deformation. As a final transformation we translate and rescale the point x_{gen} based on x_{center} and the radius R.

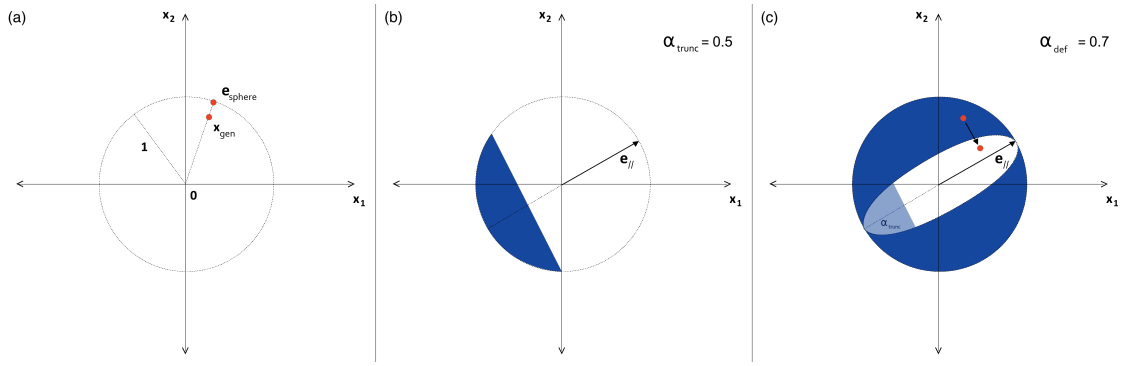

Figure 6: Constructing the geometric region, based on the unit hyper sphere, truncation and deformation, adopted from [Douzas and Bacao, 2017]

The process is repeated until $N_{intra} + N_{inter}$ are generated. The final output of the G-SOMO algorithm is the oversampled matrix S', consisting of the input data set and the set $S_{synthetic}$ of synthetic minority instances.

4.3 Time complexity of G-SOMO

As it was explained above, G-SOMO algorithm consists of three stages. Below we present the time complexity of G-SOMO, focusing on parameters that are related to the characteristics of the dataset.

The first stage is the application of the SOM algorithm. The time complexity of SOM is $\mathcal{O}(N_{samples} \cdot N_{grid}^2)$ where $N_{samples} = |S_{maj} \cup S_{min}|$ is the number of the dataset's observations.

The second and third stages include the identification of the filtered clusters and the creation of the synthetic instances using the G-SMOTE algorithm. Both of them require the calculation of the distance matrix between samples that has a time complexity $\mathcal{O}(N_{samples}^2 \cdot N_{features})$ where $N_{features}$ is the number of the dataset's features.

Since the number of SOM nodes N_{grid}^2 is less than the number of samples, the final time complexity of G-SOMO is $\mathcal{O}(N_{samples}^2 \cdot N_{features})$. While this is identical to the time complexity of the underlying G-SMOTE oversampler when the *majority* or *combined* selection strategies are used, it can be considered as a limitation of G-SOMO since it is higher than the time complexity of the SMOTE algorithm and some of its variations that calculate the nearest neighbors only on the minority class samples.

5 Research methodology

This section describes the evaluation process of G-SOMO using multiple classifiers, datasets and metrics. The experimental data as well as the experimental procedure are presented in details, while a description of the evaluation measures, the classification algorithms and the software implementation is provided.

5.1 Experimental data

To ensure meaningful and significant insights we evaluated our models on binary classification problems for a total of 69 datasets, mainly from the UCI Machine Learning repository. In order to reach this high

number of datasets, we randomly undersampled the minority class of some of them so that their IR is increased. Using this approach, we managed to create additional and more challenging datasets. Table 1 provides an overview of our datasets, the number of features, instances and their IR:

