geometric-smote: A package for flexible and efficient over-sampling

Georgios Douzas, Fernando Bacao*

NOVA Information Management School, Universidade Nova de Lisboa

---

Abstract

Classification of imbalanced datasets is a challenging task for standard algorithms. G-SMOTE is a state-of-the-art oversampling algorithm that has been shown to outperform the popular SMOTE oversampler and its variations. In this paper, a Python implementation of G-SMOTE is provided that integrates with the Scikit-Learn ecosystem. Therefore, machine learning researchers and practitioners can benefit from its use in a straightforward manner.

Keywords: Machine learning, Imbalanced learning problem, Oversampling

---

*Postal Address: NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal, Telephone: +351 21 382 8610

Email addresses: `gdouzas@novaims.unl.pt` (Georgios Douzas), `bacao@novaims.unl.pt` (Fernando Bacao)

Code metadata	
Current code version	v0.1.2
Permanent link to code/repository used for this code version	`https://github.com/AlgoWit/geometric-smote`
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Python, Travis CI, AppVeyor, Read the Docs, Codecov, CircleCI, zenodo, Anaconda Cloud
Compilation requirements, operating environments & dependencies	Linux, Mac OS, Windows
If available Link to developer documentation/manual	`https://geometric-smote.readthedocs.io/`
Support email for questions	georgios.douzas@gmail.com

Table 1: Code metadata

1. Motivation and significance

1.1. Introduction

The imbalanced learning problem is defined as a machine learning classification task using datasets with binary or multi-class targets where one of the classes, called the majority class, outnumbers significantly the remaining classes, called the minority class(es) [1]. Learning from imbalanced data is a frequent non-trivial problem for academic researchers and industry practitioners. The imbalance learning problem can be found in multiple domains such as chemical and biochemical engineering, financial management, information technology, security, business, agriculture or emergency management [2].

The Imbalance Ratio (IR) is defined as the ratio between the number of samples of the majority class and each of the minority classes. IR values between 100 and 100.000 have been observed [3], [4]. Standard machine learning classification algorithms induce a bias towards the majority class during training. This results in low performance when metrics suitable for imbalanced data are used for the classifier's evaluation. Figure 1 shows an example of imbalanced data in two dimensions and the resulting decision boundary of a typical classifier when they are used as a training set.

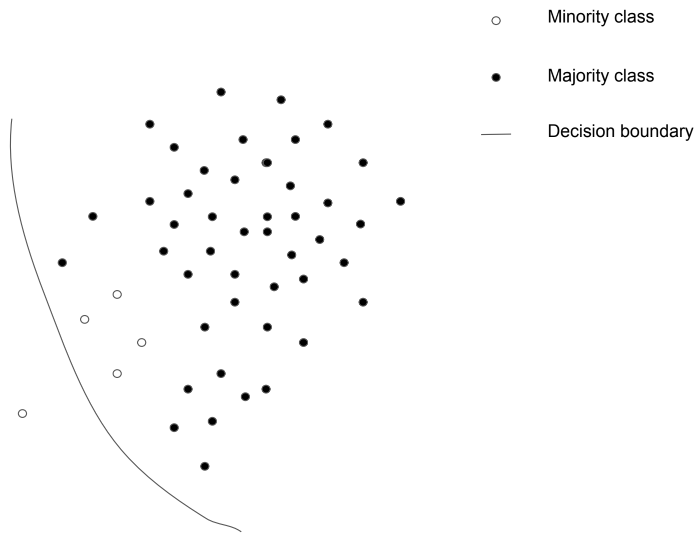

Figure 1: Imbalanced data in two dimensions. The decision boundary of a classifier shows a bias towards the majority class,

1.2. Oversampling algorithms

Various approaches have been proposed to deal with the imbalanced learning problem [5]. One general approach is the modification at the data level by oversampling the minority class(es). Synthetic Minority Oversampling Technique (SMOTE) [3] is the most popular oversampling algorithm. It generates synthetic instances along a line segment that joins minority class instances. Although SMOTE has been shown to be effective for oversampling imbalanced data, it also has some weaknesses [6]. In order to improve the quality of the generated data, many variations of SMOTE have been proposed. Nevertheless, all of these variations use the same data generation mechanism, i.e. linear interpolation between minority class samples.

