Applying Map-Reduce to Imbalanced Data Classification

Joanna Jędrzejowicz
Jakub Neumann
Piotr Synowczyk
Magdalena Zakrzewska
e-mail: magdalena.zakrzewska@inf.ug.edu.pl
Institute of Informatics,
Faculty of Mathematics, Physics and Informatics,
University of Gdansk,
80-308 Gdansk, Poland,

***Abstract*—The aim of the paper was to apply MapReduce paradigm to the algorithm SplitBal which classifies imbalanced datasets and perform the evaluation of results for different parameters. Parallelization of time consuming operations allows to classify larger datasets, in perspective Big Data.**

***Index Terms*—imbalanced data, classification, parallelization**

1. Introduction

The imbalance classification algorithms deal with datasets in which the cardinality of one of the classes is much bigger than that of other classes. In case of binary classification which is considered in this paper it means that we deal with data containing instances from two classes and instances from one class, majority class, outnumber instances from the other class, that is minority class. As follows from many research projects special approach to classification of imbalanced data has to be applied since traditional classification algorithms do not give satisfactory results. There are several reasons why classifying imbalanced data is a challenge. Standard classifiers such as Support Vector Machines, logistic regression, decision trees, Naive Bayes are not suitable because they often give good coverage of the majority examples but minority classes are misrepresented. Even if the obtained accuracy is for example 99% which would be quite satisfactory for usual classification can be misleading if the minority class is 1% of the whole dataset. What is more, minority class examples can be treated as noise by the classifier, and vice versa, noise wrongly identified as minority class. Therefore special methods are needed. What is more, it appears that imbalanced data occur often in every day life situations. In [1] they are denoted as rare events and include software defects, natural disasters, fraud detection, text classification etc. Misclassifying mentioned rare events can result in high costs which is another reason for looking for better classification methods for imbalanced data.

The most popular methods for classifying imbalanced datasets make use of sampling in the form of over-sampling or under-sampling. Both methods change the distribution of data by eliminating the majority class instances, or increasing the minority class. In such case there is danger of deleting some potentially useful data, or in case of over-sampling, increase the possibility of over-fitting. Another group of methods is based on ensemble learning, for example Bagging or Boosting. The considered SplitBal method is a variant of ensemble learning since for each considered bin (see Section 2) a different classifier can be used. Methods that make use of discriminative ability of features in the context of class imbalance data are considered in [2] and suggest among others one method EDBC which is furthered examined in the paper (Section 2). SplitBal also belongs to a group of specialized algorithms where well examined classification methods are adjusted to the case of imbalanced datasets. Another representative of that group of methods is recently introduced [3] KRNN - which is a modification of k-nearest neighbor classifier for imbalanced case.

For an up to date review of papers and ideas on the imbalanced data classification see [1], [4].

The paper is organized as follows. In Section 2 two algorithms: EDBC and SplitBal as well as the method of parallelizing SplitBal are discussed. Section 3 describes the used benchmark datasets and the results of experiments, and finally Section 4 contains conclusions and future plans.

2. Proposed Approach

In [5] the review of several algorithms for imbalanced data classification was given and exhaustive experiments for different techniques were performed. In this paper we consider two methods studied in [5] which are potentially suited for parallelization. They are: dissimilarity based imbalance data classification (EDBC) introduced in [2] and splitting based data balancing method (SplitBal) introduced in [6].

The first method, EDBC, makes use of three strategies well founded in data mining, that is feature selection and data reduction, prototype selection and data transformation based on dissimilarity calculation. Feature selection strategies are broadly applied in a preprocessing stage of clas-

```
Require: imbalanced training data $TR = \bigcup_{c \in C} TR_c$, $C$ - set of classes, $TR_c = \{< d, c > \mid d \in D\}$, where $D$ is the space of attribute vectors $d = (w_1^d, \dots, w_N^d)$ with $w_i^d$ being a numeric value and $N$ is the number of attributes. $TS$ - testing data, $\tau_1 > 0$ - threshold for clustering-based feature selection, $k$ - number of nearest neighbors, $\tau_2 > 0$ - threshold for selecting prototypes, $dis$ -dissimilarity measure.
Ensure: quality of classification in the form of AUC - area under ROC curve
    {feature selection step}
 1: perform feature selection on $TR$ and filter onto $TR' = \bigcup_{c \in C} TR'_c$ using threshold $\tau_1$,
    {prototype selection step}
 2: for each class $c \in C$ do
 3:     perform Jarvis-Patric clustering on $TR'_c$ with parameter $k$,
 4:     using threshold $\tau_2$ select prototypes of the clusters for the class $P_c = \{p_1^c, \dots, p_r^c\}$,
 5: end for
    {transformation step}
 6: for each class $c \in C$ do
 7:     for each $x \in TR'_c$ do
 8:         perform dissimilarity transformation $x$ into $\hat{x} = (dis(x, p_1^c, \dots, dis(x, p_r^c))$,
 9:     end for
10: end for
11: use $\bigcup_{c \in C} \{\hat{x} : x \in TR'_c\}$ as a training set for the classifier,
12: perform testing on the data set $TS$
13: calculate AUC - area under ROC curve
```