Dataset name	Features	Instances	Minority instances	Majority instances	Imbalance Ratio
HEART	13	270	120	150	1.25
LIVER	6	345	145	200	1.38
WINE	13	178	71	107	1.51
PIMA	8	769	268	501	1.87
BREAST TISSUE	9	106	36	70	1.94
IRIS	4	150	50	100	2.00
GLASS	9	214	70	144	2.06
YEAST 1	8	1,484	429	1,055	2.46
HEART (2)	13	210	60	150	2.50
LIVER (2)	6	272	72	200	2.78
HABERMAN	3	306	81	225	2.78
WINE (2)	13	142	35	107	3.06
VEHICLE	18	846	199	647	3.25
PIMA (2)	8	635	134	501	3.74
HEART (3)	13	190	40	150	3.75
BREAST TISSUE (2)	9	88	18	70	3.89
IRIS (2)	4	125	25	100	4.00
GLASS (2)	9	179	35	144	4.11
LIVER (3)	6	248	48	200	4.17
WINE (3)	13	130	23	107	4.65
YEAST 1 (2)	8	1,269	214	1,055	4.93
NEW THYROID 2	5	215	35	180	5.14
NEW THYROID 1	5	215	35	180	5.14
ECOLI	7	336	52	284	5.46
EUCALYPTUS	8	642	98	544	5.55
HABERMAN (2)	3	265	40	225	5.62
PIMA (3)	8	590	89	501	5.63
SEGMENTATION	16	2,310	330	1,980	6.00
IRIS (3)	4	116	16	100	6.25
GLASS (3)	9	167	23	144	6.26
VEHICLE (2)	18	746	99	647	6.54
YEAST 1 (3)	8	1,198	143	1,055	7.38
YEAST 3	8	1,484	163	1,321	8.10
HABERMAN (3)	3	252	27	225	8.33
PAGE BLOCKS 0	10	5,472	559	4,913	8.79
VEHICLE (3)	18	713	66	647	9.80
VOWEL	13	988	90	898	9.98
NEW THYROID 1 (2)	5	197	17	180	10.59
NEW THYROID 2 (2)	5	197	17	180	10.59
ECOLI (2)	7	310	26	284	10.92
LED	7	443	37	406	10.97
EUCALYPTUS (2)	8	593	49	544	11.10
SEGMENTATION (2)	16	2,145	165	1,980	12.00
LIBRAS	90	360	24	336	14.00
PAGE BLOCKS 1 3	10	472	28	444	15.86
YEAST 3 (2)	8	1,402	81	1,321	16.31
ECOLI (3)	7	301	17	284	16.71
DERMATOLOGY	34	358	20	338	16.90

Dataset name	Features	Instances	Minority instances	Majority instances	Imbalance Ratio
EUCALYPTUS (3)	8	576	32	544	17.00
PAGE BLOCKS 0 (2)	10	5,192	279	4,913	17.61
SEGMENTATION (3)	16	2,090	110	1,980	18.00
VOWEL (2)	13	943	45	898	19.96
LED (2)	7	424	18	406	22.56
YEAST 3 (3)	8	1,375	54	1,321	24.46
PAGE BLOCKS 0 (3)	10	5,099	186	4,913	26.41
MANDELON 1	20	4,000	142	3,858	27.17
MANDELON 2	200	3,000	105	2,895	27.57
YEAST 4	8	1,484	51	1,433	28.10
VOWEL (3)	13	928	30	898	29.93
YEAST 5	8	1,484	44	1,440	32.73
YEAST 6	8	1,484	35	1,449	41.40
MANDELON 1 (2)	20	3,929	71	3,858	54.34
MANDELON 2 (2)	200	2,947	52	2,895	55.67
YEAST 4 (2)	8	1,458	25	1,433	57.32
YEAST 5 (2)	8	1,462	22	1,440	65.45
MANDELON 1 (3)	20	3,905	47	3,858	82.09
MANDELON 2 (3)	200	2,930	35	2,895	82.71
YEAST 4 (3)	8	1,450	17	1,433	84.29
YEAST 6 (2)	8	1,466	17	1,449	85.24

Table 1: Overview of experimental data.

5.2 Evaluation measures

The model evaluation is based on different evaluation measures. Traditional metrics, such as accuracy, show a strong bias towards the majority class and should not be used on imbalanced datasets. The accuracy would seem to be precise, even though the model might not perform well on the minority class. To retrieve more accurate insights the following evaluation measures are used:

– Area Under The ROC Curve (AUC)

  The ROC Curve is created by plotting the True Positive Rate against the False Positive Rate as the decision threshold is changing [Hand, 2009].

– F-score

  It is defined as the harmonic mean between precision and recall, assuming that both metrics are of equal importance [Guo et al., 2018].

– Geometric Mean Score (G-mean)

  The geometric mean score, as the name implies, is defined as the geometric mean between sensitivity and specificity.

5.3 Machine learning algorithms

The goal of the paper is to validate the effectiveness of G-SOMO as oversampling method. Therefore G-SOMO was evaluated on the aforementioned 69 datasets and compared to ROS, SMOTE, K-Means SMOTE, SOMO and G-SMOTE. K-Means SMOTE was included in order to verify the assumption that replacing K-Means and SMOTE with SOM and G-SMOTE respectively results into a performance gain. Similarly, SOMO and G-SMOTE are included since G-SOMO, as a minimum requirement, is expected to improve the performance compared to its component algorithms. Additionally, the performance of the classifiers without applying any type of oversampling is also included in the experimental results (NO OVER).

