Complex Intell. Syst. (2017) 3:105–120
DOI 10.1007/s40747-017-0037-9

REVIEW ARTICLE

An insight into imbalanced Big Data classification: outcomes and challenges

Alberto Fernández[1] · **Sara del Río**[1] · **Nitesh V. Chawla**[2,3] · **Francisco Herrera**[1]

Received: 10 November 2016 / Accepted: 10 February 2017 / Published online: 1 March 2017

Abstract Big Data applications are emerging during the last years, and researchers from many disciplines are aware of the high advantages related to the knowledge extraction from this type of problem. However, traditional learning approaches cannot be directly applied due to scalability issues. To overcome this issue, the MapReduce framework has arisen as a "de facto" solution. Basically, it carries out a "divide-and-conquer" distributed procedure in a fault-tolerant way to adapt for commodity hardware. Being still a recent discipline, few research has been conducted on imbalanced classification for Big Data. The reasons behind this are mainly the difficulties in adapting standard techniques to the MapReduce programming style. Additionally, inner problems of imbalanced data, namely lack of data and small disjuncts, are accentuated during the data partitioning to fit the MapReduce programming style. This paper is designed under three main pillars. First, to present the first outcomes for imbalanced classification in Big Data problems, introducing the current research state of this area. Second, to analyze the behavior of standard pre-processing techniques in this particular framework. Finally, taking into account the experimental results obtained throughout this work, we will carry out a discussion on the challenges and future directions for the topic.

Keywords Big Data · Imbalanced classification · MapReduce · Pre-processing · Sampling

✉ Alberto Fernández
alberto@decsai.ugr.es

Sara del Río
srio@decsai.ugr.es

Nitesh V. Chawla
nchawla@nd.edu

Francisco Herrera
herrera@decsai.ugr.es

[1] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain

[2] Department of Computer Science and Engineering, 384 Fitzpatrick Hall, University of Notre Dame, Notre Dame, IN 46556, USA

[3] Interdisciplinary Center for Network Science and Applications, 384 Nieuwland Hall of Science, University of Notre Dame, Notre Dame, IN 46556, USA

Introduction

The topic of imbalanced classification has gathered a wide attention of researchers during the last several years [1–5]. It occurs when the classes represented in a problem show a skewed distribution, i.e., there is a minority (or positive) class, and a majority (or negative) one. This case study may be due to rarity of occurrence of a given concept, or even because of some restrictions during the gathering of data for a particular class. In this sense, class imbalance is ubiquitous and prevalent in several applications such as microarray research [6], medical diagnosis [7], oil-bearing of reservoir recognition [8], or intrusion detection systems [9].

The issue of class imbalance can be further confounded by different costs in making errors, i.e., false negatives can be more costly than false positives, such as in those problems previously presented, i.e., medical diagnosis. In these cases, the problem is that a bias towards the majority class is shown during the learning process, seeking for both accuracy and generalization. A direct consequence is that minority classes cannot be well modeled, and the final performance decays.

To successfully address the task of imbalanced classification, a number of different solutions have been proposed, which mainly fall into three categories [1–4]. The first is the family of pre-processing techniques aiming to rebalance the

training data [10]. The second one is related to the algorithmic approaches that alter the learning mechanism by taking into account the different class distribution [11]. The third category comprises cost-sensitive learning approaches that consider a different cost for the misclassification of each class [12,13].

The emergence of Big Data brings new problems and challenges for the class imbalance problem. Big Data posits the challenges of Volume, Velocity, and Variety [14,15]. In addition, other attributes have been linked to Big Data such as Veracity or Value, among others [16]. The scalability issue must be properly addressed to develop new solutions or adapt existing ones for Big Data case studies [17,18]. Spark [19] has emerged as a popular choice to implement large-scale Machine Learning applications on Big Data.

In this paper, we focus on learning from imbalanced data problems in the context of Big Data, especially when faced with the challenge of Volume. We will analyze the strengths and weaknesses of various MapReduce [20]-based implementation of algorithms that address imbalanced data. We have implemented a Spark library for undersampling and oversampling based on MapReduce.[1] This experimental study will show that some of the inner characteristics of imbalanced data are accentuated in this framework. In particular, we will stress the influence of the lack of data and small disjuncts that are a result from the data partitioning in this type of process. Finally, we will enumerate several guidelines that can allow researchers to develop high-quality solutions in this area of research. Specifically, we will take into account how data resampling techniques should be designed for imbalanced Big Data classification.

In summary, the novel contributions and objectives included in this research work are:

1. To understand the inner structure of these methodologies that have been proposed to overcome the imbalanced data problem in Big Data.
2. To acknowledge the real challenges imposed by imbalanced datasets in Big Data.
3. To carry out a thorough discussion on the main issues to be addressed for future work on the topic.

To address all these objectives, this paper is organized as follows. First, "Preliminaries" section presents an introduction on classification with imbalanced datasets, and a short description for Big Data and the MapReduce framework. "Addressing imbalanced classification in Big Data problems: current state" section includes an overview on those works that address imbalanced classification for Big Data problems. "Practical study on imbalanced Big Data classification using MapReduce" section 4 presents an experimental analysis for studying the behavior of pre-processing techniques in imbalanced Big Data problems, and a discussion on the difficulties associated with this scenario. Then, we will enumerate some challenges and open problems in "Challenges for imbalanced Big Data classification" section. Finally, "Concluding remarks" section summarizes and concludes this paper.

[1] https://spark-packages.org/package/saradelrio/Imb-sampling-ROS_and_RUS.

Preliminaries

"Classification with imbalanced datasets" section includes a description for imbalanced classification, focusing on its characteristics, metrics of performance, and standard solutions. Then, "Big Data and the MapReduce framework" section presents the scenario of Big Data and the MapReduce framework.

Classification with imbalanced datasets

The task of classification in imbalanced domains is defined when the elements of a dataset are unevenly distributed among the classes [3,5]. The majority class(es), as a result, overwhelms the data mining algorithms skewing their performance towards it. Most algorithms simply compute the accuracy based on the percentage of correctly classified observations. However, in the case study of skewed distributions, results are highly deceiving since minority classes hold minimum effect on overall accuracy. As a result, the performance on the majority class can overwhelm the poor performance on the minority class. For example, if a given dataset has a class distribution of 98:2, as is fairly common in various real-world scenarios, (a common example is medical diagnosis [21,22]) then one can be 98% accurate by simply predicting all examples as majority class. But this prediction performance is misleading as the minority class (the class of interest) is missed.

Thus, we must consider the complete confusion matrix (Table 1) from which we may obtain the classification performance of both, positive and negative, classes independently:

- **True-positive rate** $\mathrm{TP}_{\mathrm{rate}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ is the percentage of positive instances correctly classified. This value is often known as sensitivity or recall.

Table 1 Confusion matrix for a two-class problem

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

- **True-negative rate** $\mathrm{TN}_{\mathrm{rate}} = \frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}$ is the percentage of negative instances correctly classified. This value is often known as specificity.
- **False-positive rate** $\mathrm{FP}_{\mathrm{rate}} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$ is the percentage of negative instances misclassified.
- **False-negative rate** $\mathrm{FN}_{\mathrm{rate}} = \frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}}$ is the percentage of positive instances misclassified.

We must point out that none of these measures alone are adequate independently, but we must derive additional robust metrics. Two of the most common metrics that seek at maximizing the joint performance of the classes are both the AUC [23] and Geometric Mean (GM) of the true rates [24]. The former demonstrates the trade-off between the benefits ($\mathrm{TP}_{\mathrm{rate}}$) and costs ($\mathrm{FP}_{\mathrm{rate}}$), whereas the latter attempts to maximize the accuracy of each one of the two classes at the same time [sensitivity or recall ($\mathrm{TP}_{\mathrm{rate}}$) and specificity ($\mathrm{TN}_{\mathrm{rate}}$)], as depicted in Eq. 1. In addition to this, there are other widely used performance metrics such as the F-measure [25], which combines the precision ($\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$) and recall.

$$\mathrm{GM} = \sqrt{\mathrm{TP}_{\mathrm{rate}} \cdot \mathrm{TN}_{\mathrm{rate}}} \tag{1}$$

Specific methods must be applied so that traditional classifiers are able to deal with the imbalance between classes. Three different methodologies are traditionally followed to cope with this problem [3,13]: data level solutions that rebalance the training set [10], algorithmic level solutions that adapt the learning stage towards the minority classes [11] and cost-sensitive solutions which consider different costs with respect to the class distribution [12]. It is also possible to combine several classifiers into ensembles [26], by modifying or adapting the combination of the algorithm itself of ensemble learning and any of the techniques described above, namely at the data level or by algorithmic approaches based on cost-sensitive learning [27].