A Python implementation of SMOTE and a couple of its variations is available in the Imbalanced-Learn [7] library. Imbalanced-Learn is fully compatible with Scikit-Learn [8], one of the most popular machine learning libraries.

1.3. Geometric SMOTE

Geometric SMOTE (G-SMOTE) [9] uses a different approach compared to existing SMOTE's variations. More specifically, G-SMOTE oversampling algorithm substitutes the data generation mechanism of SMOTE by defining a flexible geometric region around each minority class instance and generating synthetic instances inside the boundaries of this region. The algorithm requires the selection of the hyperparameters `truncation_factor`, `deformation_factor`, `selection_strategy` and `k_neighbors`. The first three of them, called geometric hyperparameters, control the shape of the geometric region while the later adjusts its size. Figure 2 presents a visual comparison between the data generation mechanisms of SMOTE and G-SMOTE.

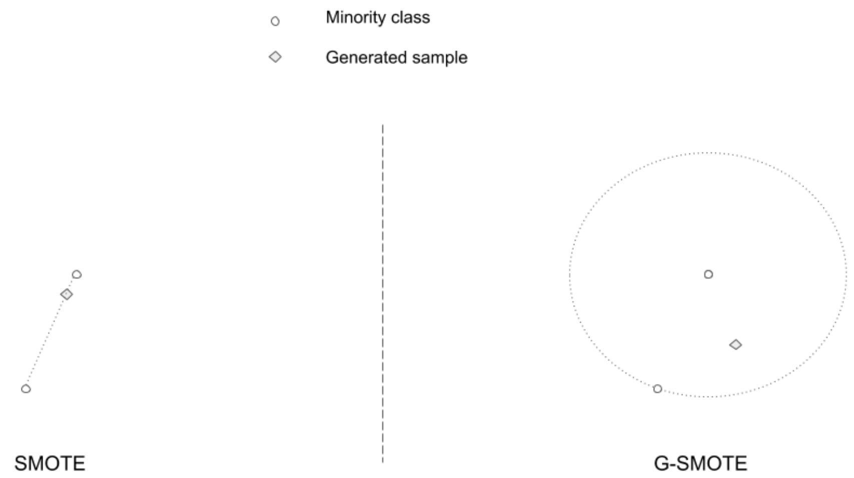

Figure 2: Comparison between the data generation mechanisms of SMOTE and G-SMOTE. SMOTE uses linear interpolation, while G-SMOTE defines a circle as the permissible data generation area.

G-SMOTE algorithm has been shown to outperform SMOTE across 69 imbalanced datasets for various classifiers and evaluation metrics. This paper, presents a Python implementation of G-SMOTE. In section 2, the software description is given while section 3 provides a demonstrative example of its functionalities.

2. Software description

The `geometric-smote` project is written in Python 3.7. It contains an object-oriented implementation of the G-SMOTE algorithm as well as an extensive online documentation. The implementation provides an API that is compatible with Imbalanced-Learn and Scikit-Learn libraries, therefore it makes full use of various features that support standard machine learning functionalities.

2.1. Software Architecture

The `geometric-smote` project contains the Python package `gsmote`. The main module of `gsmote` is called `geometric-smote`. It contains the class `GeometricSMOTE` that implements the G-SMOTE algorithm. The initialization of an `GeometricSMOTE` instance includes G-SMOTE's hyperparameters that control the generation of synthetic data. Additionally, `GeometricSMOTE` inherits from the `BaseOverSampler` class of Imbalanced-Learn library. Therefore, an instance of `GeometricSMOTE` class provides the `fit` and `fit_resample` methods, the two main methods for resampling as explained in subsection 2.2. This is achieved by implementing the `_fit_resample` abstract method of the parent class `BaseOverSampler`. More specifically, the function `_make_geometric_sample` implements the data generation mechanism of G-SMOTE as shortly described in section 1.3. This function is called in the `_make_geometric_samples` method of the `GeometricSMOTE` class in order to generate the appropriate number of synthetic data for a particular minority class. Finally, the method `_make_geometric_samples` is called in `_fit_resample` method to generate synthetic data for all minority classes. Figure 3 provides a visual representation of the above classes and functions hierarchy.