Figure 1. Algorithm EDBC

sification. They contain filtering, wrapping and embedded methods. The main idea is to select a subset of features from the entire feature space that allows the classifier to achieve optimal performance. The core of prototype selection is to perform clustering and then extract representative instances from clusters as prototypes. Data transformation is used to project data into dissimilarity space (see Figure 1 line 8:) to improve discriminant ability of features via dissimilarities among examples. It is worth noting that in case of numerical data classification, by features we usually mean attributes. It is different, for example, in text classification - 'feature' is usually a broader concept than 'attribute'. The algorithm EDBC is used for the experiments (Section 3) to compare with parallel SplitBal. The algorithm EDBC is described in Figure 1 and more details can be found in the original paper [2] and in [5].

The main idea of SplitBal [6] is to divide the majority class instances into several bins so that each bin contains the minority class and a part of majority class of the equal size. Thus the number of bins K is approximately equal to the imbalance ratio $IR = \frac{\text{number of majority class instances}}{\text{number of minority class instances}}$. After the training dataset is divided into bins, for each bin an independent training takes place and base, binary probabilistic classifiers $CL_1, \dots, CL_K$ are evolved. These binary classifiers are further combined into an ensemble to classify new data. As follows from the review of methods performed in [5] with different algorithms for imbalanced data classification, the results for SplitBal were among the best for many benchmark data sets. Applying parallelization to the used algorithm emerged as a natural step as follows from the above description. Training of different bins can be performed in parallel since no data exchange among computations for different bins is needed. What is more parallelization allows to perform experiments on significantly bigger datasets, datasets considered in [5] did not exceed 500 instances. Also the computation time can be significantly reduced in case of parallel experiments.

In our approach for parallel SplitBal the MapReduce paradigm was used. The Map step includes: dividing the training set into bins, for each bin training a classifier on the training set restricted to the bin and then applying the evolved classifier to each instance from the testing set. In the Reduce step the results of the classifiers are merged to evolve the decision of the ensemble. To present the ensemble rules the following notation is assumed. Let r stand for an instance from the testing set which is classified by the i-th classifier with probability P_{ij} as c_j, for $j = 1, 2$. Further let, D_{ij} stand for the average distance of r from the data with class label c_j in the i-th bin. The following ensemble rules allow to calculate final classsification results R_1 and R_2 as shown below.

