ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017, i6doc.com publ., ISBN 978-287587039-1. Available from http://www.i6doc.com/en/.

A multi-criteria meta-learning method to select under-sampling algorithms for imbalanced datasets

Romero F. A. B. Morais, Péricles B. C. Miranda and Ricardo M. A. Silva *

Universidade Federal de Pernambuco - CIn - Recife - Brazil

Abstract. Standard classifiers consider a balanced distribution of examples' classes in the data, thus, imbalanced datasets may hinder the learning process. Sampling techniques balance the data by adjusting the examples' classes distribution. However, selecting an appropriate sampling technique and its parameters for a given imbalanced dataset is still an open problem. This work proposes a method that uses Meta-Learning to recommend a technique for an imbalanced dataset considering multiple performance criteria. The experiments revealed that the proposal reached results comparable to those achieved by the brute-force approach, overcame the techniques with their default parameters most of the time, and always surpassed the random search approach.

1 Introduction

Common classifiers consider an equal distribution of classes in the data, a hypothesis that if not satisfied may prevent learning [1]. Given an imbalanced dataset, sampling techniques produce a new dataset where the distribution of examples in each class is more balanced. Under-sampling techniques have been largely used in the context of imbalanced datasets [1]. These techniques exclude examples from the majority class to balance both classes. Moreover, noisy and redundant instances when eliminated may also contribute to a more robust dataset [1]. The quality of the pre-processed dataset is dependent on the adequate choice of the sampling technique and its parameters. The task of selecting an adequate sampling technique and its parameters to pre-process an imbalanced dataset is still problematic, with few solutions existing for it. Initially, brute-force approaches evaluated all combinations of algorithms and their parameters to find the best solution, as performed in [2]. However, due to time restrictions, these alternatives are not always possible to be employed. Instead, Tong et al. [3] developed an analytical method to find an optimal configuration for a combination of the random over-sampling and the random under-sampling techniques. Despite an analytical solution was devised, the method still had to evaluate many configurations before selecting the most appropriate configuration.

Herein, we present a novel way of selecting a sampling algorithm and its parameters for an imbalanced dataset. The proposed method relies on Meta-Learning (MtL), which treats the sampling algorithm selection as a supervised learning task. The MtL recommends a solution (sampling technique) to a new problem (imbalanced dataset) based on stored solutions for previous problems (meta-data). Given a new imbalanced dataset, the most similar problem is retrieved from the meta-data and its solution is recommended. Once the knowledge is acquired by the MtL, algorithms can be recommended

*The authors would like to thank Motorola Mobility for funding this research.

for new problems without the need of evaluating other candidates again, as performed using search techniques [4]. Thus, MtL becomes a computationally cheaper alternative when compared to search methods. It is important to highlight that the application of the MtL method has not been investigated for the current problem. To evaluate the proposed method, seven well-known under-sampling algorithms were employed. In the experiments, 491 variations of the seven under-sampling algorithms were considered in the recommendation. The proposal was executed on 29 different classification problems, and the results revealed that the proposed method overcame the random search approach in all problems, surpassed the algorithms using their default parameters most of the time, and it was usually comparable to the brute-force solution.

Section 2 formalize the algorithm selection problem. Section 3 presents the proposed MtL method. Section 4 brings the experimental methodology and presents the obtained results. Section 5, draws the conclusions and discuss potential future works.

2 Sampling Algorithm Selection Problem

Here we present a formal definition of the algorithm selection problem, which can also be employed to sampling algorithm selection. Let us consider that we have a finite pool of parametrized algorithms A, where each algorithm $a_d \in A$ has a set of hyper-parameters, where $d = 1, ..., |A|$, and each of these hyper-parameters may take different configurations. A configuration of a specified algorithm is a single assignment of values to hyper-parameters (θ) among the possibly infinite set of all configurations Θ for that algorithm. Thus, we call $a_d(\theta)$, the algorithm a_d which is set with a possible configuration θ. When considering a problem to be solved by this algorithm, the set of possible problems is defined as I. The performance of each algorithm, when applied to a single problem instance $i \in I$, is represented as a performance vector where each element represents a criterion to be optimized. The goal of automatic selection is finding the best algorithm $a_d(\theta)^*$ for each $i \in I$, considering all the criteria.

