SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data

María José Basgall[1,2], Waldo Hasperué[2], Marcelo Naiouf[2], Alberto Fernández[3], and Francisco Herrera[3]

[1] *UNLP, CONICET, III-LIDI, La Plata, Argentina*
[2] *Instituto de Investigación en Informática (III-LIDI), Facultad de Informática - Universidad Nacional de La Plata*
[3] *DaSCI Andalusian Institute of Data Science and Computational Intelligence, University of Granada, Granada, 18071, Spain*
{mjbasgall, whasperue, mnaiouf}@lidi.info.unlp.edu.ar
{alberto, herrera}@decsai.ugr.es

Abstract

The volume of data in today's applications has meant a change in the way Machine Learning issues are addressed. Indeed, the Big Data scenario involves scalability constraints that can only be achieved through intelligent model design and the use of distributed technologies. In this context, solutions based on the Spark platform have established themselves as a de facto standard.

In this contribution, we focus on a very important framework within Big Data Analytics, namely classification with imbalanced datasets. The main characteristic of this problem is that one of the classes is underrepresented, and therefore it is usually more complex to find a model that identifies it correctly. For this reason, it is common to apply preprocessing techniques such as oversampling to balance the distribution of examples in classes.

In this work we present SMOTE-BD, fully scalable preprocessing approach for imbalanced classification in Big Data. It is based on one of the most widespread preprocessing solutions for imbalanced classification, namely the SMOTE algorithm, which creates new synthetic instances according to the neighborhood of each example of the minority class. Our novel development is made to be independent of the number of partitions or processes created to achieve a higher degree of efficiency. Experiments conducted on different standard and Big Data datasets show the quality of the proposed design and implementation.

Keywords: Big Data, Imbalanced classification, Preprocessing, SMOTE, Spark

1 Introduction

In Machine Learning, imbalanced data classification occurs when the classes in a problem show a skewed distribution [1, 2]. Canonical classifiers are designed to optimize overall accuracy not taking into account the relative class distribution. Hence, these classifiers tend to ignore small classes while concentrating on classifying the large ones accurately. This topic is very significant due to the large amount of real applications, where the minority class represents the key concept (e.g., many biological problems) [3].

A strategy to deal with imbalanced datasets consists of applying a preprocessing step to resampling the training data. To do so, the most popular technique is known as SMOTE ("Synthetic Minority Oversampling TEchnique") [4, 5], which forms new minority class examples by interpolating between several neighbour minority class examples.

Preprocessing methods were initially designed for standard size datasets [6]. As such, they cannot be applied directly when it comes to Big Data, due to scalability issues [7]. To overcome this, those techniques must be adapted and/or reimplemented to be executed in a distributed way, using frameworks such as Apache Spark [8, 9].

However, the translation from the original preprocessing method towards a distributed approach is not straightforward. First, the programming framework may imply a complete redesign of the procedure to be adapted to a divide-and-conquer strategy, e.g., MapReduce. Second, the data division required to address Big Data problems may lead to lack of data when processing local models. For these reasons, there is still few research on the topic [10].

These facts imply the need to develop a new research line on the generation of minority instances for Big Data. It is important to point out that current technologies to work with Big Data present two different approaches with respect to how partial models from the Map stage are aggregated [11]. On the one hand, there are "local" strategies which produce an approximate model by applying a direct aggregation on partial models. On the other hand, there are "global" methods, which distribute both data and models to iteratively build a final result. It is straightforward to notice that the first type of approaches are preferred when seeking for an exact solution.

In this work, we propose SMOTE-BD, an exact solution for the implementation of SMOTE in Big Data. To do so, our methodology focuses on the calculation of the neighborhood based on a recent k nearest neighbors (kNN) approach [12]. Additionally, it incorporates a thorough design based on the use of scalable data structures and functions. Specifically, it is implemented under Scala for the Spark framework [9].

A brief experimental study is carried out in order to show the quality of the proposed oversampling implementation. First, a comparison between SMOTE-BD and the standard sequential methodology is shown for small datasets. Then, we contrast the algorithm's performance and scalability in the scenario of Big datasets using different numbers of partitions.

The remainder of this paper is organized as follows. Section 2 introduces the current state of the art in imbalanced Big Data classification. Section 3 details the proposed model of a fully scalable SMOTE in Spark framework. Then, in Section 4 the experimental study carried out. Finally, in Section 5 the conclusions and future works are described.