- MaxDistance

$$R_1 = argmax_{1 \leq i \leq K} \frac{P_{i1}}{D_{i1} + 1},$$

$$R_2 = argmax_{1 \leq i \leq K} \frac{P_{i2}}{D_{i2} + 1}$$

- MinDistance

$$R_1 = argmin_{1 \leq i \leq K} \frac{P_{i1}}{D_{i1} + 1},$$

$$R_2 = argmin_{1 \leq i \leq K} \frac{P_{i2}}{D_{i2} + 1}$$

- ProDistance

$$R_1 = \Pi_{i=1}^{K} \frac{P_{i1}}{D_{i1} + 1},$$

$$R_2 = \Pi_{i=1}^{K} \frac{P_{i2}}{D_{i2} + 1}$$

- MajDistance

$$R_1 = \sum_{i=1}^{K} \frac{sg(P_{i1}, P_{i2})}{D_{i1} + 1}$$

$$R_2 = \sum_{i=1}^{K} \frac{sg(P_{i2}, P_{i1})}{D_{i2} + 1}$$

```
Require: imbalanced training data $TR$, $C = \{c_1, c_2\}$ set
    of classes, $N$ is the number of attributes. $TS$ - testing
    data, $K$ - number of bins, X- ensemble rule.
Ensure: quality of classification in the form of AUC - area
    under ROC curve
 1: divide the training set $TR$ into $K$ balanced bins
    $B_1, \dots, B_K$
    {MAP step}
 2: for $k = 1$ to $K$ do
 3:     build the classifier $CL_k$ for training data $B_k$
 4:     for $i = 1$ to $|TS|$ do
 5:         apply classifier $CL_k$ to $i{-}th$ instance $< d_i, cf_i > \in$
            $TS$
 6:         let $C_j^{ik}$ stand for the probability assigned by clas-
            sifier $CL_k$ to instance $d_i$ for class $c_j$, for $j = 1, 2$
 7:     end for
 8: end for
 9: apply team classifier
    {REDUCE step}
10: for $i = 1$ to $|TS|$ do
11:     using $C_j^{ik}$ for $j = 1, 2$, $k = 1, \dots, K$ and strategy X
        find class $l_i$
12: end for
13: calculate AUC - area under ROC curve
```

Figure 2. Algorithm SplitBal-parallel version

where sg is the sign function:

$$sg(x, y) = \begin{cases} 1 & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

- SumDistance

$$R_1 = \sum_{i=1}^{K} \frac{P_{i1}}{D_{i1} + 1}$$

$$R_2 = \sum_{i=1}^{K} \frac{P_{i2}}{D_{i2} + 1}$$

Finally, the instance r is classified as c_1 if $R_1 \geq R_2$, otherwise as c_2.

The details of the algorithm are given as Figure 2. Referring to experiments they are parametrized by:

- dataset with parameters: number of instances, number of attributes, imbalance ratio,
- the choice of a base classifier,
- the choice of ensemble rule: MaxDistance, MinDistance, ProDistance, MajDistance, SumDistance.

When performing the experiments, it was observed that Reduce step took definitely longer than Map. Therefore further speed-up was introduced by running in parallel computations performed in Reduce step. Besides, we also exploited in-memory data structure store, used as a database. This allowed to definitely decrease the processing time. To compare the processing times of single process SplitBal with parallel version using Map-Reduce paradigm with the mentioned speed-up of Reduce step, the latter version is further denoted as SplitBalPar.

3. Experiments and Datasets

To assess performance of the proposed approach we tested it over a set of publicly available benchmark datasets obtained from KEEL [7] and UCI [8]. Repository KEEL contains a special section containing 145 datasets for imbalanced classification. The data from UCI were artificially adapted to imbalanced format, by deleting randomly chosen instances from one class. Used datasets range from highly imbalanced (imbalance ratio IR=129.44) to low imbalanced (IR=6.02). The details on the datasets are given in Table 1. The table in each row contains the name of the data set, its origin, number of attributes, number of instances and imbalance ratio.

In the course of implementation the MapReduce paradigm was applied. MapReduce is proposed by Google as a programming model to process large and scalable datasets by parallel and distributed algorithms. The two major steps that is Map and Reduce are natural for the used SplitBal algorithm- the first used to process each bin with a map and the second to collect the results. Experiments were conducted in software environment including interpreter of R language and Java Platform as well as Mongo as a document database. Scripts in R were used for preprocessing task, dividing data into training and testing subsets as well as further distinguishing bins from testing data.

After some experiments with Naive Bayes, Logistic regression and C5.0 decision tree classifier, the latter was chosen. From R package we applied C50 as well as ROCR package for scoring the classifiers. For the sake of Map-Reduce paradigm the program in Java language was implemented. It precisely controls the number of processes dedicated for computations in both Map and Reduce steps. These numbers belong to the parameters of the computations. During the computations Java threads run and control operating system processes which perform computations as such, implemented as R language scripts. The Map step parallelization is straightforward and consists of performing independent computations for each bin. The Reduce step is a bit more complicated and relies on computing sums for pairs as they appear in the queue of already finished computations. The Java SE BlockingQueue class was used as a mechanism to synchronize the processes and their results. If the number of processes available to run is set to 1, then the computations will run sequentially, one after another. It allows to simulate well the sequential computations for SplitBal.

The experiments were conducted on 32 core processor machine and during the experiments approximately 15 cores were used (the number 15 was used as parameter of number of available threads/processes for both Map-Reduces steps). To measure the quality of classification on the benchmark datasets, AUC - the area under the ROC (Receiver Operating Characteristic) curve was used. This measure is recommended for the case of imbalanced datasets as ROC curve is

TABLE 1. BENCHMARK DATASETS USED IN THE EXPERIMENT

Dataset	Origin	Attributes	Instances	Imbalance ratio
yeast	[7]	8	1484	41.4
page block	[7]	10	5472	8.79
abalone19	[7]	8	4174	129.44
segment0	[7]	19	2308	6.02
magic	[8]	11	19020	61.35
bank	[8]	17	45211	75.47