3 Proposal

This work proposes a meta-learning method to select under-sampling algorithms for an input imbalanced problem. The proposed method works in two steps, meta-data generation and algorithm selection, as illustrated in Figure 1.

3.1 Meta-data generation

The meta-data generation step relies on a set of different imbalanced problems chosen to build a meta-data. For each input problem, two procedures are executed: algorithms evaluation and extraction of characteristics from problems.

Evaluation of candidate algorithms: We considered seven well-known under-sampling algorithms: Edited Nearest Neighbors (ENN or Wilson's Editing) [5], *k*-means Under-sampling (KMUS), Most Distant (MD) [6], Neighborhood Cleaning Rule (NCL), NearMiss-1 (NM1) [6], NearMiss-3 (NM3) [6] and Random Under-sampling

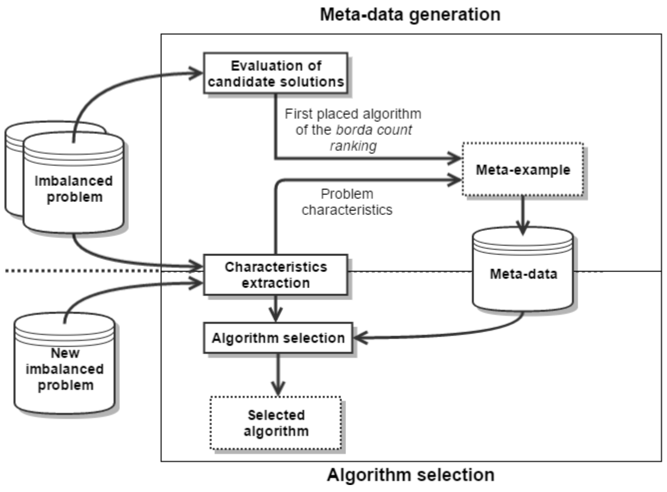

Fig. 1: Architecture of the proposed method.

(RUS). As each algorithm has hyper-parameters that can assume several possible values, the number of possible algorithm configurations becomes large. On one hand, a large pool of candidate algorithms is interesting because it may include the best possible algorithm. On the other hand, the computational cost to execute all experiments to build the meta-data may become prohibitive. Thus, we considered a set of algorithms with a limited range of values for each hyper-parameter (see Table 1). Thus, considering all algorithms and possible parameter variations, a total of 491 candidate algorithms are available to be recommended to a given imbalanced problem. It is worth mentioning that all parameters' ranges previously described were determined empirically. To assess the quality of a single algorithm configuration, we performed a 10-fold cross-validated resampling procedure repeated five times. For classification purposes, we adopted a Support Vector Machine (SVM) with a Gaussian Kernel where the C parameter value was set with value equals to 1 throughout all problems, while σ had a fixed value calculated for each problem following the heuristic determined in [7].

The performance of a classifier cannot be assessed considering only the accuracy, because it may be unrepresentative [1]. Therefore, we adopted multiple criteria to evaluate the classifier's performance: Accuracy (Acc), Area Under the ROC Curve (AUC), F_1 score (F_1), Specificity ($Spec$), and Negative Predictive Value (NPV). The values of each criterion range from 0 to 1, and the objective is to maximize each criterion. To choose one algorithm, from a set of possible candidates, considering all the performance criteria mentioned previously, we applied the Borda count method [8]. This method is a single-winner election method in which each algorithm is ranked by each criterion, and then an average rank is returned, where the algorithm in the first place is the winner.

Characteristics Extraction: It applies measures to describe the problem. In this work, we have used the meta-features described in [4]: *Number of examples*, *Ratio of the number of examples to the number of attributes*, *Proportion of the attributes with*

Table 1: Under-sampling techniques with their hyper-parameters and the range of values used in the experimentation.