2 Imbalanced Big Data Classification

The problem of imbalanced classification appeared at the same time that researchers realized that traditional classification algorithms could not model well the underrepresented classes [1, 2]. When it comes to the Big Data context, this problem is accentuated.

In [10], authors presented an exhaustive study with the objective of evaluating the performance of the traditional solutions for class imbalanced in the Big Data context. To this end, several preprocessing techniques were adapted and embedded in a MapReduce workflow. Specifically, in this research the random oversampling (ROS-BigData), random undersampling (RUS-BigData) and SMOTE MapReduce version for Hadoop (SMOTE-H) were employed.

Following the Hadoop MapReduce philosophy, each Map process was responsible for adjusting the class distribution for its data partition, either by random replication of minority class instances (ROS-BigData), random elimination of majority class instances (RUS-BigData) or the generation of synthetic data carried out by SMOTE technique (SMOTE-H). Subsequently, a single Reduce process was responsible for collecting the results generated by each mapper and assigning them randomly to form the balanced dataset.

All those preprocessing methods worked locally within each Map, thus limiting the potential of these algorithms. As remarks of the aforementioned work, they observed random oversampling to be more robust than the other techniques when the number of data partitions increased.

The MapReduce solutions, which are based on the "divide-and-conquer" approach implies a partitioning of the data that can lead to two serious consequences [1]. One of them is the extreme lack of positive data in each partition data, which represents a partial representation of the dataset information, particularly with regard to the real neighborhood of the instances. The other consequence of the partitioning, and very related to the previous one, is the presence of very local data with low density called "small-disjuncts" which represents the concepts of interest. Creating minority instances from small-disjuncts can lead to noise or an over-generalization, and they would enter the "safe" zones of the majority class.

Figure 1 depicts the lack of data situation for the training data of the *yeast5* imbalanced problem from KEEL dataset repository [13]. It can be noticed the low concentration of minority instances that they can be considered as noise or rare data.

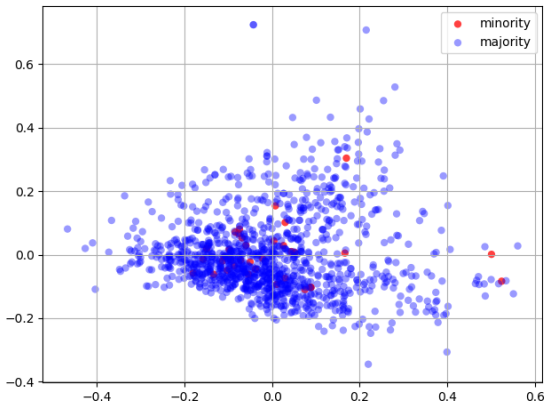

Figure 1: Lack of density on the *yeast5* dataset

3 SMOTE-BD: An exact oversampling solution in Spark

In this section, an exact SMOTE fully scalable methodology in Spark for Big Data is presented.

Figure 2 depicts the general scheme of this proposal. First, the algorithm performs a filtering over the training set (stored in the HDFS) to get the minority and majority subsets of instances. Then, the minority data, which is partitioned according to an algorithm parameter, is normalized taking into account the statistics of the full training set and is cached to be reused in the following steps.

Later, nearest neighbors for each positive instance is obtained using an exact implementation of kNN in Spark (kNN-IS) [12] which splits the training dataset in a user-defined number of partitions, calculates for each instances in a chunk its neighbors and finally, in a reduction phase, makes a final list of k nearest neighbors.

After that, the generation of artificial minority class instances is begun. All the nearest neighbors obtained in the previous step are broadcasted to the main memory of the all nodes in the cluster. The *broadcast*

operation allows to keep a read-only variable cached on each node rather than shipping a copy to each task, and it performs this action in an efficient manner.

Then, for each positive instance in a data partition and using the broadcasted variable, the algorithm generates the corresponding number of synthetic examples by interpolating between each minority instance and its k nearest neighbors. Figure 3 depict how to create synthetic data points in the SMOTE algorithm.

Finally, the algorithm performs a denormalization process over the artificial dataset and joins the original positive and negative instances with the artificial ones in order to conform the balanced dataset, and saves it in the HDFS.