Among these methodologies, data-level solutions are more versatile, since their use is independent of the classifier selected. Three possible schemes can be applied here: undersampling of the majority class examples, oversampling of the minority class examples, and hybrid techniques [10].

The simplest approach, random undersampling, removes instances from the majority class usually until the class distribution is completely balanced. However, this may imply ignoring significant examples from the training data. On the other hand, random oversampling makes exact copies of existing minority instances. The hitch here is that this method can increase the likelihood of overfitting [10], as it tends to strengthen all minority clusters disregard their actual contribution to the problem itself.

More sophisticated methods have been proposed based on the generation of synthetic samples. The basis of this type of procedures is the Synthetic Minority Oversampling Technique (SMOTE) oversampling method [28]. Its core idea is to form new minority class examples by interpolating between several minority class examples that lie together. On the one hand, SMOTE expands the clusters of the minority data, and thus to strengthen the borderline areas between classes. On the other hand, it can result in the over-generalization problem. To tackle this issue of over-generalization versus increasing the minority class representation, various adaptive sampling methods have been proposed in recent years [29–31].

Although these techniques are mainly designed to rebalance the training data prior to the learning stage, there are additional characteristics that can degrade the performance in this scenario. In particular, in both [3] and [32] authors emphasize several factors that, in combination with the uneven distribution, impose harder learning constraints. Among them, they stressed the overlap between classes [33,34], the presence of noise and small disjuncts and noisy data [35,36], and the lack of data for training [37,38].

Big Data and the MapReduce framework

The rapid growth and influx of data from private and public sectors have popularized the notion of "Big data [14]". The surge in Big Data has led to the development of custom models and algorithms that are able to extract significant value and insight into different areas such as medical, health care, business, and management [15,17,18].

To provide robust and scalable solutions, new research paradigms and developmental tools have been made available. These frameworks have been designed to ease both the storage necessities and the processing of Big Data problems [14].

The MapReduce execution environment [20] is the most common framework used in this scenario. Being a privative tool, its open source counterpart, known as Hadoop, has been traditionally used in academia research [39]. It has been designed to allow distributed computations in a transparent way for the programmer, also providing a fault-tolerant execution scheme. To take advantage of this scheme, any algorithm must be divided into two main stages: Map and Reduce. The first one is devoted to split the data for processing, whereas the second collects and aggregates the results.

Additionally, the MapReduce model is defined with respect to an essential data structure: the <key,value> pair. The processed data, the intermediate and final results work in terms of <key,value> pairs. To summarize its procedure, Fig. 1 illustrates a typical MapReduce program with its *Map* and *Reduce* steps. The terms $k_i : v_j$ refer to the key and value pair that are computed within each Map process. Then, values are grouped linking them to the same key, i.e., $k_i : v_j, \ldots, v_h$, and feed to the same Reduce process. Finally, values are

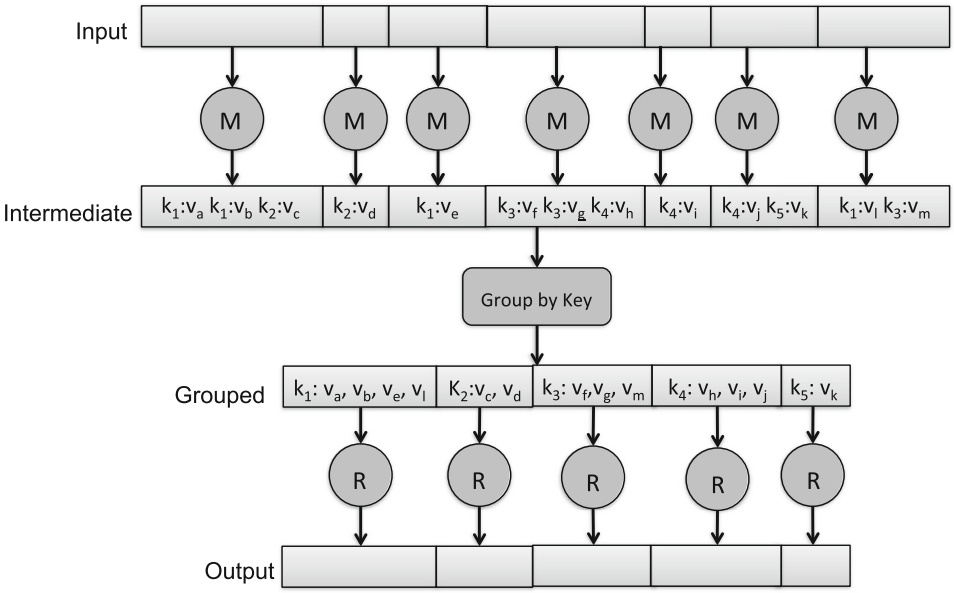


Fig. 1 The MapReduce programming model

aggregated with any function within the Reduce process to obtain the final result of the algorithm.

From this initial model, more recent alternative frameworks have arisen to provide more efficient computations for iterative process, which are the basis for any Machine Learning algorithm. Among them, Apache Spark [19,40] is clearly emerging as a more commonly embraced platform for implementing Machine Learning solutions that scale with Big Data.

Addressing imbalanced classification in Big Data problems: current state

In this section, we discuss the current state of the art on the topic of imbalanced classification for Big Data. These include initial approaches for addressing the problem, making use of those methods and solutions that were mentioned in the previous section. In Table 2 we show the list of selected proposal in a taxonomy regarding the type of methodology applied to handle the imbalanced data distribution, or whether they comprise an application paper.

In the remainder of the section we describe each one of these models. Specifically, "Data pre-processing studies" section contains the description for those techniques related to data pre-processing. "Cost-sensitive learning studies" section those approaches that carry out an algorithmic modification by means of a cost-sensitive learning. Finally, "Applications on imbalanced Big Data" section presents the application papers on the topic.

Table 2 Summary of approaches for imbalanced classification in Big Data

Type of technique	References
Data pre-processing	[41–48]
Cost-sensitive learning	[41,49–51]
Applications on imbalanced Big Data	[52–54]

Data pre-processing studies

Among the different solutions to address imbalanced classification in Big Data, data pre-processing is possibly the one that has attracted the highest attention from researchers. Therefore, we may find several approaches that aim at adapting directly the standard undersampling and oversampling techniques to the MapReduce framework. In this sense, both random undersampling, random oversampling and SMOTE are the widely used algorithms, being applied within each Map process seeking for scalability. Furthermore, some ad hoc approaches based on undersampling and oversampling have been also developed, including evolutionary undersampling, a rough set-based SMOTE, and an ensemble algorithm. Finally, we will point out a first approach for the multi-class case study.

Traditional data-based solutions for Big Data

In the first work to be discussed, authors performed a thorough study with the objective of evaluating the performance of traditional solutions for the class imbalance in the context of Big Data [41]. With this aim, several pre-processing techniques were adapted and embedded in a MapReduce workflow. Specifically, the random oversampling (ROS-BigData), random undersampling (RUS-BigData) and the SMOTE (SMOTE-BigData) MapReduce versions were pro-

posed in this research. For every technique, each Map process was responsible for adjusting the class distribution for their data partition, either by the random replication of minority class instances (ROS-BigData), the random removal of majority class instances (RUS-BigData) or the synthetic data generation carried out by SMOTE (SMOTE-BigData). Then, a unique Reduce process was responsible for collecting the outputs generated by each mapper and randomized them to form the balanced dataset. The Random Forest implementation from Mahout[2] [55,56] was selected as baseline classifier for the experiments, applied over three different imbalanced Big Data problems from the UCI dataset repository [57], with up to 6 millions of instances. One of the outcomes of their study [41] was the observation that random oversampling is more robust than the other techniques when the number of data partitions is increased. In contrast, the performance of SMOTE, random undersampling and cost-sensitive learning methods was not as high as expected.