Technique Name	**Hyper-parameters**	**Experimentation Values**
Edited Nearest Neighbors (ENN) [5]	k: Number of neighbors (default is $k = 3$).	k: [2, 3, . . ., 15]
k-means Under-sampling (KMUS)	k: Number of centroids (default value k is equals to the size of the minority class).	k: [1, 2, . . ., 5] $\times$ size of the minority class.
Most Distant (MD) [6]	k: Number of neighbors (default is $k = 3$). m: Percent of examples to select from the majority class (default value is $m = 0$).	k: [2, 3, . . ., 15]. m: [0, 10, . . ., 80].
Neighborhood Cleaning Rule (NCL)	k: Number of neighbors (default is $k = 3$).	k: [2, 3, . . ., 15].
NearMiss-1 (NM1) [6]	k: Number of neighbors (default is $k = 3$). m: Percent of examples to select from the majority class (default is $m = 0$).	k: [2, 3, . . ., 15]. m: [0, 10, . . ., 80].
NearMiss-3 (NM3) [6]	k: Number of neighbors (default is $k = 3$). p: Number of majority class examples to select from the k neighbors (default is $p = 2$).	k: [2, 3, . . ., 15]. p: [1, 2, . . ., 5].
Random Under-sampling (RUS)	m: Percent of examples to select from the majority class (default is $m = 0$).	m: [0, 10, . . ., 80].

outliers, *Coefficient of variation of the target*, *Sparsity of the target, Stationarity of the target*, *Average absolute correlation between numeric predictor attributes* and *Average dispersion gain*.

The composition of the best algorithm (output from the *Algorithms Evaluation*) and the characteristics of the input problem (provided by the *Characteristics Extraction*) represents a *meta-example* to be stored in the meta-data. Thus, algorithms used to solve previous problems can be reused for new problems according to their observed descriptive features.

3.2 Algorithm selection

Once the meta-data is built, a k-NN ranking method is used to provide under-sampling algorithms for a new imbalanced problem. Initially, the characteristics of the given problem are extracted. Next, the k nearest neighbors among the existing meta-examples are identified. Finally, the algorithm of each nearest neighbor is recommended to the new problem. To determine the k nearest neighbors, the distance between the meta-features of the new dataset and all the meta-examples is calculated, and the k closest datasets are selected. The distance function implemented was the Euclidean distance. It is worth mentioning that we used the k-NN with $k = 1$, where only one algorithm is recommended to the new problem. As it can be seen, once the meta-data is created, algorithms are recommended to new problems without the necessity of evaluating different candidate algorithms as performed using search techniques.

4 Experiments and results

Herein, we present the experiments that assessed the proposed solution on the set of 29 classification problems (see column one of the Table 2), totaling 29 meta-examples (one for each problem). The 29 datasets were selected from three different sources: the PROMISE repository [9], the KEEL-dataset repository [10], and the UCI Machine

Table 2: Average values of each one of the five criteria achieved by the approaches.