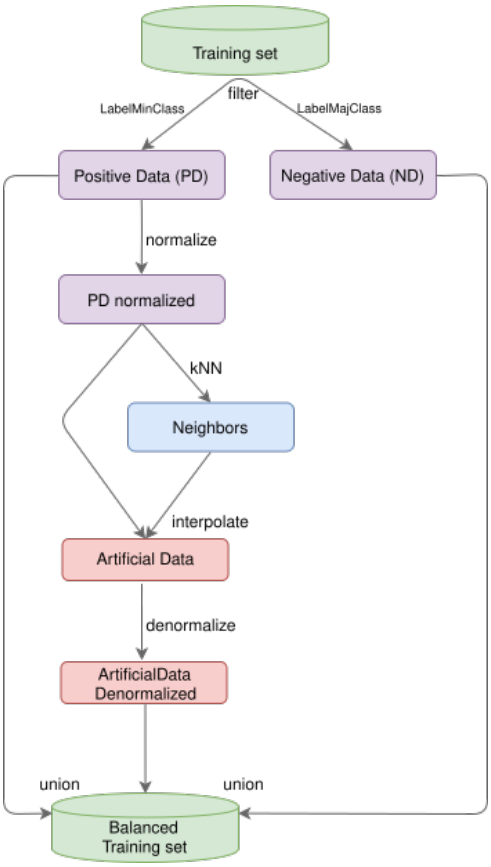

Figure 2: SMOTE-BD flowchart

Algorithms 1 and 2 show a pseudocode of the sequence of actions described above . The former covers the main program and the latter the function to create each artificial instance. This function invokes another function called *interpolation* which is in charge of doing the interpolation between two points. There is no pseudocode of this due to its simplicity.

4 Experimental Study: Analysis of the behavior of SMOTE-BD

In this section, the performance achieved by classification algorithms in synergy with SMOTE-BD implementation is presented. In order to compare the

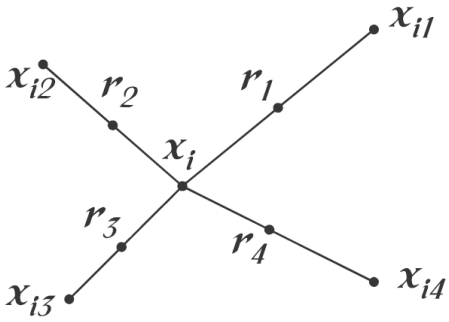

Figure 3: Interpolation between a minority instance and its k nearest neighbors.

performance of SMOTE, eight imbalanced datasets were selected and divided into two categories based on the number of examples which each dataset contains. Table 1 shows the datasets summary, where the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (majority and minority), number of instances for each class, class distribution and imbalance ratio (IR) are included.

The behavior of the resultant preprocessed datasets was tested using the Decision Trees classifier (DT), implemented in the Spark's MLlib library [14]. Table 2 shows the parameters used for the methods according to their authors' specification.

Regarding the infraestructure used to perform the experiments, the Hadoop cluster at University of Granada was used, which consists of fourteen nodes connected via a Gigabit Ethernet network. Each node has a Intel Core i7-4930K microprocessor at 3.40GHz, 6 cores (12 threads) and 64 GB of main memory working under Linux CentOS 6.9. The cluster works with Hadoop 2.6.0 (Cloudera CDH5.8.0), where the head node is configured as NameNode and ResourceManager, and the rest are DataNodes and NodeManagers. Moreover, the cluster is configured with Spark 2.2.0.

The quality measures of classification are built from a confusion matrix (shown in Table 3), which organizes the samples of each class according to their correct or incorrect identification. From this matrix four metrics that describe both classes independently are obtained:

True Positive Rate $TPR = \frac{TP}{TP+FN}$ is the percentage of positive instances correctly classified.

True Negative Rate $TNR = \frac{TN}{FP+TN}$ is the percentage of negative instances correctly classified.

False Positive Rate $FPR = \frac{FP}{FP+TN}$ is the percentage of negative instances misclassified.

False Negative Rate $FNR = \frac{FN}{TP+FN}$ is the percentage of positive instances misclassified.