TABLE 2. CLASSIFICATION RESULTS FOR SPLITBAL

Dataset	Ensemble rule	AUC area
yeast	MaxDistance	0.9699
page block	SumDistance	0.977
abalone19	SumDistance	0.825
segment0	SumDistance	0.998
magic	SumDistance	0.871
bank	SumDistance	0.7295

irrespective to class distribution and rather than calculating accuracy it makes use of the partition of the testing set into four subsets: TP - true positive, TN - true negative, FP - false positive and FN - false negative. The results of experiments are shown in Table 2. The table in each row contains the name of the data set, the name of the applied ensemble rule and the mean AUC value obtained 30 experiments. To confront the results of SplitBal we collect in Table 3 the outcome of SplitBal and EDBC. Here for SplitBal the ensemble rule SumDistance was used and the results differ with Table 2. It can be noticed that the improvement of parallel SplitBal over EDBC is substantial.

The comparison of processing times of the algorithm EDBC and single process version of SplitBal and parallel SplitBalPar is given in Table 4. The processing time for EDBC and big data sets was very long, for example for magic data it was longer than 12 hours and bank data longer than 24 hours, the results are omitted and marked $*$ in Table 3 and 4.

TABLE 3. CLASSIFICATION RESULTS- SPLITBAL VERSUS EDBC

Dataset	SplitBal AUC	EDBC AUC
yeast	0.925	0.86
page block	0.977	0.76
abalone19	0.825	0.65
segment0	0.998	0.86
magic	0.871	$*$
bank	0.895	$*$

TABLE 4. PROCESSING TIME

Dataset	EDBC	SplitBal	SplitBalPar
yeast	4m32	1m13s	0m21s
page block	1h15m8s	0m19s	0m8s
abalone19	46m18s	4m36s	0m59s
segment0	8m55s	0m14s	0m7s
magic	$*$	2m40	0m46s
bank	$*$	5m32s	1m10s

4. Conclusions and Future Work

The main contribution of the work is the design and implementation of an environment for classification experiments with imbalanced data sets. This environment is easily scalable and allows to further experiment with different classifiers, separate for bins, with different reduce rules and big data sets.

As for future developments we plan to introduce parallelization into EDBC algorithm (Figure 1). This will allow to experiment with bigger data sets. What is more, since EDBC is parametrized by the methods of feature selection and clustering it seems that further improvements are possible. So far the method is much inferior to SplitBal specially for the data sets with high imbalance ratio (e.g abalone19 data set).

In [4] it was observed that the preprocessing of imbalanced data performed before classification boosts the results, allows to reduce noisy data and decreases the training time. Future plans include further studies on SplitBal including feature selection and/or feature extraction of the original datasets. This is planned as a parallel implementation of Relief algorithm as advised by [9].

References

[1] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220 – 239, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416307175

[2] X. Zhang, Q. Song, G. Wang, K. Zhang, L. He, and X. Jia, "A dissimilarity-based imbalance data classification algorithm," *Appl. Intell.*, vol. 42, no. 3, pp. 544–565, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10489-014-0610-5

[3] X. Zhang, Y. Li, K. Ramamohanarao, L. Wu, Z. Tari, and M. Cheriet, "KRNN: k rare-class nearest neighbour classification," *Pattern Recognition*, vol. 62, pp. 33–44, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2016.08.023

[4] T. T. Nguyen, D. Hwang, and J. J. Jung, "Handling imbalanced classification problem: A case study on social media datasets," *Journal of Intelligent and Fuzzy Systems*, vol. 32, no. 2, pp. 1437–1448, 2017. [Online]. Available: http://dx.doi.org/10.3233/JIFS-169140

[5] P. Synowczyk, "Metody klasyfikacji danych slabo zrownowazonych (master thesis, in polish)," University of Gdansk, 2016.

[6] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2014.11.014

[7] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesús, S. Ventura, J. M. G. i Guiu, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: a software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00500-008-0323-y

[8] M. Lichman, "Uci machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml/

[9] J. Yazidi, W. Bouaguel, and N. Essoussi, “A parallel implementation of relief algorithm using mapreduce paradigm,” in *Computational Collective Intelligence - 8th International Conference, ICCCI 2016, Halkidiki, Greece, September 28-30, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, N. T. Nguyen, Y. Manolopoulos, L. S. Iliadis, and B. Trawinski, Eds., vol. 9876. Springer, 2016, pp. 418–425. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-45246-3_40