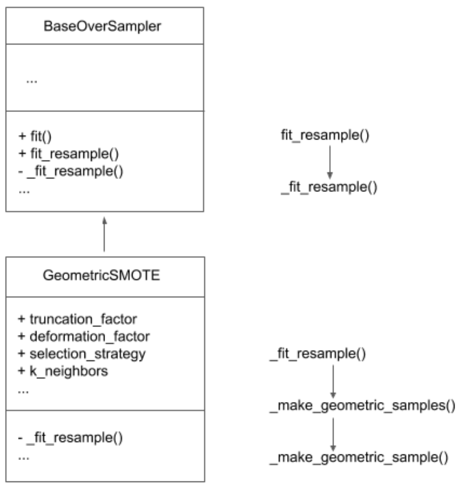

Figure 3: UML class diagrams and callgraphs of main classes and methods.

2.2. Software Functionalities

As it was mentioned in subsection 2.1, the class `GeometricSMOTE` represents the G-SMOTE oversampler. The intializer of `GeometricSMOTE` includes the following G-SMOTE's hyperparameters: `truncation_factor`, `deformation_factor`, selection_strategy and k_neighbors as explained in subsection 1.3. Once the GeometricSMOTE object is initialized with a specific parametrization, it can be used to resample the imbalanced data represented by the input matrix X and the target labels y. Following the Scikit-Learn API, both X, y are array-like objects of appropriate shape.

Resampling is achieved by using the two main methods of fit and fit_resample the GeometricSMOTE object. More specifically, both of them take as input parameters the X and y. The first method computes various statistics which are used to resample X while the second method does the same but additionally returns a resampled version of X and y.

The geometric-smote project has been designed to integrate with the Imbalanced-Learn toolbox and Scikit-Learn ecosystem. Therefore the GeometricSMOTE object can be used in a machine learning pipeline, through Imbalanced-Learn's class Pipeline, that automatically combines samplers, transformers and estimators. The next section provides examples of the above functionalities.

3. Illustrative Examples

3.1. Basic example

An example of resampling multi-class imbalanced data using the fit_resample method is presented in Listing 1. Initially, a 3-class imbalanced dataset is generated. Next, GeometricSMOTE object is initialized with default values for the hyperparameters, i.e. truncation_factor = 1.0, deformation_factor = 0.0, selection_strategy = combined. Finally, the object's fit_resample method is used to resample the data. Printing the class distribution before and after resampling confirms that the resampled data are perfectly balanced. The resampled data X_res, y_res can be used as training data for any classifier in the place of X, y.

Listing 1: Resampling of imbalanced data using the fit_resample method.

```
# Import classes and functions.
from collections import Counter
from gsmote import GeometricSMOTE
from sklearn.datasets import make_classification

# Generate an imbalanced 3-class dataset.
X, y = make_classification(
    random_state=23,
    n_classes=3,
    n_informative=5,
```

```
    n_samples=500,
    weights=[0.8, 0.15, 0.05]
)

# Create a GeometricSMOTE object with default hyperparameters.
gsmote = GeometricSMOTE(random_state=10)

# Resample the imbalanced dataset.
X_res, y_res = gsmote.fit_resample(X, y)

# Print number of samples per class for initial and resampled data.
init_count = list(Counter(y).values())
resampled_count = list(Counter(y_res).values())

print(f'Initial class distribution: {init_count}.')
# Initial class distribution: [400, 75, 25].

print(f'Resampled class distribution: {resampled_count}.')
# Resampled class distribution: [400, 400, 400].
```

3.2. Machine learning pipeline

The `GeometricSMOTE` object can be used as a part of a machine learning pipeline. Listing 2 presents a pipeline composed by a G-SMOTE oversampler, a PCA tranformation and a decision tree classifier. The pipeline is trained on imbalanced binary-class data and evaluated on a hold-out set. The user applies the process in a small number of steps while the internal details of the calculations are hidden.

