

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

Georgios Douzas, Fernando Bacao*

NOVA Information Management School, Universidade Nova de Lisboa

Abstract

Learning from class-imbalanced data continues to be a frequent and challenging problem in machine learning. Standard classification algorithms are designed under the assumption that the distribution of classes is balanced. To mitigate this problem several approaches have been proposed. The most general and popular approach is the generation of artificial data for the minority classes, known as oversampling. Geometric SMOTE is a state-of-the-art oversampling algorithm that has been shown to outperform other standard oversamplers in a large number of datasets. In order to make available Geometric SMOTE to the machine learning community, in this paper we provide a Python implementation. It is important to note that this implementation integrates seamlessly with the Scikit-Learn ecosystem. Therefore, machine learning researchers and practitioners can benefit from its use in a straightforward manner.

Keywords: Machine learning, Classification, Imbalanced learning, 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/georgedouzas/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 and non-trivial problem for academic researchers and industry practitioners alike. 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].

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. An important characteristic of imbalanced data is the Imbalance Ratio (IR) which is defined as the ratio between the number of samples of the majority class and each of the minority classes. For example, in a fraud detection task with 1% of fraudulent transactions, corresponding to an $IR = \frac{0.99}{0.01} = 99$, a trivial classifier that always labels a transaction as legit will score a classification accuracy of 99%. However in this case, all fraud cases remain undetected. IR values between 100 and 100.000 have been observed [3], [4]. Figure 1 shows an example of imbalanced data in two dimensions and the

23 resulting decision boundary of a typical classifier when they are used as a
 24 training set.

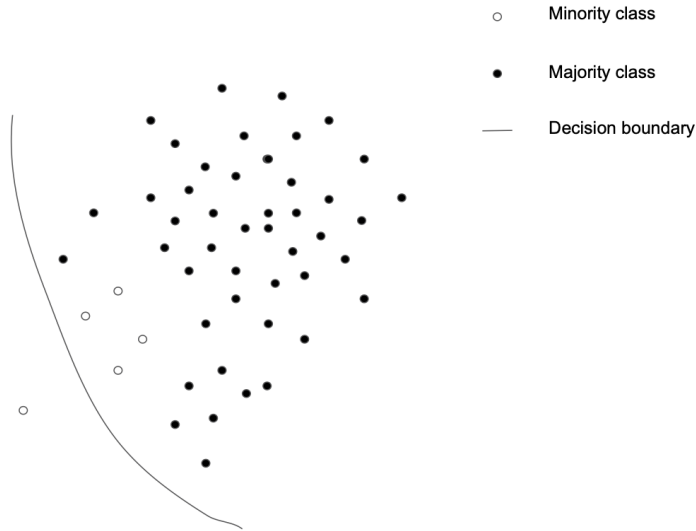


Figure 1: Imbalanced data in two dimensions. The decision boundary of a classifier shows a bias towards the majority class.

25 1.2. *Oversampling algorithms*

26 Various approaches have been proposed to deal with the imbalanced learn-
 27 ing problem. The most general approach is the modification at the data level
 28 by oversampling the minority class(es) [5]. Synthetic Minority Oversampling
 29 Technique (SMOTE) was the first informed oversampling algorithm proposed
 30 and continuous to be extensively used [3]. It generates synthetic instances
 31 along a line segment that joins minority class samples. Although SMOTE
 32 has been shown to be effective for generating artificial data, it also has some
 33 weaknesses [6]. In order to improve the quality of the generated data, many
 34 variants of SMOTE have been proposed. Nevertheless, all of these variations
 35 use the same data generation mechanism, i.e. linear interpolation between
 36 minority class samples as shown in figure 2.

37 A Python implementation of SMOTE and several of its variants is avail-
 38 able in the Imbalanced-Learn [7] library, which is fully compatible with the
 39 popular machine learning toolbox Scikit-Learn [8].