This literature showed a preliminary study made using Hadoop, which implied a lower scalability in contrast with Spark-based approaches. Additionally, all pre-processing and classification methods worked locally within each Map, thus limiting the potential of these algorithms.

Random oversampling with evolutionary feature weighting and random forest (ROSEFW-RF)

Another work which showed the success of the application of random oversampling in the scenario of Big Data can be found in [42]. This literature described the methodology followed to achieve the first place of the ECBDL'14 Big Data challenge. This dataset consisted of an imbalance bioinformatics Big Data problem formed by 32 million instances and more than 600 attributes with just a 2% of positive instances. The algorithm, named as ROSEFW-RF, was based on several MapReduce approaches to (1) balance the classes distribution through random oversampling, (2) detect the most relevant features via an evolutionary feature weighting process and a threshold to choose them, (3) build an appropriate Random Forest model from the pre-processed data and finally (4) classify the test data.

In accordance with these issues, this work has two novel contributions with respect to [41]:

- On the one hand, authors stressed that to deal with extremely imbalanced Big Data problems such as the one described above, this implies an increment in the density of the underrepresented class using higher oversampling ratios [43].
- On the other hand, a feature selection approach was suggested to avoid the curse of dimensionality. Specifically, the authors developed a MapReduce implementation based on the evolutionary approach for Feature Weighting proposed in [58]. In this method, each map task performed a whole evolutionary feature weighting cycle in its data partition and emitted a vector of weights. Then, the Reduce process was responsible for the iterative aggregation of all the weights provided by the maps. Finally, the resulting weights were used with a threshold to select the most important characteristics.

The combination of the instance and feature pre-processing approaches was shown to achieve high-quality results in this case study. However, this proposed methodology has the constraint of applying a high ratio of oversampling, thus requiring a high training time.

Evolutionary undersampling

Regarding undersampling approaches, in [44] authors developed a parallel model to enable evolutionary undersampling methods under the MapReduce scheme. Specifically, the aforementioned model consisted of two MapReduce procedures. The first MapReduce task learns a decision tree in each map after performing evolutionary undersampling preprocessing. Then, a second MapReduce job is initiated to classify the test set. The evolutionary undersampling step is further accelerated by adding a windowing scheme adapted to the imbalanced scenario. To analyze the quality of this proposed method [44], authors carried out an experimental study with the C4.5 decision tree over different versions of the KDDCup'99 dataset, by gradually increasing its number of instances. Results shown the goodness of the global model in terms of accuracy and efficiency. An extension of this model implemented within the Spark framework has been recently presented in [59].

Data cleaning

Another research that carries out a data reduction scheme (data cleaning) can be found in [45]. Specifically, authors proposed a MapReduce-based k-Nearest Neighbor (k-NN) classifier for DNA Big Data problems. As stated previously, authors included a data reduction stage within the Map processes prior to the learning to study the best suited option, together with an analysis of the scalability.

NRSBoundary-SMOTE

In [46], authors proposed a MapReduce design of the NRSBoundary-SMOTE, an algorithm based on Neighborhood RoughSet Theory [60]. This adaptation consisted of two MapReduce procedures.

[2] http://mahout.apache.org/.

The first MapReduce task was responsible for the partition of the dataset, and the second MapReduce task carried out the oversampling of the minority class examples. More specifically, the first MapReduce job divided the training set according to neighborhood relation and it generated three subsets as output, called Positive, Minority and Boundary. The Positive subset contained the majority class samples where its neighbors have the sample class label, the Minority subset contained the minority samples, and the Boundary subset contained the minority samples that have any majority class sample in its neighbors. In the second MapReduce job, every map gets a data block of the Boundary set and it computed for each sample in its partition the k nearest neighbors. Then, the reduce process selected for each sample one of its neighbors randomly to interpolate with it. If the new synthetic sample belonged to the neighbor of samples that in Positive, another neighbor were selected from the list. Otherwise, the synthetic example was generated.

In both MapReduce processes the Positive and Minority sets were added to the Hadoop Distributed Cache [39]. This feature disables the scalability of the algorithm, as long as the training dataset must fit on the Hadoop Distributed Cache.

Extreme learning machine with resampling

A MapReduce approach based on ensemble learning and data resampling can be found in [47]. This algorithm consists of four stages: (1) alternately over-sample p times between positive class instances and negative class instances; (2) construct l balanced data subsets based on the generated positive class instances; (3) train l component classifiers with extreme learning machine algorithm on the constructed l balanced data subsets; (4) integrate the l ELM classifiers with simple voting approach.

To carry out the data pre-processing, the algorithm first calculated the center of positive class instances, and then sample instance points along the line between the center and each positive class instance, in a similar style than SMOTE [28]. Next, for each instance point in the new positive class, the method first finds its k-nearest neighbors in negative class instances with MapReduce, and then sample instance points along the line between the instance and its k-nearest negative neighbors. The process of oversampling is repeated p times. In the second stage, the algorithm sample instances l times from the negative class with the same size as the generated positive class instances. Each round of sampling, the method puts positive class and negative class instances together thus obtains l balanced data subsets.

To verify the effectiveness of this proposed method [47], authors selected seven data sets from UCI repository (with less than half million examples) and compared with three state-of-the-art approaches for classical data mining (no Big Data approaches): SMOTE-Vote, SMOTE-Boost and SMOTE-Bagging, showing better speedup and performance in terms of the g-mean metric. The drawback of this proposal is the iterative oversampling process applied in the first stage, being computationally expensive.

Multi-class imbalance

Finally, a preliminary study regarding multi-class imbalanced classification was introduced in [48]. This methodology consisted of two steps. First, they used the One-vs.-All (OVA) binarization technique [61] for decomposing original dataset into subsets of binary classes. This process was carried out in a sequential way. Then, the SMOTE Big Data approach [41] was applied for each binary subset of imbalanced binary class to balance the data distribution, following the same scheme as suggested in [62]. Finally, to carry out the classification step the Random Forest implementation of Mahout was used [55,56]. This work is interesting as a first step on the topic, but it lacks from a true Big Data experimental framework as all datasets selected contain less than 5,000 examples.

Summary

We can see that the recent years have seen a significant interest in adapting current methods to work on Big Data computing paradigms for imbalanced data. Between undersampling and oversampling, the latter is the widely used approach, and it seems to be more robust to the scalability in terms of number of Maps. All these existing implementations can be regarded as the state of the art to improve the performance with more sophisticated techniques, both for the binary and multi-class imbalanced datasets.

Cost-sensitive learning studies

In this section we enumerate several methodologies that include algorithmic modifications for taking into account a higher significance for the positive class. Specifically, four approaches are being revised, two of which are based on Support Vector Machines (SVMs), and two for rule-based systems, i.e., decision trees (random forest) and fuzzy rule learning.

Cost-sensitive SVM

In [49] a cost-sensitive SVM using randomized dual coordinate descent method (CSVM-RDCD) was proposed. The authors performed an experimental study with several datasets, where they compared their proposed approach with some cost-sensitive SVMs from the state of the art. However, in this paper, the authors did not use any of the existing

solutions for the development of algorithms to address massive amounts of data, such as algorithms based on Hadoop or Spark frameworks. The proposal performed iterative calculations that could not be carried out when the problem size grows in size.

Instance weighting SVM

Another approach based on SVMs can be found in [50]. In the aforementioned research, authors combined an instance-weighted variant of the SVM with a Parallel Meta-learning algorithm using MapReduce. Specifically, a symmetric weight-boosting method was developed to optimize the instance-weighted SVM. In the MapReduce design, each Map process applies a sequential Instance Boosting SVM algorithm in the examples of its partition and generates a base learner. Then, the models generated by the all Maps form an ensemble of classifiers. Therefore, no Reduce step is used as no fusion of the models was required. One of the limitations of this MapReduce scheme is the iterative process that is performed in each Map task. In addition, the datasets used in the experiments do not exceed half a million instances, raising the question whether this approach can be scalable for real Big Data problems.