Finally, it is crucial to ensure that the obtained results are generalizable to different classifiers and not only to specific ones, due to their different characteristics. Therefore, different classifiers are selected to evaluate the performance of the oversampling methods. These classifiers are Logistic Regression (LR) [McCullagh and Nelder, 1989], K-Nearest Neighbors (KNN) [Cover and Hart, 1967], Decision Tree (DT) [Salzberg, 1994] and Gradient Boosting Classifier (GBC) [Friedman, 2001].

5.4 Experimental procedure

In order to evaluate the performance of each combination of oversampler and classifier, n-fold cross-validation was applied with $n = 5$. The data set was splitted into n different subsets (also called folds). $n-1$ folds were used to train the classifier and generate artificial data by utilizing the oversampler while the last fold remained unseen to validate. The process was applied in an iterative manner, such that each fold was used for validation once. The validation scores of each model were averaged afterwards. The process was repeated three times to avoid bias due to random effects [Japkowicz, 2013].

A variety of hyper-parameters were used for the oversamplers. Below we group the common hyper-parameters and present their range of values:

- For SMOTE, K-Means SMOTE, SOMO, G-SMOTE, G-SOMO

  $k \in \{3, 5\}$ where k is the value of k-nearest neighbors.

- For K-Means SMOTE

  $N_{clusters} \in \{0\%, 25\%, 50\%, 75\%\}$ where $N_{clusters}$ is the number of K-Means clusters in units that correspond to the percentage of samples for each dataset (particulalrly 0% does not cluster the input space). The threshold IR_f was set equal to 1.0.

- For SOMO, G-SOMO

  $N_{grid} \in \{0\%, 25\%, 50\%, 75\%\}$ where N_{grid} is the dimension of the SOM grid in units that correspond to the percentage of samples for each dataset (particulalrly 0% does not cluster the input space). Also, instead of selecting N_{intra} and N_{inter} seperately, the ratio $\frac{N_{intra}}{N_{intra}+N_{inter}} \in \{0.75, 1.0\}$ is selected with the constrain that $N_{intra} + N_{inter} = N$, where N is the number of generated instances that make the initial dataset perfectly balanced. Finally, the threshold IR_f was set equal to 1.0.

- For G-SMOTE, G-SOMO

  $a_{trunc} \in \{-1.0, 0.0, 0.25, 1.0\}$, $a_{def} \in \{0.0, 0.25, 1.0\}$, $a_{sel} \in \{minority, majority, combined\}$ where a_{trunc}, a_{def} and a_{sel} are the truncation factor, deformation factor and selection strategy respectively.

Using the above hyper-parameters, the highest cross validation score for each combination of datasets, classifiers, oversamplers and evaluation metrics was reported. A ranking score was applied to compare the performance of the oversampling methods, aggregated over all datasets. The ranking score for the best performing method is 1, while for the worst performing method is 6. The Friedman test was applied to confirm the statistical significance of the ranking results [Guyon, 2003]. The Friedman test is used to detect differences for a set of experimental attempts when normality assumption may not hold. In this case the null hypothesis represents the situation in which the classifiers show an identical performance, in terms of their mean ranking, independently of the oversampling method and performance metric used. Additionally, the Holms test was applied, using G-SOMO as the control method [Guyon, 2003]. The Holms test is a non-parametric version of the t-test, where the null hypothesis is whether the proposed G-SOMO algorithm performs similarly to the other methods as the control method.

5.5 Software implementation

The implementation of the classifiers and oversamplers are based on the Python libraries Scikit-Learn [Pedregosa et al., 2011] and Imbalanced-Learn [Lemaitre et al., 2016]. All functions, algorithms, experiments and results reported are provided at the GitHub repository of the project, while the G-SOMO implementation is also available as an open-source library. Additionally, Research-Learn library provides a framework to implement comparative experiments, being also fully integrated with the Scikit-Learn ecosystem.

6 Results and discussion

In this section the performance of the different oversamplers is presented. The results are analyzed, by applying various statistical tests, to establish their significance. Additionally, a taxonomy of G-SOMO hyper-parameters is presented as well as guidelines for their tuning relative to the characteristics of the imbalanced datasets.