40 1.3. *Geometric SMOTE*

41 Geometric SMOTE (G-SMOTE) [9] uses a different approach compared
 42 to existing SMOTE’s variations. More specifically, G-SMOTE oversampling

43 algorithm substitutes the data generation mechanism of SMOTE by defin-
 44 ing a flexible geometric region around each minority class instance and gen-
 45 erating synthetic instances inside the boundaries of this region. The al-
 46 gorithm requires the selection of the hyperparameters `truncation_factor`,
 47 `deformation_factor`, `selection_strategy` and `k_neighbors`. The first three
 48 of them, called geometric hyperparameters, control the shape of the geomet-
 49 ric region while the later adjusts its size. Figure 2 presents a visual compar-
 50 ison between the data generation mechanisms of SMOTE and G-SMOTE.

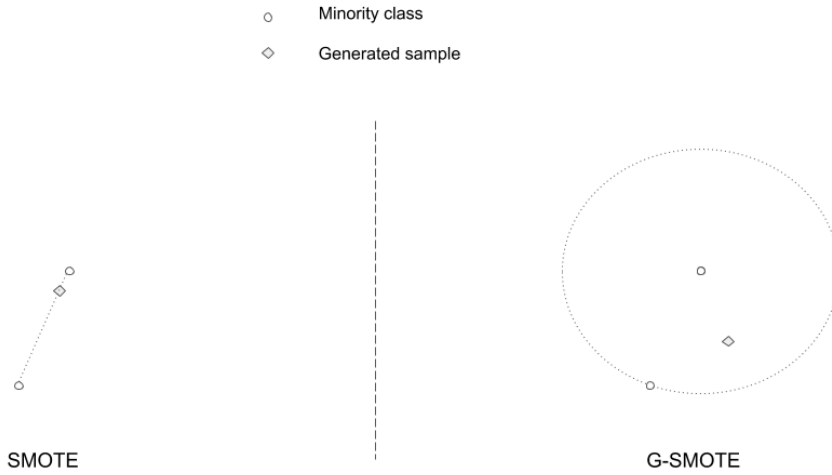


Figure 2: Comparison between the data generation mechanisms of SMOTE and G-SMOTE. SMOTE uses linear interpolation, while G-SMOTE defines a circle as the permissible data generation area.

51 G-SMOTE algorithm has been shown to outperform SMOTE and its
 52 variants across 69 imbalanced datasets for various classifiers and evaluation
 53 metrics [9]. In this paper, we present a Python implementation of G-SMOTE.
 54 In section 2, the software description is given while section 3 provides a
 55 demonstrative example of its functionalities.

56 2. Software description

57 The `geometric-smote` software project is written in Python 3.7. It con-
 58 tains an object-oriented implementation of the G-SMOTE algorithm as well
 59 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.py`. It contains the class `GeometricSMOTE` that implements the G-SMOTE algorithm. The initialization of a `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.

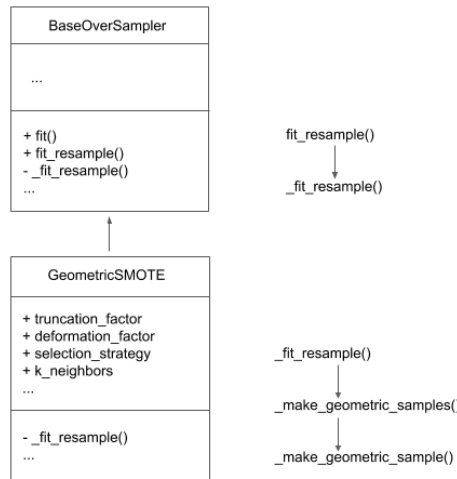


Figure 3: UML class diagrams and callgraphs of main classes and methods.

81 2.2. Software Functionalities

82 As it was mentioned in subsection 2.1, the class `GeometricSMOTE` repre-
83 sents the G-SMOTE oversampler. The initializer of `GeometricSMOTE` includes
84 the following G-SMOTE’s hyperparameters: `truncation_factor`, `deformation_factor`,
85 `selection_strategy` and `k_neighbors` as explained in subsection 1.3. Once
86 the `GeometricSMOTE` object is initialized with a specific parametrization, it
87 can be used to resample the imbalanced data represented by the input ma-
88 trix `X` and the target labels `y`. Following the Scikit-Learn API, both `X`, `y` are
89 array-like objects of appropriate shape.