Cost-sensitive random forest

In addition to the study of data pre-processing techniques, another contribution made in [41] was the extension of the Random Forest classifier to a cost-sensitive learning approach for enhancing the learning of the minority class examples. In particular, it consisted of two MapReduce process. The first process was devoted to the creation of the model where each map task built a subset of the forest with the data block of its partition and generated a file containing the built trees. Then, the second MapReduce process was initiated to estimate the class associated to a data test set. In this process, each map estimated the class for the examples available in its partition using the previously learned model, and then the predictions generated by each map were concatenated to form the final predictions file.

Cost-sensitive fuzzy rule-based classification system (FRBCS)

In [51] authors extended Chi-FRBCS-BigData, a MapReduce implementation of a FRBCS made in [63], to address imbalanced Big Data. Specifically, they modified the computation of the rule weights during the learning stage by considering the data distribution. This way, the initial fuzzy learning algorithm was transformed to a cost-sensitive learning scheme, which authors noted as Chi-FRBCS-BigDataCS. Following the workflow defined in [63], the Chi-FRBCS-BigDataCS algorithm consisted of two MapReduce procedures: the first MapReduce process was devoted to the creation of the model, then, the second MapReduce process was responsible to estimate the class associated to a dataset. More specifically, in the first MapReduce process, each Map process was responsible for building a rule base using only the data included in its partition, then, the Reduce process was responsible for collecting and combining the rule bases generated by each map task to form the final rule base. When the first MapReduce process devoted to the building of the model had finished, the second MapReduce process was initiated. In this process, each map task estimated the class for the examples included in its data partition using the previous learned model, then, the predictions generated by each map were aggregated to conform the final predictions file. The classification job did not include a reduce step.

To analyze the quality of their proposed approach, the authors run the experiments over three datasets up to 6 million instances from the UCI repository [57]. The experimental study showed that the proposal is able to handle imbalanced Big Data obtaining competitive results both in the classification performance of the model and the time needed for the computation.

Summary

Cost-sensitive classification has not witnessed the body of works as with the data or algorithmic-based approaches for directly addressing the issues of class imbalance. This could also imply the complexity of the underlying process of cost-sensitive classification—from procuring costs for different types of errors to the algorithmic complexity.

Applications on imbalanced Big Data

In addition to novel proposals and experimental analysis, there are also significant applications in the area of imbalanced Big Data. It is of extreme importance not only to design novel approaches for the research community, but also to add a practical perspective that can be of interest for common users and corporations. In this section, we give three examples of real application areas for Big Data, the first one for bioinformatics, second one for traffic accident prediction, and the last one for biomedical purposes.

Pairwise ortholog detection

In [52] authors focused on the Pairwise Ortholog Detection (POD) problem. It combined several gene pairwise features (alignment-based and synteny measures with others derived from the pairwise comparison of the physicochemical properties of amino acids) to address Big Data problems. The methodology followed to address this problem consisted

of three steps: (1) the calculation of gene pair features to be combined, (2) the building of the classification model using machine learning algorithms to deal with Big Data from a pairwise dataset, and (3) the classification of related gene pairs. To achieve high-quality results, authors made use of several Big Data supervised techniques that manage imbalanced datasets. Specifically, they selected those presented in [41,43] such as Random Forest for Big Data with Cost-Sensitive (RF-BDCS), Random Oversampling with Random Forest for Big Data (ROS + RF-BD) and the Support Vector Machines for Big Data (SVM-BD) for the Apache Spark MLib [64] combined with Random Oversampling (ROS + SVM-BD). The effectiveness of the supervised approach for POD is compared to the well-known unsupervised Reciprocal Best Hits, Reciprocal Smallest Distance and a Automated Project for the Identification of Orthologs from Complete Genome Data algorithms. For the experiments, the authors focused on benchmark datasets derived from the following yeast genome pairs: S. cerevisiae and K. lactis, S. cerevisiae and C. glabrata and S. cerevisiae and S. pombe. Four datasets were derived from each genome pair comparison with different alignment settings. The authors found that the supervised approach outperformed traditional methods, mainly when they applied ROS combined with SVM-BD.

Traffic accidents prediction

In [53] authors developed a data mining process for classification of imbalance data based on MapReduce to predict traffic accidents with the highway traffic data. More concretely, the previous paper presented a classification analysis process of imbalance data prediction based on Apache Hadoop [39], Hive [65] and Mahout [56]. It consisted of five processing steps: (1) a pre-processing step that combined the datasets and creates training datasets (using Hive), (2) oversampling technique to solve imbalance data problem in the training dataset, (3) cluster classification analysis to discover the numbers of cluster and the data ratio of cluster using the *k*-means MapReduce implementation from Mahout, (4) a classification analysis with several clusters using a MapReduce implementation of logistic regression (also from Mahout) and, (5) analysis of the results. To validate the classification analysis process, the authors used data form Korea Highway Corporation which contain traffic data created between Jan. 1st, 2011 and Jun. 30th, 2013 on the Gyeongbu line which connects Seoul with Busan, having a total size of about 300GB. This work was an extension of their previous approach presented in [66] by including a MapReduce implementation of the SMOTE algorithm. The first MapReduce task was responsible to calculate distances among every example. In the second MapReduce job, each Map task was devoted to sort the results by the distances so that every examples *k*-NN are revealed. Then, in the Reduce phase, the SMOTE calculations were conducted to create synthetic examples using the *k*-NN and the attributes of the entire dataset from the Hadoop Distributed Cache [39]. This last feature imposes hard constraints for the scalability of the algorithm.

Biomedical data

A large-scale machine learning classifier based on functional networks was used in [54] for the classification of biomedical data. In their methodology, an iterative process is carried out to sample the data in a MapReduce sampling strategy prior to the learning stage of the model until the accuracy reaches a stable value. The algorithm is based on a neural network using a the Newton–Raphson's method with the maximum likelihood, obtaining more robust results than other well-known algorithms such as SVMs, feed-forward neural networks, *k*-NN or random forest techniques.

Summary

Addressing real problems maybe much harder than trying to design and test a given model. In the previous case, we must take into account the specific features of the case study and to adapt or create new models to obtain the highest performance. We have reviewed three different cases: in the first one, the random forest and SVM classifiers have been applied in conjunction with data pre-processing and cost-sensitive learning with the objective of finding the most robust solution. In the second case study, a combination of several steps, including SMOTE pre-processing, clustering, and logistic regression, allows the achievement of quality solutions. In the last case, a data reduction scheme was used for the biomedical data in conjunction with neural networks.

Practical study on imbalanced Big Data classification using MapReduce

In this section, we study the current performance achieved by classification algorithms in synergy with pre-processing techniques in the context of imbalanced Big Data problems. With this aim, we will make use of three different models under the MapReduce framework. On the one hand, we have implemented Random OverSampling (ROS) and Random UnderSampling (RUS) for MapReduce using the Spark framework. On the other hand, we have selected the original SMOTE-BigData Hadoop implementation from [41] ("Analysis of pre-processing techniques in MapReduce" section).

We will enumerate several lessons learned and to carry out a discussion on the constraints derived by the lack of data in the Map partitions ("The lack of data for the Map stage in imbalanced classification for Big Data" section).

Table 3 Summary of datasets

Datasets	#Ex.	#Atts.	Class (maj; min)	#Class(maj; min)	%Class(maj; min)	IR
ECBDL14-subset-12mill-90features	12,000,000	90	(0; 1)	(11,760,000; 240,000)	(98; 2)	49
ECBDL14-subset-0.6mill-90features	600,000	90	(0; 1)	(588,000; 12,000)	(98; 2)	49

Analysis of pre-processing techniques in MapReduce

In this section we carry out an analysis of some pre-processing approaches that follow the standard MapReduce workflow, i.e., using the classical algorithms in each Map and fusing the instances in the Reduce phase.

We have implemented the RUS and ROS algorithms, available in the Spark package called Imb-sampling-ROS_and_RUS.[3] In addition, we have used the Hadoop implementation for the SMOTE algorithm based on MapReduce [41].

One of the goals of this study is to confirm that the use of pre-processing is beneficial for Big Data problems with a skewed class distribution. Specifically, we seek to observe a similar improvement than in the standard imbalanced class case study with respect to not applying any ad hoc solution. In addition, we analyze the behavior of the RUS, ROS and SMOTE pre-processing algorithms in terms of the number of selected Maps, i.e., the degree of partitioning of the original dataset.