6.1 Comparative presentation

The mean and standard error of cross validation scores across datasets for each combination of classifiers, metrics and oversamplers are presented in Table 2:

Classifier	Metric	NO OVER	ROS	SMOTE	K-MEANS SMOTE	SOMO	G-SMOTE	G-SOMO
LR	AUC	0.89 ± 0.02	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.01	0.91 ± 0.01	0.90 ± 0.01	**0.93** ± 0.01
LR	F-SCORE	0.54 ± 0.05	0.67 ± 0.03	0.68 ± 0.03	0.71 ± 0.03	0.72 ± 0.03	0.69 ± 0.03	**0.74** ± 0.03
LR	G-MEAN	0.59 ± 0.05	0.84 ± 0.02	0.84 ± 0.02	0.85 ± 0.02	0.86 ± 0.02	0.85 ± 0.02	**0.88** ± 0.02
KNN	AUC	0.83 ± 0.02	0.82 ± 0.02	0.84 ± 0.02	0.84 ± 0.02	0.85 ± 0.02	0.85 ± 0.02	**0.86** ± 0.02
KNN	F-SCORE	0.58 ± 0.04	0.62 ± 0.03	0.64 ± 0.03	0.67 ± 0.03	0.68 ± 0.03	0.67 ± 0.03	**0.69** ± 0.03
KNN	G-MEAN	0.66 ± 0.03	0.78 ± 0.02	0.80 ± 0.02	0.80 ± 0.02	0.81 ± 0.02	0.81 ± 0.02	**0.82** ± 0.02
DT	AUC	0.83 ± 0.02	0.83 ± 0.02	0.85 ± 0.02	0.87 ± 0.01	0.87 ± 0.01	0.88 ± 0.02	**0.89** ± 0.01
DT	F-SCORE	0.63 ± 0.04	0.65 ± 0.03	0.68 ± 0.03	0.71 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	**0.74** ± 0.03
DT	G-MEAN	0.73 ± 0.03	0.80 ± 0.02	0.82 ± 0.02	0.83 ± 0.02	0.84 ± 0.02	0.85 ± 0.02	**0.86** ± 0.02
GBC	AUC	0.90 ± 0.01	0.89 ± 0.02	0.90 ± 0.02	0.91 ± 0.01	0.92 ± 0.01	0.91 ± 0.01	**0.93** ± 0.01
GBC	F-SCORE	0.68 ± 0.04	0.69 ± 0.03	0.71 ± 0.03	0.74 ± 0.03	0.74 ± 0.03	**0.76** ± 0.03	**0.76** ± 0.03
GBC	G-MEAN	0.76 ± 0.03	0.81 ± 0.02	0.82 ± 0.02	0.83 ± 0.02	0.84 ± 0.02	**0.85** ± 0.02	**0.85** ± 0.02

Table 2: Results for mean cross validation scores of oversamplers across the datasets.

Table 2 shows that G-SOMO performs better on average than the rest of the methods. Particularly, the mean and standard error of percentage difference between G-SOMO and SMOTE, K-Means SMOTE, SOMO and G-SMOTE is presented in Table 3:

Classifier	Metric	SMOTE	K-MEANS SMOTE	SOMO	G-SMOTE
LR	AUC	3.55 ± 0.13	2.88 ± 0.04	1.92 ± 0.03	3.12 ± 0.11
LR	F-SCORE	10.28 ± 1.24	3.75 ± 0.18	2.60 ± 0.11	8.34 ± 1.30
LR	G-MEAN	4.62 ± 0.34	3.02 ± 0.06	2.03 ± 0.04	2.97 ± 0.33
KNN	AUC	3.22 ± 0.51	3.08 ± 0.06	2.08 ± 0.04	1.42 ± 0.44
KNN	F-SCORE	10.52 ± 1.29	4.12 ± 0.22	2.84 ± 0.15	4.40 ± 0.73
KNN	G-MEAN	3.62 ± 0.60	3.31 ± 0.07	2.20 ± 0.05	1.59 ± 0.40
DT	AUC	4.74 ± 0.37	2.85 ± 0.06	2.00 ± 0.04	1.49 ± 0.37
DT	F-SCORE	9.64 ± 1.49	3.49 ± 0.18	2.57 ± 0.12	1.95 ± 0.82
DT	G-MEAN	4.97 ± 0.58	2.94 ± 0.07	2.09 ± 0.04	1.13 ± 0.53
GBC	AUC	4.04 ± 0.22	2.77 ± 0.05	1.91 ± 0.03	2.14 ± 0.31
GBC	F-SCORE	7.53 ± 0.95	3.55 ± 0.16	2.51 ± 0.12	0.73 ± 0.79
GBC	G-MEAN	3.90 ± 0.52	2.88 ± 0.08	2.09 ± 0.05	0.00 ± 0.57

Table 3: Results for percentage difference between G-SOMO and the rest of main oversamplers.

