Racing for Unbalanced Methods Selection

Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi

Abstract

State-of-the-art classification algorithms suffer when the data is skewed towards one class. This led to the development of a number of techniques to cope with unbalanced data. However, no technique appears to work consistently better in all conditions. This paper presents a new R package, called **unbalanced**, which implements some well-known techniques for unbalanced classification tasks and provides a racing strategy to adaptively select the best methods for a given dataset, classification algorithms and accuracy measure adopted.

Keywords: R, unbalanced classification, Racing.

1. Introduction

Learning from unbalanced datasets is a difficult task since most classification algorithms are not designed to cope with a large difference between the number of cases belonging to different classes Dal Pozzolo, Caelen, Waterschoot, and Bontempi (2013). The unbalanced nature of the data is typical of many applications such as medical diagnosis, text classification and oil spills detection. Credit card fraud detection Dal Pozzolo, Caelen, Le Borgne, Waterschoot, and Bontempi (2014) is another well-known instance of highly unbalanced problem since (fortunately) the number of fraudulent transactions is typically much smaller than legitimate ones. In literature several methods for dealing with unbalanced datasets have been proposed.

Since in real large variate tasks it is hard to know a priori the nature of the unbalanced tasks, the user is recommended to test all techniques with a consequent high computational cost. Under different conditions, such as distinct datasets and algorithms, the best methods may change. In this context we propose a racing strategy Maron and Moore (1993) to automatically select the most adequate technique for a given dataset. The rationale of the racing strategy consists in testing in parallel a set of alternative balancing strategies on a subset of the dataset and to remove progressively the alternatives which are significantly worse.

By adopting a racing strategy we are able to select in an efficient manner either the best balancing method or a method which is not significantly different from the best one Dal Pozzolo *et al.* (2013). Moreover, racing is able to reduce consistently the computation needed before finding the right methods for the dataset.

2. Methods for unbalanced classification

The **unbalanced** package implements some of the most well-known sampling and distance-based methods for unbalanced classification task. Within the family of sampling methods, we

have functions for random undersampling (`ubUnder`) and oversampling (`ubOver`) Drummond and Holte (2003). The package contains also a function called `ubSMOTE` that implements SMOTE Chawla, Bowyer, Hall, and Kegelmeyer (2002). Other distance-based methods available in **unbalanced** are OSS Kubat, Matwin *et al.* (1997) (`ubOSS`), CNN Hart (1968) (`ubCNN`), ENN Wilson (1972) (`ubENN`), NCL Laurikkala (2001) (`ubNCL`) and Tomek Link Tomek (1976) (`ubTomek`). All these methods can be called by a wrapper function `ubBalance` that allows testing all these strategies by simpling changing the argument `type`.

The package includes the `ubIonosphere` datasets, which is a modification of the Ionosphere dataset contained in **mlbench** package. It has only numerical input variables, i.e. the first two variables are removed. The *Class* variable, originally taking values *bad* and *good*, has been transformed into a factor where 1 denotes the minority (bad) and 0 the majority class (good). This variable is our target and it is in the last column of the dataset. In the following we will also called the minority class as positive and the majority as negative.

For example, let's apply oversampling to the Ionosphere dataset to have a balanced dataset.

```
library(unbalanced)
data(ubIonosphere)
n <- ncol(ubIonosphere)
output <- ubIonosphere[ ,n]
input <- ubIonosphere[ ,-n]

set.seed(1234)
#apply oversampling
data <- ubBalance(X=input, Y=output, type="ubOver", k=0)
#oversampled dataset
overData <- data.frame(data$X, Class=data$Y)
#check the frequency of the target variable after oversampling
summary(overData$Class)

##   0   1
## 225 225
```

In this case we replicate the minority class until we have as many positive as negative instances. Alternativelly, we can balance the dataset using undersampling (i.e. removing observations from the majority class):

```
#apply undersampling
data <- ubBalance(X=input, Y=output, type="ubUnder", perc=50,  method="percPos")
#undersampled dataset
underData <- data.frame(data$X, Class=data$Y)
#check the frequency of the target variable after oversampling
summary(underData$Class)

##   0   1
## 126 126
```