Problems	**Algorithms**		
	$Bruteforce$	$Proposal$	$Default$
Bank auth.	1.00, 1.00, 1.00, 1.00, 1.00	**1.00**, **1.00**, **1.00**, **1.00**, **1.00**	**1.00**, **1.00**, **1.00**, **1.00**, **1.00**
Blood Transf.	0.78, 0.73, 0.85, 0.54, 0.60	**0.78**, **0.73**, **0.85**, 0.53, **0.60**	0.68, 0.63, 0.74, 0.43, 0.49
Breast Cancer	0.94, 0.98, 0.94, 0.96, 0.93	**0.94**, **0.98**, **0.94**, **0.96**, **0.93**	0.84, 0.92, 0.90, 0.86, 0.91
cleveland-0_vs_4	0.96, 0.99, 0.98, 0.95, 0.78	0.94, **0.99**, **0.98**, **0.95**, 0.75	0.83, 0.90, 0.89, 0.91, 0.67
Diabetic Retin.	0.71, 0.78, 0.70, 0.78, 0.66	**0.71**, **0.78**, **0.70**, 0.77, **0.66**	0.70, 0.74, 0.65, 0.76, 0.63
ecoli1	0.90, 0.94, 0.93, 0.76, 0.82	**0.90**, **0.94**, **0.93**, **0.76**, **0.82**	0.83, 0.90, 0.89, 0.71, 0.73
glass1	0.75, 0.83, 0.80, 0.67, 0.67	**0.75**, **0.83**, **0.80**, **0.67**, 0.66	0.71, 0.80, 0.76, 0.50, 0.66
haberman	0.74, 0.71, 0.83, 0.41, 0.72	**0.74**, **0.71**, 0.82, **0.41**, **0.72**	0.70, 0.64, 0.75, 0.35, 0.70
iris	1.00, 1.00, 1.00, 1.00, 1.00	**1.00**, **1.00**, 0.99, **1.00**, **1.00**	0.99, **1.00**, 0.99, 0.97, **1.00**
JM1"	0.77, 0.65, 0.86, 0.30, 0.46	**0.77**, **0.65**, **0.86**, 0.29, **0.46**	0.70, 0.62, 0.84, 0.08, 0.45
KC3"	0.82, 0.76, 0.90, 0.55, 1.00	**0.82**, **0.76**, **0.90**, 0.46, **1.00**	0.76, 0.74, 0.88, 0.27, 0.67
MC1"	0.97, 0.84, 0.98, 0.30, 0.61	**0.97**, **0.84**, **0.98**, **0.30**, 0.60	0.95, 0.62, 0.95, 0.28, 0.48
MC2"	0.72, 0.77, 0.78, 0.59, 0.60	**0.72**, **0.77**, 0.75, **0.59**, 0.57	0.60, 0.71, 0.68, 0.31, 0.52
MW1"	0.90, 0.83, 0.94, 0.33, 0.60	**0.90**, **0.83**, 0.93, **0.33**, **0.60**	0.80, 0.70, 0.87, 0.25, 0.56
new-thyroid1	0.98, 1.00, 0.99, 0.97, 0.96	**0.98**, **1.00**, 0.98, 0.94, 0.94	0.87, 0.90, 0.92, 0.83, 0.91
page-blocks0	0.97, 0.98, 0.98, 0.94, 0.84	**0.97**, **0.98**, **0.98**, **0.94**, **0.84**	0.90, 0.91, 0.89, 0.82, 0.60
PC1"	0.86, 0.85, 0.92, 0.54, 0.36	**0.86**, **0.85**, **0.92**, **0.54**, **0.36**	0.79, **0.85**, 0.90, 0.47, 0.30
PC2"	0.97, 0.81, 0.98, 0.45, 0.49	**0.97**, **0.81**, 0.97, **0.45**, **0.49**	0.85, 0.74, 0.82, 0.12, 0.12
PC3"	0.87, 0.80, 0.93, 0.67, 0.60	0.86, **0.80**, 0.89, **0.67**, 0.59	0.72, 0.70, 0.74, 0.60, 0.51
PC4"	0.90, 0.90, 0.94, 0.80, 0.74	**0.90**, **0.90**, **0.94**, 0.78, **0.74**	0.78, 0.80, 0.83, 0.76, 0.41
PC5"	0.75, 0.75, 0.83, 0.47, 0.57	**0.75**, **0.75**, **0.83**, **0.47**, 0.56	0.65, 0.71, 0.73, 0.40, 0.52
pima	0.76, 0.82, 0.82, 0.74, 0.70	**0.76**, **0.82**, 0.80, **0.74**, 0.69	0.68, 0.71, 0.77, 0.51, 0.65
QSAR biodegradation	0.88, 0.93, 0.91, 0.84, 0.84	**0.88**, **0.93**, **0.91**, **0.84**, **0.84**	0.80, 0.85, 0.82, 0.81, 0.73
segment0	0.98, 0.99, 0.98, 0.99, 0.89	**0.98**, **0.99**, **0.98**, 0.98, 0.83	0.95, 0.99, 0.96, 0.95, 0.68
shuttle-c0-vs-c4	0.99, 1.00, 0.99, 0.96, 1.00	**0.99**, **1.00**, **0.99**, **0.96**, **1.00**	**0.99**, 0.99, **0.99**, **0.96**, **1.00**
vehicle2	0.98, 0.99, 0.99, 0.95, 0.97	**0.98**, **0.99**, **0.99**, **0.95**, **0.97**	0.92, 0.95, 0.91, 0.92, 0.96
vowel0	1.00, 1.00, 1.00, 1.00, 1.00	**1.00**, **1.00**, **1.00**, **1.00**, **1.00**	0.92, 0.91, 0.95, **1.00**, 0.98
winequality-red-4	0.77, 0.73, 0.87, 0.66, 0.28	**0.77**, **0.73**, 0.85, 0.64, 0.27	0.65, 0.69, 0.80, 0.65, 0.08
yeast1	0.76, 0.78, 0.84, 0.57, 0.63	**0.76**, **0.78**, 0.83, 0.55, **0.63**	0.72, 0.70, 0.77, 0.52, 0.50

Learning repository. All datasets were cleaned by first removing duplicated instances, followed by elimination of inconsistent instances.