Listing 2: Training and evaluation of a machine learning pipeline that contains the `GeometricSMOTE` object.

```
# Import classes and functions.
from gsmote import GeometricSMOTE
from sklearn.datasets import make_classification
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from imblearn.pipeline import make_pipeline

# Generate an imbalanced binary-class dataset.
X, y = make_classification(
```

```
            random_state=23,
            n_classes=2,
            n_samples=500,
            weights=[0.8, 0.2]
)

# Split the data to training and hold-out sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state

# Create the pipeline's objects with default hyperparameters.
gsmote = GeometricSMOTE(random_state=11)
pca = PCA()
clf = DecisionTreeClassifier(random_state=3)

# Create the pipeline.
pip = make_pipeline(gsmote, pca, clf)

# Fit the pipeline to the training set.
pip.fit(X_train, y_train)

# Evaluate the pipeline on the hold-out set using the F-score.
test_score = f1_score(y_test, pip.predict(X_test))

print(f'F-score on hold-out set: {test_score}.')
# F-score on hold-out set: 0.7.
```

4. Impact and conclusions

Classification of imbalanced datasets is a challenging task for standard machine learning algorithms. G-SMOTE, as a enhancement of the SMOTE data generation mechanism, provides a flexible and effective way for resampling the imbalanced data that has been shown to outperform SMOTE and its variations. Machine learning researchers and industry practitioners can benefit from using G-SMOTE in their work since the imbalanced learning problem is an active research area as well as a common feature of real-world applications.

The `geometric-smote` project provides a Python implementation of the state-of-the-art oversampling algorithm G-SMOTE. The main advantage of this implementation is that it is built on top of Scikit-Learn's ecosystem. Therefore, using the oversampler in typical machine learning workflows is an effortless task. Also, the public API of the main class `GeometricSMOTE` is identical to the one implemented in Imbalanced-Learn for all oversamplers. Consequently, users of Imbalanced-Learn and Scikit-Learn, that apply oversampling on imbalanced data, can integrate the `gsmote` package in their existing work in a straightforward manner.

5. Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] N. V. Chawla, A. Lazarevic, L. Hall, K. Boyer, SMOTEBoost: improving prediction of the minority class in boosting, Principles of Knowledge Discovery in Databases, PKDD-2003 (2003) 107–119doi:10.1007/b13634. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.80.1499

[2] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Systems with Applications 73 (2017) 220–239. doi:10.1016/j.eswa.2016.12.035.
URL https://doi.org/10.1016/j.eswa.2016.12.035

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357. arXiv:1106.1813, doi:10.1613/jair.953.

[4] S. Barua, M. M. Islam, X. Yao, K. Murase, MWMOTE - Majority weighted minority oversampling technique for imbalanced data set learning, IEEE Transactions on Knowledge and Data Engineering 26 (2) (2014) 405–425. doi:10.1109/TKDE.2012.232.

[5] A. Fernández, V. López, M. Galar, M. J. del Jesus, F. Herrera, Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches, Knowledge-Based Systems 42 (2013) 97–110. doi:http://dx.doi.org/10.1016/j.knosys.2013.01.018.
URL http://www.sciencedirect.com/science/article/pii/S0950705113000300

[6] H. He, E. A. Garcia, Learning from Imbalanced Data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284. arXiv:arXiv:1011.1669v3, doi:10.1109/TKDE.2008.239.

[7] G. Lemaitre, F. Nogueira, C. K. Aridas, Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning, Journal of Machine Learning Research 18 (2016) 1–5. arXiv:1609.06570, doi:http://www.jmlr.org/papers/volume18/16-365/16-365.pdf. URL http://arxiv.org/abs/1609.06570

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Vol. 12, 2011. arXiv:arXiv:1201.0490v2, doi:10.1007/s13398-014-0173-7.2. URL http://dl.acm.org/citation.cfm?id=2078195

[9] G. Douzas, F. Bacao, Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE, Information Sciences 501 (2019) 118–135. doi:10.1016/J.INS.2019.06.007. URL https://www.sciencedirect.com/science/article/pii/S0020025519305353?via{%}3Dihub