Table 4 below illustrates the mean ranking of each oversampler on each metric and each classifier averaged across all datasets:

Classifier	Metric	NO OVER	ROS	SMOTE	K-MEANS SMOTE	SOMO	G-SMOTE	G-SOMO
LR	AUC	6.05 ± 0.16	5.67 ± 0.13	5.52 ± 0.10	3.68 ± 0.13	2.02 ± 0.02	4.06 ± 0.11	**1.00** ± 0.00
LR	F-SCORE	6.11 ± 0.19	5.82 ± 0.13	5.21 ± 0.14	3.52 ± 0.12	2.17 ± 0.07	4.12 ± 0.15	**1.04** ± 0.04
LR	G-MEAN	6.61 ± 0.13	5.43 ± 0.14	5.25 ± 0.13	3.85 ± 0.13	2.33 ± 0.09	3.44 ± 0.17	**1.08** ± 0.05
KNN	AUC	5.92 ± 0.13	6.26 ± 0.13	4.65 ± 0.23	4.20 ± 0.13	2.69 ± 0.12	2.92 ± 0.20	**1.38** ± 0.10
KNN	F-SCORE	6.29 ± 0.17	5.80 ± 0.12	5.12 ± 0.18	3.80 ± 0.12	2.40 ± 0.09	3.42 ± 0.19	**1.17** ± 0.07
KNN	G-MEAN	6.62 ± 0.13	5.69 ± 0.11	4.38 ± 0.22	4.28 ± 0.13	2.71 ± 0.12	2.95 ± 0.20	**1.38** ± 0.11
DT	AUC	6.11 ± 0.14	6.07 ± 0.11	5.27 ± 0.17	3.96 ± 0.11	2.65 ± 0.10	2.73 ± 0.19	**1.21** ± 0.07
DT	F-SCORE	6.07 ± 0.16	6.10 ± 0.11	5.31 ± 0.14	3.73 ± 0.12	2.56 ± 0.09	3.05 ± 0.20	**1.17** ± 0.06
DT	G-MEAN	6.45 ± 0.13	5.77 ± 0.12	5.22 ± 0.15	3.97 ± 0.12	2.67 ± 0.10	2.70 ± 0.19	**1.23** ± 0.07
GBC	AUC	5.93 ± 0.15	6.10 ± 0.11	5.48 ± 0.14	3.96 ± 0.10	2.31 ± 0.07	3.18 ± 0.16	**1.04** ± 0.03
GBC	F-SCORE	6.17 ± 0.15	5.93 ± 0.10	5.42 ± 0.14	3.97 ± 0.08	2.60 ± 0.09	2.58 ± 0.21	**1.33** ± 0.07
GBC	G-MEAN	6.42 ± 0.12	5.80 ± 0.14	4.93 ± 0.18	4.20 ± 0.11	2.96 ± 0.12	2.39 ± 0.18	**1.31** ± 0.07

Table 4: Results for mean ranking of oversamplers across the datasets.

Based on the insights from the above tables we can conclude, that G-SOMO successfully outperforms the rest of oversampling methods, for any combination of metrics and classifiers.

6.2 Statistical analysis

The results of the application of the Friedman test are shown in Table 5:

Classifier	Metric	p-value	Significance
LR	AUC	$6.2 \cdot 10^{-50}$	True
LR	F-SCORE	$8.7 \cdot 10^{-47}$	True
LR	G-MEAN	$1.1 \cdot 10^{-47}$	True
KNN	AUC	$6.3 \cdot 10^{-41}$	True
KNN	F-SCORE	$1.1 \cdot 10^{-44}$	True
KNN	G-MEAN	$3.5 \cdot 10^{-42}$	True

Classifier	Metric	p-value	Significance
DT	AUC	$1.3 \cdot 10^{-46}$	True
DT	F-SCORE	$2.5 \cdot 10^{-46}$	True
DT	G-MEAN	$6.9 \cdot 10^{-47}$	True
GBC	AUC	$8.7 \cdot 10^{-49}$	True
GBC	F-SCORE	$1.6 \cdot 10^{-46}$	True
GBC	G-MEAN	$3.4 \cdot 10^{-45}$	True

Table 5: Results for Friedman test.