To compare the performance of these pre-processing approaches, we have selected the ECBDL14 dataset that was used at the data mining competition of the Evolutionary Computation for Big Data and Big Learning held on July 14, 2014, in Vancouver (Canada), under the international conference GECCO-2014 [67]. We created two imbalanced datasets from the ECBDL14 data with the same class distribution as in the original data. Since the ECBDL14 dataset contains a large number of features, we used the DEFW-BigData algorithm [42] to improve the classification performance by obtaining the most relevant features. At the end of this process, we obtained a subset of 90 of the 631 original features.

Table 3 shows the details of the selected datasets, including the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (Class (maj;min)), number of instances for each class (#Class (maj; min)), class distribution (%Class (maj; min)) and imbalance ratio (IR).

We applied different data pre-processing techniques, namely RUS, ROS, and SMOTE, and evaluated their impact using decision tree and random forest implementations in both Spark and Hadoop. The parameters specification for these classifiers is listed in Table 4. For more details about the internal operation of the algorithms, the reader can refer to the MLlib guide[4] [64] (Spark algorithms) or to the Mahout library[5] [56] (MapReduce algorithms).

Table 4 Configuration parameters for the Spark and Hadoop algorithms

Algorithm	Parameters
Decision trees (DT) and random forest (RF-S)	Number of trees: 1 (DT); 100 (RF-S)
	Number of bins used when discretizing continuous features: 100
	Impurity measure: gini
	Randomly selected attributes: 9
	Maximum depth of each tree: 5
Random forest Hadoop (RF-H)	Number of trees: 100
	Randomly selected attributes: 7
	Maximum depth of each tree: unlimited

We have used the parameter values recommended/used by the authors. The datasets obtained by the ROS, RUS and SMOTE algorithms were generated to achieve a balanced class distribution.

Regarding the infrastructure used to perform the experiments, we used the Atlas cluster at University of Granada, which consists of 16 nodes connected via a 40Gb/s Infiniband network. Each node has two Intel Xeon E5-2620 microprocessors (at 2.00 GHz, 15MB cache) and 64GB of main memory working under Linux CentOS 6.6. The head node of the cluster has two Intel Xeon E5-2620 microprocessors (at 2.00 GHz, 15MB cache) and 96GB of main memory. 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 1.6.2, where the head node is configured as master and NameNode, and the rest are workers and DataNodes.

Table 5 shows the average GM results in training and test sets for the Spark and Hadoop classifiers using 1, 8, 16, 32 and 64 partitions over the two Big Data cases of study. Note that the experiments using 1 partition are trying to emulate a sequential behavior, using exactly the same methods for a fair comparison. The underlined values highlight the best value in test with regard to the number of Maps used to distribute the execution, i.e., the highest value by row. Additionally, bold

[3] https://spark-packages.org/package/saradelrio/Imb-sampling-ROS_and_RUS.

[4] https://spark.apache.org/mllib/.

[5] http://mahout.apache.org/.

Table 5 Average GM results for the Spark and Hadoop algorithms using 1, 8, 16, 32 and 64 partitions on the big data cases of study

Algorithm	1 partitions		8 partitions		16 partitions		32 partitions		64 partitions	
	GM_{tr}	GM_{tst}	GM_{tr}	GM_{tst}	GM_{tr}	GM_{tst}	GM_{tr}	GM_{tst}	GM_{tr}	GM_{tst}
ECBDL14-subset-0.6mill-90features										
RF-H										
Without pre-processing	0.56912	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
ROS	1.00000	0.02240	0.98590	0.49250	0.94960	0.64410	0.89980	**0.67770**	0.85110	**0.66520**
SMOTE	0.37530	0.12380	0.13270	0.08060	0.12760	0.07070	0.12640	0.09210	0.13780	0.07740
RUS	0.85480	**0.66450**	0.74610	**0.65550**	0.72660	**0.65970**	0.71910	0.65540	0.70300	0.64610
RF-S										
Without pre-processing	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
ROS	0.71220	0.64909	0.71241	**0.65035**	0.70892	0.64709	0.71142	**0.64666**	0.71121	**0.65042**
SMOTE	0.75876	0.60567	0.76859	0.62237	0.76594	0.62036	0.77538	0.62326	0.78057	0.62496
RUS	0.71437	**0.65197**	0.71779	0.64921	0.71289	**0.64782**	0.71458	0.63898	0.71683	0.64767
DT										
Without pre-processing	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
ROS	0.70195	**0.62987**	0.70408	0.62827	0.70195	0.62987	0.70504	**0.62873**	0.70211	**0.62987**
SMOTE	0.71366	0.55672	0.72406	0.50999	0.73219	0.45379	0.73166	0.44642	0.73323	0.47313
RUS	0.70828	0.62528	0.70542	**0.62987**	0.70583	**0.63204**	0.70413	0.62584	0.70390	0.61696
ECBDL14-subset-12mill-90features										
RF-H										
Without pre-processing	*	*	0.02579	0.00500	0.01000	0.00000	0.00447	0.00000	0.00000	0.00000
ROS	*	*	0.98350	0.50720	0.95120	0.63760	0.90890	**0.69310**	0.86160	**0.70560**
SMOTE	*	*	N.D.	N.D.	N.D.	N.D.	0.06625	0.05005	0.07765	0.03535
RUS	*	*	0.75970	**0.69920**	0.74340	**0.69510**	0.73370	0.69190	0.72550	0.68880
RF-S										
Without pre-processing	*	*	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
ROS	*	*	N.D.	N.D.	0.70902	0.66998	0.70673	**0.67091**	0.70599	0.66695
SMOTE	*	*	N.D.	N.D.	N.D.	N.D.	0.75375	0.63239	0.75816	0.63184
RUS	*	*	0.70911	**0.66983**	0.70887	**0.67055**	0.70827	0.66700	0.70780	**0.66802**
DT										
Without pre-processing	*	*	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
ROS	*	*	N.D.	N.D.	0.70433	0.66517	0.70422	**0.66472**	0.70396	0.66551
SMOTE	*	*	N.D.	N.D.	N.D.	N.D.	0.71970	0.46296	0.71037	0.44341
RUS	*	*	0.70540	**0.66625**	0.70508	**0.66652**	0.69734	0.66406	0.70618	**0.66615**

Boldunderlined values indicate overall best results for a given preprocessing technique

values indicate which is the best pre-processing technique for each classifier and number of Maps, i.e., the highest value by column. The N.D. (Not Determinable) symbol indicates that the algorithm was not able to complete the experiment, for example when the Java Virtual Machine cannot allocate an object because it ran out of memory. Moreover, the * symbol indicates that the experiment did not proceed, since it is unfeasible to run a sequential algorithm (1 partition) on a dataset of 12 million instances.

According to the results obtained we can reach the following conclusions:

- Classification models are shown to be more accurate in the case of making use of the full dataset, instead of the 5% sampling. This implies the benefit on the use of Big Data technologies, taking advantage of the whole representation of the problem to be solved. However, we must take into account two possible issues that may hinder the classification ability in these cases. On the one hand, the amount of noise presented in this kind of data may also be higher. On the other hand, it may suffer from the curse of dimensionality; therefore, adds more complexity to the modeling.