None of these rates alone are adequate independently, therefore more robust metrics exist to evaluate

Algorithm 1 SMOTE-BD algorithm

Require: Tr, Ts, ratio, k, nP, nR, nIt, minClassLabel

$origData \leftarrow textFile(Tr)$
$minData \leftarrow origData.filter(labels == minClassLabel)$
$minData \leftarrow minData.map(normalize).repartition(nP)$
$neighbors \leftarrow kNN_IS.setup(Tr,Tr,k,nR,nI).calculatekNeighbours()$
$crFactor \leftarrow (nMaj - nMin)/nMin$
$neighbors \leftarrow broadcast(neighbors)$
$balancedData = null$
$synData = null$
for $i < nIt$ **do**
    $synTmp \leftarrow minData.mapPartitionsWithIndex(createSynthData(index, partData, neighbors, crFactor, k))$
    if $synData == null$ **then**
        $synData \leftarrow synTmp$
    else
        $synData \leftarrow synData.union(synTmp)$
    end if
end for
$synData \leftarrow synData.map(denormalize)$
$balancedData \leftarrow synData.union(origData)$

Algorithm 2 Function to create synthetic instances between the minority class examples and their neighbors

procedure CREATESYNTHDATA($index, partData, neighbors, crFactor, k$)
    $artificialData = null$
    for $firstInstance \leftarrow partitionData; nc = 0$ *to* $crFactor$ **do**
        $selNeighbor \leftarrow newRandom().nextInt(k)$
        $secondInstance \leftarrow neighbors(selNeighbor)$
        $newIntance \leftarrow interpolation(firstInstance, secondInstance)$
        $artificialData.add(newInstance)$
    end for
    return $artificialData$
end procedure

Table 1: Datasets summary

Big datasets	#Ex.	#Atts.	Class (maj;min)	#Class(maj; min)	%Class(maj; min)	IR
covtype7	464677	54	(negative; positive)	(448421; 16256)	(96.5; 3.5)	27.58
poker0_vs_2	450022	10	(negative; positive)	(410960; 39062)	(91.32; 8.68)	10.52
poker0_vs_5	412600	10	(negative; positive)	(410960; 1640)	(99.60; 0.4)	250.58
susy_ir4	2712175	18	(negative; positive)	(2169740; 542435)	(80; 20)	4
Small datasets	**#Ex.**	**#Atts.**	**Class (maj;min)**	**#Class(maj; min)**	**%Class(maj; min)**	**IR**
page-blocks0	5472	10	(negative; positive)	(4913; 559)	(89.78; 10.22)	8.78
segment0	2308	19	(negative; positive)	(1979; 329)	(85.75; 14.25)	6.02
shuttle-c0-vs-c4	1829	9	(negative; positive)	(1706; 123)	(93.28; 6.72)	13.88
yeast5	1484	8	(negative; positive)	(1440; 44)	(97.04; 2.96)	32.78

performance in classification scenarios. One of the most widely used metric in imbalanced classification is called Geometric Mean (GM) [15] which is defined in equation 1. The GM attempts to maximize the accuracy of each one of the two classes at the same time.

$$GM = \sqrt{TPR * TNR} \quad (1)$$

The results to apply SMOTE sequential implementation (available in KEEL Software Tool [13]) and SMOTE-BD on small datasets are shown in Table 4 and depicted in Figure 4. For SMOTE-BD implementation, tests with different number of data partitions (1 to 4) have been performed.

Table 5 shows the average GM results in training and testing sets for the Spark SMOTE implementation using 1, 8 and 32 partitions over the four Big Data cases of study.

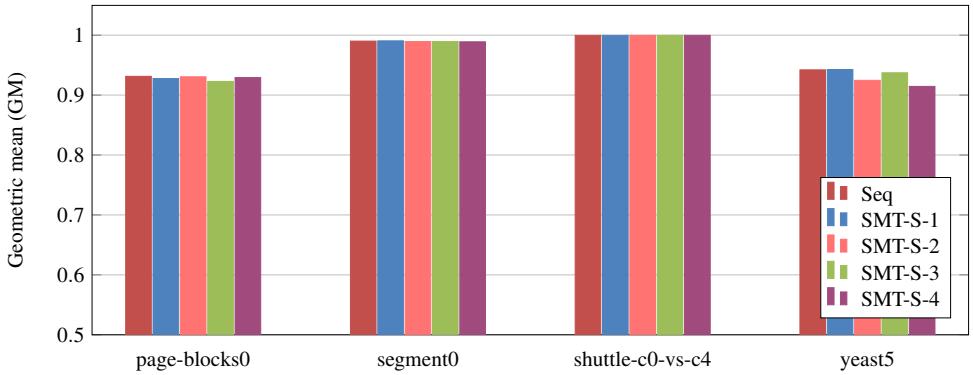

Figure 4: Average results of the sequential SMOTE and the SMOTE-BD version, combined with the Spark version of Decision Trees over the small datasets using the GM measure.