Another well-know method for unbalanced distribution is SMOTE, which oversample the minority class by creating new synthetic observations. Let's compare the performances of two **randomForest** classifiers, one trained on the original unbalanced dataset and a second trained on a dataset obtained after applying SMOTE.

```
set.seed(1234)

#keep half for training and half for testing
N <- nrow(ubIonosphere)
N.tr <- floor(0.5*N)
id.tr <- sample(1:N, N.tr)
id.ts <- setdiff(1:N, id.tr)
X.tr  <- input[id.tr, ]
Y.tr <- output[id.tr]
X.ts <- input[id.ts, ]
Y.ts <- output[id.ts]

unbalTrain <- data.frame(X.tr, Class=Y.tr)
summary(unbalTrain$Class)

##   0   1
## 111  64

library(randomForest)
#use the original unbalanced training set to build a model
model1 <- randomForest(Class ~ ., unbalTrain)
#predict on the testing set
preds <- predict(model1, X.ts, type="class")
confusionMatrix1 <- table(prediction=preds, actual=Y.ts)
print(confusionMatrix1)

##           actual
## prediction   0   1
##          0 111   5
##          1   3  57

#rebalance the training set before building a model
balanced <- ubBalance(X=X.tr, Y=Y.tr, type="ubSMOTE", percOver=200, percUnder=150)
balTrain <- data.frame(balanced$X, Class=balanced$Y)
summary(balTrain$Class)

##   0   1
## 192 192

#use the balanced training set
model2 <- randomForest(Class ~ ., balTrain)
```

```
#predict on the testing set
preds <- predict(model2, X.ts, type="class")
confusionMatrix2 <- table(prediction=preds, actual=Y.ts)
print(confusionMatrix2)

##           actual
## prediction   0   1
##          0 103   1
##          1  11  61

#we can now correctly classify more minority class instances
```

Using SMOTE we alter the original class distribution and we are able to increase the number of minority instances correctly classified. After smoting the dataset we have fewer false negatives, but a larger number of false positives. In unbalanced classification, it often desired to correctly classify all minority instances (reducing the number of false negatives), because the cost of missing a positive instances (a false negative) is much higher than the cost of missing a negative instance (a false positive).

3. Racing for strategy selection

The variety of approaches available in the **unbalanced** package allows the user to test multiple unbalanced methods. In a real situation where we have no prior information about the data distribution, it is difficult to decide which unbalanced strategy to use. In this case testing all alternatives is not an option either because of the associated computational cost.

A possible solution comes from the adoption of the Racing approach which was proposed in Maron and Moore (1993) to perform efficiently model selection in a learning task. The principle of Racing consists in testing in parallel a set of alternatives and using a statistical test to determine if an alternative is significantly worse than the others. In that case such alternative is discarded from the competition, and the computational effort is devoted to differentiate the remaining ones. The *F-race* version was proposed in Birattari, Stützle, Paquete, and Varrentrapp (2002) and combines the Friedman test with Hoeffding Races Maron and Moore (1993) to eliminate inferior candidates as soon as enough statistical evidence arises against them. In F-race, the Friedman test is used to check whether there is evidence that at least one of the candidates is significantly different from others and post-tests are applied to eliminate those candidates that are significantly worse than the best one.

Here we adopt F-Race to search efficiently for the best strategy for unbalanced data. The candidates are assessed on different subsets of data and, each time a new assessment is made, the Friedman test is used to dismiss significantly inferior candidates. We used a 10 fold cross validation to provide the assessment measure to the race. If a candidate is significantly better than all the others than the race is terminated without the need of using the whole dataset. In case there is not evidence of worse/better methods, the race terminates when the entire dataset is explored and the best candidate is the one with the best average result. F-Race is available in **unbalanced** with the `ubRacing` function and its implementation is a modification of the `race` function available in the **race** package. The function `ubRacing` compares the

8 unbalanced methods (**ubUnder**, **ubOver**, **ubSMOTE**, **ubOSS**, **ubCNN**, **ubENN**, **ubNCL**, **ubTomek**) against the unbalanced distribution, so we have 9 candidates starting the race.