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

95 The `geometric-smote` project has been designed to integrate with the
96 Imbalanced-Learn toolbox and Scikit-Learn ecosystem. Therefore the `GeometricSMOTE`
97 object can be used in a machine learning pipeline, through Imbalanced-
98 Learn’s class `Pipeline`, that automatically combines `samplers`, `transformers`
99 and `estimators`. The next section provides examples of the above function-
100 alities.

101 3. Illustrative Examples

102 3.1. Basic example

103 An example of resampling multi-class imbalanced data using the `fit_resample`
104 method is presented in Listing 1. Initially, a 3-class imbalanced dataset is
105 generated. Next, `GeometricSMOTE` object is initialized with default values for
106 the hyperparameters, i.e. `truncation_factor = 1.0`, `deformation_factor =`
107 `0.0`, `selection_strategy = combined`. Finally, the object’s `fit_resample`
108 method is used to resample the data. Printing the class distribution before
109 and after resampling confirms that the resampled data `X_res`, `y_res` are per-
110 fectly balanced. `X_res`, `y_res` can be used as training data for any classifier
111 in the place of `X`, `y`.

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

```
112 # Import classes and functions.  
113 from collections import Counter  
114 from gsmote import GeometricSMOTE  
115 from sklearn.datasets import make_classification  
116  
117 # Generate an imbalanced 3-class dataset.
```

```

118 X, y = make_classification(
119     random_state=23,
120     n_classes=3,
121     n_informative=5,
122     n_samples=500,
123     weights=[0.8, 0.15, 0.05]
124 )
125
126 # Create a GeometricSMOTE object with default hyperparameters.
127 gsmote = GeometricSMOTE(random_state=10)
128
129 # Resample the imbalanced dataset.
130 X_res, y_res = gsmote.fit_resample(X, y)
131
132 # Print number of samples per class for initial and resampled data.
133 init_count = list(Counter(y).values())
134 resampled_count = list(Counter(y_res).values())
135
136 print(f'Initial_class_distribution: {init_count}.')
137 # Initial class distribution: [400, 75, 25].
138
139 print(f'Resampled_class_distribution: {resampled_count}.')
140 # Resampled class distribution: [400, 400, 400].

```

141 3.2. Machine learning pipeline

142 As mentioned before, the **GeometricSMOTE** object can be used as a part
143 of a machine learning pipeline. Listing 2 presents a pipeline composed by a
144 G-SMOTE oversampler, a PCA tranformation and a decision tree classifier.
145 The pipeline is trained on imbalanced binary-class data and evaluated on a
146 hold-out set. The user applies the process in a simple way while the internal
147 details of the calculations are hidden.

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

```

148 # Import classes and functions.
149 from gsmote import GeometricSMOTE
150 from sklearn.datasets import make_classification
151 from sklearn.decomposition import PCA
152 from sklearn.tree import DecisionTreeClassifier
153 from sklearn.model_selection import train_test_split
154 from sklearn.metrics import f1_score

```

```

155 from imblearn.pipeline import make_pipeline
156
157 # Generate an imbalanced binary-class dataset.
158 X, y = make_classification(
159     random_state=23,
160     n_classes=2,
161     n_samples=500,
162     weights=[0.8, 0.2]
163 )
164
165 # Split the data to training and hold-out sets.
166 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=
167
168 # Create the pipeline's objects with default hyperparameters.
169 gsmote = GeometricSMOTE(random_state=11)
170 pca = PCA()
171 clf = DecisionTreeClassifier(random_state=3)
172
173 # Create the pipeline.
174 pip = make_pipeline(gsmote, pca, clf)
175
176 # Fit the pipeline to the training set.
177 pip.fit(X_train, y_train)
178
179 # Evaluate the pipeline on the hold-out set using the F-score.
180 test_score = f1_score(y_test, pip.predict(X_test))
181
182 print(f'F-score on hold-out set: {test_score}.')
183 # F-score on hold-out set: 0.7.

```

184 4. Impact and conclusions

185 Classification of imbalanced datasets is a challenging task for standard
186 machine learning algorithms. G-SMOTE, as an enhancement of the SMOTE
187 data generation mechanism, provides a flexible and effective way for resam-
188 pling the imbalanced data. G-SMOTE's empirical results prove that it out-
189 performs SMOTE and its variants. Machine learning researchers and indus-
190 try practitioners can benefit from using G-SMOTE in their work since the
191 imbalanced learning problem is a common characteristic of many real-world
192 applications.