The proposal was assessed by following a leave-one-out cross-validated (loocv) resampling experiment. At each step of the experiment, one meta-example was left out (considered as new problem) to evaluate the implemented prototype, and the remaining 28 meta-examples were kept in the meta-data to be selected by the MtL. Thus, the MtL provides, for the new problem, the best algorithm (first placed in the borda count average rank) used to solve the most similar (we used the k-NN with $k = 1$) meta-example considering the new problem. To evaluate the quality of the recommendation, the selected algorithm is executed on the new problem and all five criteria are calculated.

Table 2 presents the results (the average values of each one of the five criteria) achieved by the proposal and other three approaches in each problem. The criteria values are presented in Table 2 following the order Acc, AUC, F_1, $Spec$, NPV. The first column, from the *Algorithms* area, presents the results achieved by the best algorithms found by a brute-force approach for each problem. In other words, the results presented in this column represent the results achieved by the best algorithm for each problem. The second column presents the results achieved by the proposed approach. The third column, named *Default*, presents the results reached by the seven algorithms when configured with their default parameters. We highlight that the proposed method was also compared to a random search approach using 50 iterations. As the results reached by the random search approach were statistically worse than the results achieved by the proposal in all problems, we decided to remove these results from Table 2. The criteria values, presented in Table 2, which are in bold mean that the values are statistically equal to the results reached by the brute-force approach. As it can be seen in Table 2, the algorithms recommended by the proposal matched many results reached by the brute force approach and surpassed the algorithms using default parameters in most of the problems. According to the results, the task of providing an adequate under-sampling algorithm for an imbalanced problem can be solved using the proposed method. The proposal decreases the human interference on this task, automatically recommending a satisfactory algorithm for the new problem.

5 Conclusions

In this work, we proposed a method to automatically recommend a sampling technique for a given imbalanced dataset according to solutions that worked well for previous problems (the meta-data). The results of our experimentation showed that our method is comparable to the brute-force approach, overcame the sampling algorithms with their default parameters most of the time, and always surpassed the random search. Once the meta-data is built, the proposed method has the advantage of finding a good solution in linear-time (on the size of the meta-data). As future work, instead of recommending a single solution based on the most similar problem in the meta-data, the system could recommend a combination of techniques to be applied to a new problem.

References

[1] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

[2] David A Cieslak and Nitesh V Chawla. Start globally, optimize locally, predict globally: Improving performance on imbalanced data. In *Internat. Conf. on Data Mining*, pages 143–152. IEEE, 2008.

[3] Lee-Ing Tong, Yung-Chia Chang, and Shan-Hui Lin. Determining the optimal re-sampling strategy for a classification model with imbalanced data using design of experiments and response surface methodologies. *Expert Systems with Applic.*, 38(4):4222–4227, 2011.

[4] Petr Kuba, Pavel Brazdil, Carlos Soares, and Adam Woznica. Exploiting sampling and meta-learning for parameter setting for support vector machines. In *Proc. of VIII Iberoamerican Conf. on Artif. Intel.*, pages 209–216, 2002.

[5] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transac. on Systems, Man and Cybern.*, 2(3):408–421, 1972.

[6] Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proc. of Workshop on Learn. from Imbalanced Datasets*, 2003.

[7] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statist. Software*, 11(9):1–20, 2004.

[8] Duncan Black. Partial justification of the borda count. *Public Choice*, 28(1):1–15, 1976.

[9] The promise repository of empirical software engineering data, 2015.

[10] J Alcalá, A Fernández, J Luengo, J Derrac, S García, L Sánchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(255-287):11, 2010.