Therefore at a significance level of $a = 0.05$ the null hypothesis is rejected, i.e. the classifiers do not perform similarly in the mean rankings across the oversampling methods and evaluation metrics. Following the Friedman test, the Holm's method is applied to adjust the p-values of the paired difference test with G-SOMO algorithm as the control method. The adjusted p-values are presented in Table 6:

Classifier	Metric	NO OVER	ROS	SMOTE	K-MEANS SMOTE	SOMO	G-SMOTE
LR	AUC	$7.7 \cdot 10^{-19}$	$1.4 \cdot 10^{-37}$	$1.4 \cdot 10^{-37}$	$6.5 \cdot 10^{-55}$	$6.7 \cdot 10^{-75}$	$6.9 \cdot 10^{-37}$
LR	F-SCORE	$3.1 \cdot 10^{-9}$	$3.7 \cdot 10^{-10}$	$2.4 \cdot 10^{-12}$	$1.4 \cdot 10^{-44}$	$2.0 \cdot 10^{-73}$	$2.4 \cdot 10^{-9}$
LR	G-MEAN	$8.7 \cdot 10^{-9}$	$7.0 \cdot 10^{-18}$	$1.4 \cdot 10^{-20}$	$9.8 \cdot 10^{-54}$	$1.2 \cdot 10^{-72}$	$3.0 \cdot 10^{-13}$
KNN	AUC	$1.7 \cdot 10^{-26}$	$3.2 \cdot 10^{-24}$	$3.4 \cdot 10^{-7}$	$4.0 \cdot 10^{-54}$	$1.6 \cdot 10^{-74}$	$3.4 \cdot 10^{-3}$
KNN	F-SCORE	$4.7 \cdot 10^{-11}$	$6.7 \cdot 10^{-12}$	$1.6 \cdot 10^{-10}$	$6.7 \cdot 10^{-48}$	$2.3 \cdot 10^{-73}$	$1.5 \cdot 10^{-6}$
KNN	G-MEAN	$5.9 \cdot 10^{-10}$	$5.9 \cdot 10^{-19}$	$8.9 \cdot 10^{-7}$	$2.8 \cdot 10^{-58}$	$5.7 \cdot 10^{-73}$	$4.9 \cdot 10^{-4}$
DT	AUC	$8.6 \cdot 10^{-14}$	$3.0 \cdot 10^{-15}$	$4.2 \cdot 10^{-18}$	$2.9 \cdot 10^{-45}$	$2.5 \cdot 10^{-71}$	$1.1 \cdot 10^{-4}$
DT	F-SCORE	$3.8 \cdot 10^{-10}$	$1.7 \cdot 10^{-14}$	$1.8 \cdot 10^{-8}$	$2.2 \cdot 10^{-37}$	$3.5 \cdot 10^{-71}$	$1.9 \cdot 10^{-2}$
DT	G-MEAN	$1.1 \cdot 10^{-9}$	$8.6 \cdot 10^{-17}$	$1.3 \cdot 10^{-10}$	$3.5 \cdot 10^{-42}$	$3.2 \cdot 10^{-70}$	$6.2 \cdot 10^{-2}$
GBC	AUC	$4.6 \cdot 10^{-25}$	$6.5 \cdot 10^{-24}$	$3.2 \cdot 10^{-27}$	$1.4 \cdot 10^{-51}$	$9.2 \cdot 10^{-74}$	$3.6 \cdot 10^{-10}$
GBC	F-SCORE	$1.2 \cdot 10^{-11}$	$2.5 \cdot 10^{-12}$	$3.9 \cdot 10^{-10}$	$1.2 \cdot 10^{-47}$	$2.1 \cdot 10^{-69}$	$3.4 \cdot 10^{-1}$
GBC	G-MEAN	$5.8 \cdot 10^{-9}$	$7.5 \cdot 10^{-16}$	$6.3 \cdot 10^{-9}$	$7.8 \cdot 10^{-38}$	$1.9 \cdot 10^{-71}$	$9.6 \cdot 10^{-1}$

Table 6: Adjusted p-values using the Holm's method.

Therefore, the null hypothesis of the Holm's test is rejected at a significance level of $a = 0.05$ for 69 out of 72 cases, i.e. the hypothesis that G-SOMO's performance is similar to the rest of the methods is rejected. The 3 remaining cases correspond to G-SOMO and G-SMOTE performance comparison.

6.3 Analysis and tuning of optimal hyper-parameters

G-SOMO's hyper-parameters can be divided into two different categories. The first category is related to the SOMO parametrization while the second category are the α_{trunc}, α_{def}, α_{sel} hyper-parameters that control the G-SMOTE data generation mechanism.

The most crucial hyper-parameter of the first category is the optimal size N_{grid} of the SOM grid. After SOM training is applied, the high dimensional input space will be transformed into a two-dimensional grid that consists of N_{grid}^2 clusters, which are used to identify safe areas for the data generation process. The challenge hereby is to select a value allowing to discriminate between sparse and dense minority class areas. A very small value will not be able to identify sub-clusters, since the identified clusters will have a very large size of instances. Assuming a uniform distribution on the SOM map, one can set the threshold to $\sqrt{|S_{min}|}$ to ensure that each cluster contains on average one minority class. On the other hand, high values of N_{grid} will result in smaller sized clusters, which might lead to filtered clusters in areas that we would usually like to ignore, because they are considered outliers. $\sqrt{|S_{maj}|}$ provides a reasonable upper bound for N_{grid}. The optimal value for N_{grid} is dependent on the characteristics of the

data set and can only be approximated in an experimental approach.