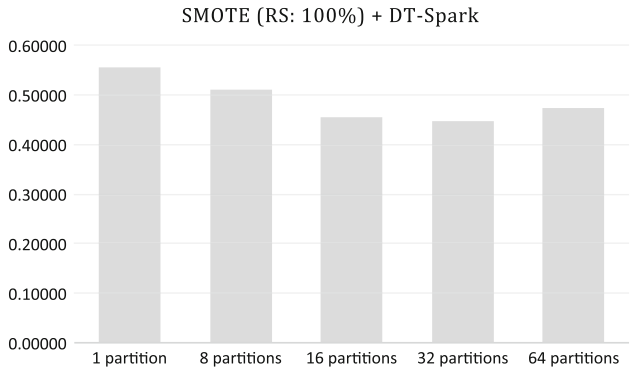


Fig. 2 Average results for the MapReduce version of SMOTE combined with the Spark version of Decision Trees over the ECBDL14-subset-0.6mill-90features dataset using the GM measure

- The results obtained with SMOTE combined with RF-S are better with respect to those obtained with SMOTE combined with RF-H or DT. Therefore, there is a significant influence of the implementation of the RF-S method to achieve this performance.
- ROS and RUS show higher quality results than SMOTE-BigData (MapReduce implementation from [41]) in all experiments. This implies a complexity in the SMOTE algorithm for Big Data that we will discuss further. This behavior is also depicted in Fig. 2, which shows how the classification performance for the SMOTE algorithm is hindered to a certain limit as the number of partitions increases.
- Regarding the previous point, we have further analyzed the source of these differences in performance between SMOTE and the random sampling techniques. Specifically, we used the full ECBDL14 dataset to show in Table 6 the true rates for the positive and negative classes obtained by the decision tree in the test partitions. We may observe that there is a good trade-off between both metrics in the case of ROS and RUS. However, for SMOTE pre-processing there is a bias towards the negative classes.
- The inner working procedure of both ROS and RUS, which is based on the sampling of the minority versus majority class, allows them to be scalable approaches.
- Finally, we must state that the degree of performance achieved mostly depends on the behavior of the learning algorithm. However, when contrasting ROS and RUS we observe better results for the former. This is due to the fact that ROS is somehow independent for the number of Maps, as it just makes exact copies of the instances which are shuffled among the chunks of data. On the contrary, for RUS the data distribution changes when removing some of the instances. Additionally, increasing the number of partitions have a more severe effect for RUS due to the lack of data.

The lack of data for the Map stage in imbalanced classification for Big Data

In the previous section, we have observed how the performance achieved by the algorithms is significantly degraded as we increase the number of Maps in search for efficiency and scalability. This way, fewer data are to be used in the learning stage within each Map process, leading to sub-optimal models.

This locality of the data problem is commonly referred to as the lack of sufficient data in the training partitions [37,38]. Measuring statistically the minimum required amount of information in a given problem to guarantee a correct classification is a hard task. Related studies confirmed that the power of classifiers suffered a dramatic reduction in the event of both imbalance and lack of data [68].

Back to the particular case of the MapReduce scheme, this fragmentation causes the problem representation in each Map (chunk of data) to be possibly quite different to the initial one. In addition, fewer positive class instances are included into each subset of data for the Map processes, and models are clearly biased.

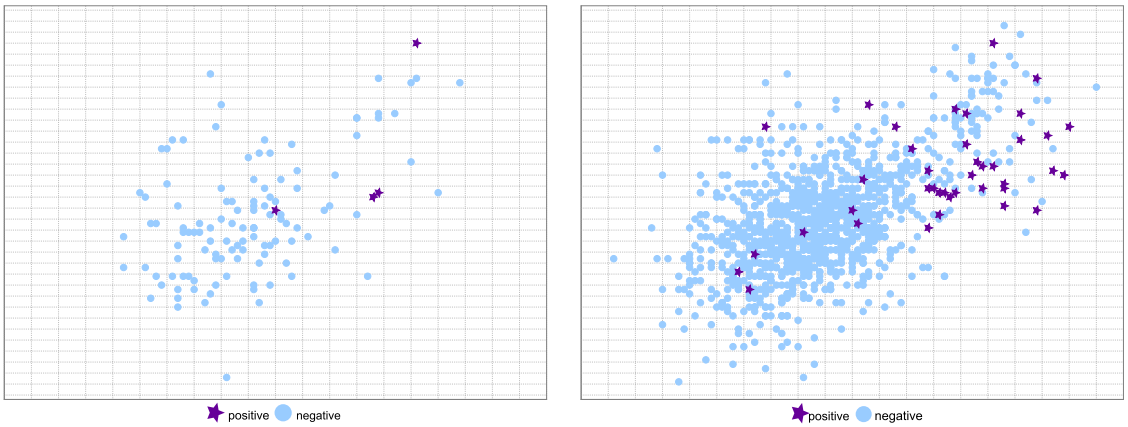
A visual representation of the lack of data problem is depicted in Fig. 3, where we show a scatter plot for the training data of the yeast4 problem from KEEL imbalanced dataset repository[6] [69] (attributes mcg vs. gvh) with only a 10 % of the original instances (Fig. 3a) and with the entire dataset (Fig. 3b). We can observe that the modeling process becomes harder in the former case. In particular, how can any learning algorithm obtain a classifier that is able to perform a good generalization when there is not enough data that represent the boundaries of the problem? In particular, the concentration of minority examples is so low that they can be simply treated as noise.

This behavior is shown in Table 5 in accordance to the differences between the results of the two datasets in favor of the larger one (12 million instances). This is especially of interest when contrasting the results of the RUS and ROS pre-processing algorithms. In the case of RUS, when the number of Maps increases the number of data for each subset becomes lower not only because of the repartition, but also for the removal of the majority class instances. This problem is accentuated when we address highly imbalanced problems, such as the ECBDL dataset. On the contrary, when applying ROS, i.e., maintaining all examples but adding a "weight" to the minority ones, the performance obtained increases when adding a higher number of Maps. These good results can be due to the issue of obtaining a higher number of models, each one within a different Map, but avoiding the issue of the lack of data by means of the instances replication. Then, the

[6] http://www.keel.es/imbalanced.php.

Table 6 Average TPR and TNR results in test for the DT algorithm using 1, 8, 16, 32 and 64 partitions on the ECBDL14-subset-12mill-90features case of study

Algorithm	1 partitions		8 partitions		16 partitions		32 partitions		64 partitions	
	TPR_{tst}	TNR_{tst}	TPR_{tst}	TNR_{tst}	TPR_{tst}	TNR_{tst}	TPR_{tst}	TNR_{tst}	TPR_{tst}	TNR_{tst}
ECBDL14-subset-12mill-90features										
DT										
Without pre-processing	*	*	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
ROS	*	*	N.D.	N.D.	0.63830	0.69317	0.63980	0.69062	0.65800	0.67312
SMOTE	*	*	N.D.	N.D.	N.D.	N.D.	0.26040	0.82308	0.23345	0.84221
RUS	*	*	0.65790	0.67470	0.64845	0.68508	0.66395	0.66417	0.64960	0.68312



(a) 10 % of training instances **(b)** 100 % of training instances

Fig. 3 Lack of density or small sample size on the yeast4 dataset. *Stars* depicts the positive class, whereas *circles* represent the negative class

fusion of all these diverse and accurate models in the Reduce step provides a better overall classifier.

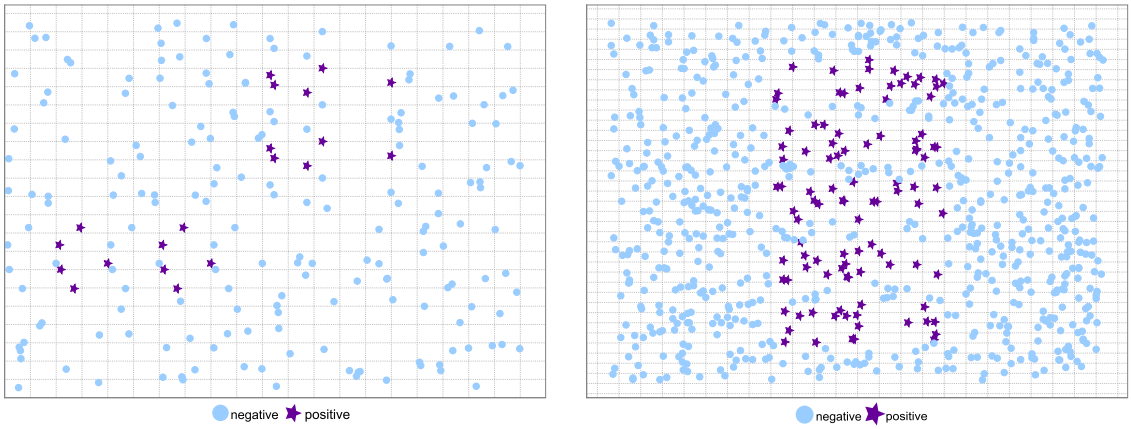
A problem related to the lack of data is the potential presence of small disjuncts [3,35] in the data associated with each Map, also as a consequence of the data division process. Small disjuncts occur when the concepts are represented within small clusters. They are also known as "rare cases", as they comprise few training examples. This situation is illustrated in Fig. 4. We depict an artificially generated dataset with small disjuncts for the minority class and the "*Subclus*" problem created in [70]. In those datasets we can find small disjuncts for both classes. We may observe that negative samples are underrepresented with respect to the positive samples in the central region of positive rectangular areas, whereas positive samples only cover a small part of the whole dataset and are placed inside the negative class.