193 The `geometric-smote` project provides the only Python implementation,
 194 to the best of our knowledge, of the state-of-the-art oversampling algorithm
 195 G-SMOTE. A significant advantage of this implementation is that it is built
 196 on top of the Scikit-Learn’s ecosystem. Therefore, using the G-SMOTE
 197 oversampler in typical machine learning workflows is an effortless task for the
 198 user. Also, the public API of the main class `GeometricSMOTE` is identical to
 199 the one implemented in Imbalanced-Learn for all oversamplers. This means
 200 that users of Imbalanced-Learn and Scikit-Learn, that apply oversampling
 201 on imbalanced data, can integrate the `gsmote` package in their existing work
 202 in a straightforward manner or even replace directly any Imbalanced-Learn’s
 203 oversampler with `GeometricSMOTE`.

204 5. Conflict of Interest

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

208 References

- 209 [1] N. V. Chawla, A. Lazarevic, L. Hall, K. Boyer, SMOTEBoost: improving
 210 prediction of the minority class in boosting, Principles of Knowledge Dis-
 211 covery in Databases, PKDD-2003 (2003) 107–119doi:10.1007/b13634.
 212 URL [http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.80.1499)
 213 [1.80.1499](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.80.1499)
- 214 [2] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing,
 215 Learning from class-imbalanced data: Review of methods and appli-
 216 cations, Expert Systems with Applications 73 (2017) 220–239. doi:
 217 10.1016/j.eswa.2016.12.035.
 218 URL <https://doi.org/10.1016/j.eswa.2016.12.035>
- 219 [3] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE:
 220 Synthetic minority over-sampling technique, Journal of Artificial Intel-
 221 ligence Research 16 (2002) 321–357. arXiv:1106.1813, doi:10.1613/
 222 jair.953.
- 223 [4] S. Barua, M. M. Islam, X. Yao, K. Murase, MWMOTE - Majority
 224 weighted minority oversampling technique for imbalanced data set learn-
 225 ing, IEEE Transactions on Knowledge and Data Engineering 26 (2) (2014)
 226 405–425. doi:10.1109/TKDE.2012.232.

- 227 [5] A. Fernández, V. López, M. Galar, M. J. del Jesus, F. Herrera, Analysing
228 the classification of imbalanced data-sets with multiple classes: Bina-
229 rization techniques and ad-hoc approaches, Knowledge-Based Systems
230 42 (2013) 97–110. doi:[http://dx.doi.org/10.1016/j.knosys.2013.](http://dx.doi.org/10.1016/j.knosys.2013.01.018)
231 01.018.
232 URL [http://www.sciencedirect.com/science/article/pii/](http://www.sciencedirect.com/science/article/pii/S0950705113000300)
233 S0950705113000300
- 234 [6] H. He, E. A. Garcia, Learning from Imbalanced Data, IEEE Transactions
235 on Knowledge and Data Engineering 21 (9) (2009) 1263–1284. arXiv:
236 arXiv:1011.1669v3, doi:10.1109/TKDE.2008.239.
- 237 [7] G. Lemaitre, F. Nogueira, C. K. Aridas, Imbalanced-learn: A Python
238 Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning,
239 Journal of Machine Learning Research 18 (2016) 1–5. arXiv:1609.06570,
240 doi:<http://www.jmlr.org/papers/volume18/16-365/16-365.pdf>.
241 URL <http://arxiv.org/abs/1609.06570>
- 242 [8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion,
243 O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander-
244 plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay,
245 Scikit-learn: Machine learning in Python, Vol. 12, 2011. arXiv:arXiv:
246 1201.0490v2, doi:10.1007/s13398-014-0173-7.2.
247 URL <http://dl.acm.org/citation.cfm?id=2078195>
- 248 [9] G. Douzas, F. Bacao, Geometric SMOTE a geometrically enhanced
249 drop-in replacement for SMOTE, Information Sciences 501 (2019)
250 118–135. doi:10.1016/J.INS.2019.06.007.
251 URL [https://www.sciencedirect.com/science/article/pii/](https://www.sciencedirect.com/science/article/pii/S0020025519305353?via=I3Dihub)
252 S0020025519305353?via=I3Dihub