An analysis of the second category of G-SOMO hyper-parameters shows a dependence on Imbalance Ratio (IR) as well as the number of samples over number of features ratio (R). Specifically we can conclude the following:

- High IR or low R.

  The *majority* or *combined* selection strategy and lower absolute values of the truncation and deformation factors dominate as optimal hyper-parameters.

- Low IR or high R.

  The *minority* selection strategy and higher absolute values of the truncation and deformation factors dominate as optimal hyper-parameters.

7 Conclusion

In this paper, we proposed a new oversampling algorithm G-SOMO. The algorithm examines the characteristics of the multi-dimensional input space while grouping the data to identify filtered clusters where minority instances dominate. Within these filtered clusters, as well as between neighboring filtered clusters, synthetic minority class instances are created inside the geometric region of a hyper-ellipsis.

G-SOMO was evaluated on binary classification problems using 69 different datasets and compared to popular oversampling methods that included also G-SOMO components i.e. SOMO and G-SMOTE algorithms. The performance was evaluated using the AUC, F-score and G-mean metrics. In order to avoid dependence of the results to a specific classifier, multiple ones were selected that differ in their characteristics. Each experiment was repeated 3 times with a 5-fold cross-validation procedure. The statistical significance of the results was established by appropriate statistical tests.

In particular, our empirical results highlight the need for oversampling algorithms. Utilizing no oversampling method produced the worst results in our mean ranking, while simple methods like ROS increased the performance when compared with it. The popular method SMOTE performed better than ROS, whereas it was confirmed that K-Means SMOTE, SOMO and G-SMOTE performed better than SMOTE. G-SOMO dominated the ranking and outperformed all the other oversamplers. Therefore, we are confident that G-SOMO is a new appealing approach for researchers and practitioners working with imbalanced datasets.

Future work may include the behavior of the algorithm on multi-class imbalanced datasets, since research about oversampling on these type of classification problems is limited compared to the binary case. Additionally, the data we used include only numerical features, therefore the underlying G-SMOTE data generation mechanism may be extended to generate also categorical data. While the proposed algorithm outperforms the rest state-of-the-art oversampling methods, a shift from the nearest-neighbors paradigm, using Conditional Generative Adversarial Network (CGAN) as an oversampler, may be worth to explore. An initial exploration of this approach has already been done in [Douzas and Bacao, 2018] with encouraging results. Finally, another important future work is the theoretical analysis of various oversampling methods, including ROS, SMOTE and G-SMOTE.

Funding

This research was funded by Fundação para a Ciência e Tecnologia (Portugal)
Grant number DSAIPA/DS/0116/2019 – MapIntel.

References

[Barua et al., 2014] Barua, S., Islam, M. M., Yao, X., and Murase, K. (2014). MWMOTE - Majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425.

[Batista et al., 2004] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20.

[Brocki and Koržinek,] Brocki, Ł. and Koržinek, D. Kohonen self-organizing map for the traveling salesperson problem. In *Recent Advances in Mechatronics*, pages 116–119. Springer Berlin Heidelberg.

[Bunkhumpornpat et al., 2009] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5476 LNAI, pages 475–482.

[Bunkhumpornpat et al., 2012] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2012). DBSMOTE: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, 36(3):664–684.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357.

[Chawla et al., 2003] Chawla, N. V., Lazarevic, A., Hall, L., and Boyer, K. (2003). SMOTEBoost: improving prediction of the minority class in boosting. *Principles of Knowledge Discovery in Databases, PKDD-2003*, pages 107–119.

[Cieslak et al., 2006] Cieslak, D. A., Chawla, N. V., and Striegel, A. (2006). Combating imbalance in network intrusion datasets. In *2006 IEEE International Conference on Granular Computing*, pages 732–737.

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

[Domingos, 1999] Domingos, P. (1999). MetaCost: A General Method for Making Classifiers Cost-Sensitive. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 55:155–164.

[Douzas and Bacao, 2017] Douzas, G. and Bacao, F. (2017). Self-organizing map oversampling (somo) for imbalanced data set learning. *Expert Systems with Applications*, 82:40 – 52.

[Douzas and Bacao, 2018] Douzas, G. and Bacao, F. (2018). Effective data generation for imbalanced

learning using conditional generative adversarial networks. *Expert Systems with Applications*, 91:464–471.