Very specific rules or models must be created for these instances, which are discarded most of the times in favor of most general ones. Additionally, when the classification algorithm is based on a divide-and-conquer approach, such as in decision trees, it may lead to data fragmentation. When the initial representation of the minority class is small, this hindrance is obviously more severe. In summary, for the learning system it becomes hard to know whether these examples represent an actual sub-concept or are merely attributed to noise [37].

These challenges seem to have a stronger influence on the SMOTE algorithm [28]. We may observe from Tables 5 and 6 and/or Fig. 2 that the SMOTE algorithm performance deteriorates, as compared to RUS and ROS. This behavior is clearer as the number of partitions increases. This is because SMOTE is relying on the positive class neighborhood to generate new examples, and when that neighborhood is sparse and disjointed, it becomes challenging for SMOTE to achieve its potential. Additionally, the higher the number of features, the harder the computation of the actual neighbors and the higher the variance for the newly created instances [71].

Challenges for imbalanced Big Data classification

Throughout this paper we have carried out a deep review for those solutions for imbalanced classification in the context of Big Data. We selected popular data pre-processing and classification techniques from the literature, and carried out a detailed empirical study to determine how the data fragmentation in the Map process affects different techniques.



(a) Artificial dataset: small disjuncts for the minority class

(b) Subclus dataset: small disjuncts for both classes

Fig. 4 Example of small disjuncts on imbalanced data. *Stars* depicts the positive class, whereas *circles* represent the negative class

We summarize some of the significant challenges for imbalanced classification in Big Data:

1. *There is a necessity for a thorough design at the implementation level for current algorithms* In other words, an effort for the design and development of robust methodologies to address Big Data imbalanced problems has to be made. Novel Big Data programming frameworks such as Spark [19] add different operators that can ease the codification of this kind of solutions, allowing to take advantage of the iterative features of these operators.
2. *The design of novel algorithms for the generation of artificial instances* To achieve this goal, the different level of partitioning must be taken into account for the sake of maintaining the robustness of the modeling when seeking for a higher level of scalability and predictive performance.
   In addition, other resampling strategies can be considered to counteract simultaneously the between-class imbalance and the within-class imbalance [37]. The main idea is to identify these small regions of data by means of clustering approaches, and to stress the significance of these areas by generating data within this area.
   We posit that it is an opportunity to expand SMOTE for a Spark or Hadoop implementation to counter the challenges of small disjuncts, fewer number of positive class examples, and high dimensionality. An appropriate extension of SMOTE will then allow us to leverage the power of SMOTE in the Spark/Hadoop frameworks of tackling Big Data.
3. *Considering different trade-offs for the ratio between classes* It has been shown that the standard 1:1 might not be the best class distribution to solve the imbalanced classification problem [72,73]. Therefore, the data fragmentation and locality related to the different subsets in each Map process can be overcome by means of the generation of additional data. In this sense, we may refer to the findings obtained in [42,43] in which a higher ratio of oversampling allows the achievement of better results.
4. *Focus on the MapReduce workflow* First, we can act on the learning classifier itself with each Map task. Because instances in the small disjuncts are likely to be difficult to predict, one could possibly use Boosting algorithms to improve their classification performance [27].
   Second, we can also take advantage of the MapReduce programming scheme focusing on the Reduce stage. Specifically, we must analyze two different schemes for the classification techniques: (1) carrying out a model aggregation (fusion) from the outputs of every Map process or (2) building an ensemble system and combine their predictions during the inference process.
   Therefore, we must be aware on the twofold perspective "fusion" versus "ensemble" of models in the MapReduce scheme, and how to introduce diversity within each Map process so that the joint of the single models can lead to an optimal solution.

Concluding remarks

In this paper, we have carried out an overview on the topic of imbalanced classification in the scenario of Big Data. With this aim, first we have presented this problem from a traditional Machine Learning perspective, providing the traditional solutions designed to address this task.

Then, we have focused on those current solutions developed for Big Data problems. We have organized these approaches into three parts considering whether they use a data pre-processing approach, an algorithmic modification via cost-sensitive learning, or they rather aim to solve a given real application.

We have carried out an experimental study to show the behavior of standard data pre-processing techniques using a MapReduce framework. Taking these results into account, we have carried out a discussion focusing on two main issues. On the one hand, the open problems related to inner data characteristics such as the lack of data and small disjuncts generated in the data partitions in the Map processes. On the other hand, we demonstrated how these issues affect the performance of data pre-processing techniques like SMOTE.

We posited several challenges that must be taken into account to develop high-quality solutions in this area of research. First, a detailed design of artificial data generation techniques to improve the behavior of pre-processing approaches. Second, study the different possibilities related to the fusion of models or the management of an ensemble system with respect to the final Reduce task.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science and Technology under Projects TIN2014-57251-P and TIN2015-68454-R, the Andalusian Research Plan P11-TIC-7765, the Foundation BBVA Project 75/2016 BigDaPTOOLS, and the National Science Foundation (NSF) Grant IIS-1447795.

References

1. He H, García EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
2. Sun Y, Wong AKC, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell 23(4):687–719
3. López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250(20):113–141
4. Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Prog Artif Intell 5(4):221–232
5. Prati RC, Batista GEAPA, Silva DF (2015) Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. Knowl Inf Syst 45(1):247–270
6. Yu H, Hong S, Yang X, Ni J, Dan Y, Qin B (2013) Recognition of multiple imbalanced cancer types based on DNA microarray data using ensemble classifiers. BioMed Res Int 2013:1–13
7. Chen Y-S (2016) An empirical study of a hybrid imbalanced-class DT-RST classification procedure to elucidate therapeutic effects in uremia patients. Med Biol Eng Comput 54:983–1001
8. Haixiang G, Yijing L, Yanan L, Xiao L, Jinling L (2016) BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification. Eng Appl Artif Intell 49:176–193
9. Elhag S, Fernández A, Bawakid A, Alshomrani S, Herrera F (2015) On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. Expert Syst Appl 42(1):193–202
10. Batista GEAPA, Prati RC, Monard MC (2004) A study of the behaviour of several methods for balancing machine learning training data. SIGKDD Explor 6(1):20–29
11. Ramentol E, Vluymans S, Verbiest N, Caballero Y, Bello R, Cornelis C, Herrera F (2015) IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. IEEE Trans Fuzzy Syst 23(5):1622–1637
12. Domingos P (1999) Metacost: A general method for making classifiers cost-sensitive. In: Proceedings of the 5th international conference on knowledge discovery and data mining (KDD'99), pp 155–164
13. López V, Fernández A, Moreno-Torres JG, Herrera F (2012) Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. Expert Syst Appl 39(7):6585–6608
14. Fernández A, Río S, López V, Bawakid A, del Jesus MJ, Benítez J, Herrera F (2014) Big data with cloud computing: an information sciencesight on the computing environment. MapReduce and programming framework. WIREs Data Min Knowl Discov 4(5):380–409
15. Kambatla K, Kollias G, Kumar V, Grama A (2014) Trends in big data analytics. J Parallel Distrib Comput 74(7):2561–2573
16. Zikopoulos PC, Eaton C, deRoos D, Deutsch T, Lapis G (2011) Understanding big data—analytics for enterprise class hadoop and streaming data, 1st edn. McGraw-Hill Osborne Media, New York
17. Chen CP, Zhang C-Y (2014) Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf Sci 275:314–347
18. Wu X, Zhu X, Wu G-Q, Ding W (2014) Data mining with Big Data. IEEE Trans Knowl Data Eng 26(1):97–107
19. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauly M, Franklin MJ, Shenker S, Stoica I (2012) Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Presented as part of the 9th USENIX symposium on networked systems design and implementation (NSDI 12), USENIX, San Jose, CA, pp 15–28
20. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
21. Li J, Fong S, Sung Y, Cho K, Wong R, Wong KKL (2016) Adaptive swarm cluster-based dynamic multi-objective synthetic minority oversampling technique algorithm for tackling binary imbalanced datasets in biomedical data classification. BioData Min 9(1):1–15
22. Tomczak JM, Zieba M (2015) Probabilistic combination of classification rules and its application to medical diagnosis. Mach Learn 101(1–3):105–135
23. Huang J, Ling CX (2005) Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 17(3):299–310
24. Barandela R, Sanchez J, García V, Rangel E (2003) Strategies for learning in class imbalance problems. Pattern Recognit 36(3):849–851
25. Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. Addison Wesley, Reading
26. Rokach L (2010) Ensemble-based classifiers. Artif Intell Rev 33(1):1–39
27. Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensembles for class imbalance problem: bagging, boosting and hybrid based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484
28. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
29. He H, Bai Y, García EA, Li S (2008) ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of the 2008 IEEE international joint conference neural networks (IJCNN'08), pp 1322–1328

30. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2012) DBSMOTE: density-based synthetic minority over-sampling technique. Appl Intell 36(3):664–684
31. Barua S, Islam MM, Yao X, Murase K (2014) MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 26(2):405–425
32. Stefanowski J (2016) Dealing with data difficulty factors while learning from imbalanced data. In: Matwin S, Mielniczuk J (eds), Challenges in computational statistics and data mining. Studies in computational intelligence, vol 605. Springer, Berlin, pp 333–363
33. García V, Mollineda RA, Sánchez JS (2008) On the k-NN performance in a challenging scenario of imbalance and overlapping. Pattern Anal Appl 11(3–4):269–280
34. Alshomrani S, Bawakid A, Shim S-O, Fernández A, Herrera F (2015) A proposal for evolutionary fuzzy systems using feature weighting: dealing with overlapping in imbalanced datasets. Knowl Based Syst 73:1–17
35. Weiss GM (2010) The impact of small disjuncts on classifier learning. In: Stahlbock R, Crone SF, Lessmann S (eds) Data mining. Springer, Berlin. Ann Inf Syst 8:193–226
36. Sáez JA, Luengo J, Stefanowski J, Herrera F (2015) SMOTE-IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Inf Sci 291:184–203
37. Jo T, Japkowicz N (2004) Class imbalances versus small disjuncts. ACM SIGKDD Explor Newslett 6(1):40–49
38. Wasikowski M, Chen X-W (2010) Combating the small sample class imbalance problem using feature selection. IEEE Trans Knowl Data Eng 22(10):1388–1400
39. White T (2015) Hadoop: the definitive guide, 4th edn. O'Reilly Media, Sebastopol
40. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I (2010) Spark: cluster computing with working sets. In: HotCloud 2010, pp 1–7
41. Río S, López V, Benítez J, Herrera F (2014) On the use of MapReduce for imbalanced Big Data using random forest. Inf Sci 285:112–137
42. Triguero I, Río S, López V, Bacardit J, Benítez JM, Herrera F (2015) ROSEFW-RF: the winner algorithm for the ECBDL'14 Big Data competition: an extremely imbalanced Big Data bioinformatics problem. Knowl Based Syst 87:69–79
43. Río S, Benítez JM, Herrera F (2015) Analysis of data preprocessing increasing the oversampling ratio for extremely imbalanced Big Data classification. In: Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, vol 2, pp 180–185
44. Triguero I, Galar M, Vluymans S, Cornelis C, Bustince H, Herrera F, Saeys Y (2015) Evolutionary undersampling for imbalanced Big Data classification. In: IEEE congress on evolutionary computation (CEC), pp 715–722
45. Kamal S, Ripon SH, Dey N, Ashour AS, Santhi V (2016) A MapReduce approach to diminish imbalance parameters for big deoxyribonucleic acid dataset. Comput Methods Programs Biomed 131:191–206
46. Hu F, Li H, Lou H, Dai J (2014) A parallel oversampling algorithm based on NRSBoundary-SMOTE. J Inf Comput Sci 11(13):4655–4665
47. Zhai J, Zhang S, Wang C (2015) The classification of imbalanced large data sets based on MapReduce and ensemble of elm classifiers. Int J Mach Learn Cybern. doi:10.1007/s13042-015-0478-7
48. Bhagat RC, Patil SS (2015) Enhanced smote algorithm for classification of imbalanced big-data using random forest. In: Souvenir of the 2015 IEEE international advance computing conference, IACC 2015, pp 403–408
49. Tang M, Yang C, Zhang K, Xie Q (2014) Cost-sensitive support vector machine using randomized dual coordinate descent method for big class-imbalanced data classification. In: Abstract and applied analysis 2014, pp 416591:1–416591:9
50. Wang X, Liu X, Matwin S (2014) A distributed instance-weighted SVM algorithm on large-scale imbalanced datasets. In: Proceedings of the 2014 IEEE international conference on Big Data, 2014, pp 45–51
51. López V, Río S, Benítez JM, Herrera F (2015) Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced Big Data. Fuzzy Sets Syst 258:5–38
52. Galpert D, Río S, Herrera F, Ancede-Gallardo E, Antunes A, Agnero-Chapin G (2015) An effective Big Data supervised imbalanced classification approach for ortholog detection in related yeast species. BioMed Res Int
53. Park S-H, Kim S-M, Ha Y-G (2016) Highway traffic accident prediction using VDS Big Data analysis. J Supercomput 72:2815–2831
54. Elsebakhi E, Lee F, Schendel E, Haque A, Kathireason N, Pathare T, Syed N, Al-Ali R (2015) Large-scale machine learning based on functional networks for biomedical Big Data with high performance computing platforms. J Comput Sci 11:69–81
55. Owen S, Anil R, Dunning T, Friedman E (2011) Mahout in action, 1st edn. Manning Publications Co., Greenwich
56. Lyubimov D, Palumbo A (2016) Apache Mahout: beyond MapReduce, 1st edn. CreateSpace Independent, North Charleston
57. Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
58. Triguero I, Derrac J, García S, Herrera F (2012) Integrating a differential evolution feature weighting scheme into prototype generation. Neurocomputing 97:332–343
59. Triguero I, Galar M, Merino D, Maillo J, Bustince H, Herrera F (2016) Evolutionary undersampling for extremely imbalanced big data classification under apache spark. In: IEEE congress on evolutionary computation (CEC 2016), Vancouver, Canada, pp 640–647
60. Hu F, Li H (2013) A novel boundary oversampling algorithm based on neighborhood rough set model: NRSBoundary-SMOTE. Math Prob Eng 2013:1–10
61. Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2011) An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. Pattern Recognit 44(8):1761–1776
62. Fernández A, López V, Galar M, Del Jesus M, Herrera F (2013) Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches. Knowl Based Syst 42:97–110
63. Río S, López V, Benítez JM, Herrera F (2015) A MapReduce approach to address Big Data classification problems based on the fusion of linguistic fuzzy rules. Int J Comput Intell Syst 8(3):422–437
64. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A (2016) MLlib: machine learning in apache spark. J Mach Learn Res 17(34):1–7
65. Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R (2009) Hive—a warehousing solution over a map-reduce framework. J Very Large DataBases 2(2):1626–1629
66. Park SH, Ha YG (2014) Large imbalance data classification based on MapReduce for traffic accident prediction. In: Proceedings of the 2014 8th international conference on innovative mobile and internet services in ubiquitous computing, IMIS 2014, pp 45–49
67. ECBDL'14 dataset. http://cruncher.ncl.ac.uk/bdcomp/
68. Guo Y, Graber A, McBurney RN, Balasubramanian R (2010) Sample size and statistical power considerations in high-dimensionality data settings: a comparative study of classification algorithms. BMC Bioinform 11:447

69. Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multi-Valued Log Soft Comput 17(2–3):255–287
70. Napierala K, Stefanowski J, Wilk S (2010) Learning from imbalanced data in presence of noisy and borderline examples. In: Proceedings of the 7th international conference on rough sets and current trends in computing (RSCTC'10). Lecture notes on artificial intelligence, vol 6086, pp 158–167
71. Blagus R, Lusa L (2013) SMOTE for high-dimensional class-imbalanced data. BMC Bioinform 14(1):106
72. Weiss GM, Provost FJ (2003) Learning when training data are costly: the effect of class distribution on tree induction. J Artif Intell Res 19:315–354
73. Chawla NV, Cieslak DA, Hall LO, Joshi A (2008) Automatically countering imbalance and its empirical relationship to cost. Data Min Knowl Discov 17(2):225–252