Table 2: Algorithms and parameters

Algorithms	**Parameters**
SMOTE Seq	k Nearest Neighbors: 5 Distance function: Euclidean Desired IR: 1
SMOTE-BigData	k Nearest Neighbors: 5 Distance function: Euclidean Desired IR: 1 Number of partitions: 1 Number of reducers: 1

Table 3: Confusion matrix for performance evaluation of a binary classification problem

Actual	Predicted	
	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

5 Conclusions and future works

In this work, we have developed SMOTE-BD, a fully scalable oversampling technique for imbalanced classification in Big Data Analytics. Two main reasons have motivated the design of this new methodology. On the one hand, the lack of current solutions for such a significant area of study. On the other hand, to consider a global procedure that takes into account the whole neighborhood of each minority class instance.

The advantages of this novel approach are clear. The most significant one is consolidating a cluster of new synthetic instances, thus avoiding the overgeneralization due to data locality for traditional MapReduce procedures. Furthermore, the number of data partitions considered to add more efficiency to the process is independent of the algorithm's procedure, so that there are no constraints regarding the lack of data.

The novelty of this area of research implies many topics for future study. Among them, we may stress three important points. First, to consider novel data structures to avoid the limitation of "broadcast" variables for very large dataset sizes. Second, to analyze the quality of the new preprocessed data regarding different parametrization, especially the percentage of oversampling. Finally, to study new scalable designs of SMOTE-based algorithms in Big Data, in particular considering the data redundancy that is present in current datasets.

References

[1] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, no. 20, pp. 113–141, 2013.

[2] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[3] D. Galpert, A. Fernández, F. Herrera, A. Antunes, R. Molina-Ruiz, and G. AgÃŒero-Chapin, "Surveying alignment-free features for ortholog detection in related yeast proteomes by using supervised big data classifiers," *BMC Bioinformatics*, vol. 19, no. 1, 2018.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over–sampling technique," *Journal of Artificial Intelligent Research*, vol. 16, pp. 321–357, 2002.

[5] A. Fernandez, S. Garcia, F. Herrera, and N. Chawla, "Smote for learning from imbalanced data: Progress and challenges. marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

Table 4: Average results of the sequential SMOTE and the SMOTE-BD version, combined with the Spark version of Decision Trees over the small datasets using the GM measure.

Datasets	Seq	SMT-1p	SMT-2p	SMT-3p	SMT-4p
page-blocks0	0.9313	0.9276	0.9305	0.9228	0.9292
segment0	0.9901	0.9905	0.9893	0.9893	0.9889
shuttle-c0-vs-c4	0.9997	0.9997	0.9997	0.9997	0.9997
yeast5	0.9421	0.9427	0.9246	0.9372	0.9145

Table 5: Average results of the SMOTE-BD algorithm for 1, 8 and 32 partitions, combined with the Spark version of Decision Trees over the big datasets using the GM measure.

Datasets	SMT-1p	SMT-8p	SMT-32p
covtype7	0.9227	0.9287	0.9181
poker0_vs_2	0.4582	0.4585	0.4713
poker0_vs_5	0.4162	0.4208	0.4206
susy_ir4	0.7615	0.7601	0.7585

[6] R. C. Prati, G. E. A. P. A. Batista, and D. F. Silva, "Class imbalance revisited: a new experimental setup to assess the performance of treatment methods," *Knowledge and Information Systems*, vol. 45, no. 1, pp. 247–270, 2015.

[7] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[8] A. Fernández, S. Río, V. López, A. Bawakid, M. J. del Jesus, J. Benítez, and F. Herrera, "Big Data with cloud computing: An insight on the computing environment, MapReduce and programming framework," *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380–409, 2014.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *HotCloud 2010*, pp. 1–7, 2010.

[10] A. Fernandez, S. del Rio, N. V. Chawla, and F. Herrera, "An insight into imbalanced big data classification: Outcomes and challenges," *Complex and Intelligent Systems*, vol. 3, no. 2, pp. 105–120, 2017.

[11] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, and F. Herrera, "Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce," *Information Fusion*, vol. 42, pp. 51–61, 2018.

[12] J. Maillo, S. Ramírez-Gallego, I. Triguero, and F. Herrera, "knn–is: An iterative spark-based design of the k-nearest neighbors classifier for big data.," *Knowledge-Based Systems*, vol. 117, pp. 3–15, 2017.

[13] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data–mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multi–Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "MLlib: Machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1–7, 2016.

[15] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems.," *Pattern Recognition*, vol. 36, no. 3, pp. 849–851, 2003.