[Douzas and Bacao, 2019] Douzas, G. and Bacao, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. *Information Sciences*.

[Douzas et al., 2018] Douzas, G., Bacao, F., and Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1 – 20.

[Fernandez et al., 2018] Fernandez, A., Garcia, S., Herrera, F., and Chawla, N. V. (2018). SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905.

[Fernández et al., 2013] Fernández, A., López, V., Galar, M., del Jesus, M. J., and Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97–110.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

[Ganganwar, 2012] Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. 2:42–47.

[Guo and Viktor, 2004] Guo, H. and Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data Generation : The DataBoost-IM approach. *ACM SIGKD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):30–39.

[Guo et al., 2018] Guo, H., Zhou, J., and Wu, C.-A. (2018). Imbalanced learning based on data-partition and SMOTE. *Information*, 9(9):238.

[Guyon, 2003] Guyon, I. (2003). Design of experiments of the NIPS 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction*.

[Haixiang et al., 2017] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., and Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239.

[Han et al., 2005] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Lecture Notes in Computer Science*, pages 878–887. Springer Berlin Heidelberg.

[Hand, 2009] Hand, D. J. (2009). Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123.

[He and Garcia, 2009] He, H. and Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

[Japkowicz, 2013] Japkowicz, N. (2013). Assessment metrics for imbalanced learning. In *Imbalanced Learning*, pages 187–206. John Wiley & Sons, Inc.

[Köküer et al., 2007] Köküer, M., Naguib, R. N., Jančovič, P., Younghusband, H. B., and Green, R. (2007). Towards automatic risk analysis for hereditary non-polyposis colorectal cancer based on pedigree data. In *Outcome Prediction in Cancer*, pages 319–337. Elsevier.

[Kubat and Matwin, 1997] Kubat, M. and Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One Sided Selection. *In Proceedings of the Fourteenth International Conference on Machine Learning*, 97:179–186.

[Lemaitre et al., 2016] Lemaitre, G., Nogueira, F., and Aridas, C. K. (2016). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18:1–5.

[McCullagh and Nelder, 1989] McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*.

[Nekooeimehr and Lai-Yuen, 2016] Nekooeimehr, I. and Lai-Yuen, S. K. (2016). Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets. *Expert Systems with Applications*, 46:405–416.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.

[Prati et al., 2004a] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C. (2004a). Class imbalances versus class overlapping: An analysis of a learning system behavior. In *MICAI 2004: Advances in Artificial Intelligence*, pages 312–321. Springer Berlin Heidelberg.

[Prati et al., 2004b] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C. (2004b). Learning with class skews and small disjuncts. In *Advances in Artificial Intelligence – SBIA 2004*, pages 296–306. Springer Berlin Heidelberg.

[Rodriguez et al., 2012] Rodriguez, E., Snasel, V., Abraham, A., Wozniak, M., Grana, M., and Cho, S. (2012). *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012, Salamanca, Spain, March 28-30th, 2012, Proceedings*. Number Teil 2 in Lecture Notes in Computer Science. Springer Berlin Heidelberg.

[Sáez et al., 2016] Sáez, J. A., Krawczyk, B., and Woźniak, M. (2016). Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, 57:164–178.

[Salzberg, 1994] Salzberg, S. L. (1994). C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240.

[Schubach et al., 2017] Schubach, M., Re, M., Robinson, P. N., and Valentini, G. (2017). Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants. *Scientific Reports*, 7(1).

[Tang and He, 2015] Tang, B. and He, H. (2015). KernelADASYN: Kernel based adaptive synthetic data generation for imbalanced learning. In *2015 IEEE Congress on Evolutionary Computation*. IEEE.

[Tang and Gao, 2007] Tang, Y. and Gao, J. (2007). Improved classification for problem involving overlapping patterns. *IEICE Transactions on Information and Systems*, E90-D(11):1787–1795.

[Ting, 2002] Ting, K. M. (2002). An Instance-Weighting Method to Induce Cost-Sensitive Trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665.

[Vong et al., 2014] Vong, C.-M., Ip, W.-F., Chiu, C.-C., and Wong, P.-K. (2014). Imbalanced learning for

air pollution by meta-cognitive online sequential extreme learning machine. *Cognitive Computation*, 7(3):381–391.

[Wan et al., 2014] Wan, X., Liu, J., Cheung, W. K., and Tong, T. (2014). Learning to improve medical decision making from imbalanced data without a priori cost. *BMC Medical Informatics and Decision Making*, 14(1).

[Yen and Lee,] Yen, S.-J. and Lee, Y.-S. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer Berlin Heidelberg.