In the following we will use a highly unbalanced dataset containing credit card transactions used in Dal Pozzolo, Caelen, Johnson, and Bontempi (2015) and available here: http://www.ulb.ac.be/di/map/adalpozz/data/creditcard.Rdata.

```
set.seed(1234)

# load the dataset
load(url("http://www.ulb.ac.be/di/map/adalpozz/data/creditcard.Rdata"))

#configuration of the sampling method used in the race
ubConf <- list(percOver=200, percUnder=200,
               k=2, perc=50, method="percPos", w=NULL)

# Race with 10 trees in the Random Forest to speed up results
results <- ubRacing(Class ~., creditcard, "randomForest", positive=1,
                    metric="auc", ubConf=ubConf, ntree=10)

##
##  Racing for unbalanced methods selection in 10 fold CV
##  Number of candidates.....................................9
##  Max number of folds in the CV.............................10
##  Max number of experiments................................100
##  Statistical test.................................Friedman test
##
##                               Markers:
##                                  x No test is performed.
##                                  - The test is performed and
##                                    some candidates are discarded.
##                                  = The test is performed but
##                                    no candidate is discarded.
##
##  +-+-----------+-----------+-----------+-----------+-----------+
##  | |       Fold|      Alive|       Best|  Mean best| Exp so far|
##  +-+-----------+-----------+-----------+-----------+-----------+
##  |x|          1|          9|          4|     0.9541|          9|
##  |=|          2|          9|          4|      0.954|         18|
##  |-|          3|          2|          4|     0.9591|         27|
##  |=|          4|          2|          4|      0.963|         29|
##  |=|          5|          2|          4|     0.9651|         31|
##  |-|          6|          1|          4|     0.9646|         33|
##  +-+-----------+-----------+-----------+-----------+-----------+
## Selected candidate: ubSMOTE   metric: auc   mean value: 0.9646

# Race using 4 cores and 500 trees (default number of trees in randomForest)
```

```
# results <- ubRacing(Class ~., creditcard, "randomForest", positive=1,
#                     metric="auc", ubConf=ubConf, ncore=4)


# Let's try with a different algorithm (see mlr package for supported packages)
# library(e1071)
# results <- ubRacing(Class ~., creditcard, "svm", positive=1, ubConf=ubConf)
# library(rpart)
# results <- ubRacing(Class ~., creditcard, "rpart", positive=1, ubConf=ubConf)
```

The best method according to the F-race is SMOTE. Please note that it is possible to change the type of statistical test used to remove candidates in the race with the argument `stat.test`. When we set `stat.test = "no"`, no statistical test is performed and the race terminates when all the folds of the cross validation are explored.

4. Conclusion

With the **unbalanced** package we have made available some of the most well-known methods for unbalanced distribution. All these methods can be called from `ubBalance` that is a wrapper to the method-specific functions. Depending on the type of dataset, classification algorithm and accuracy measure adopted, we may have different strategies that return the best accuracy.

This consideration has lead us to adopt the F-race strategy where different candidates (unbalanced methods) are tested simultaneously. This algorithm is implemented in the `ubRacing` function which selects the best candidate without having to explore the whole dataset.

References

Birattari M, Stützle T, Paquete L, Varrentrapp K (2002). "A racing algorithm for configuring metaheuristics." In *Proceedings of the genetic and evolutionary computation conference*, pp. 11–18.

Chawla N, Bowyer K, Hall LO, Kegelmeyer WP (2002). "SMOTE: synthetic minority oversampling technique." *Journal of Artificial Intelligence Research (JAIR)*, **16**, 321–357.

Dal Pozzolo A, Caelen O, Johnson R, Bontempi G (2015). "Using calibrated probability with undersampling." In *2015 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE.

Dal Pozzolo A, Caelen O, Le Borgne YA, Waterschoot S, Bontempi G (2014). "Learned lessons in credit card fraud detection from a practitioner perspective." *Expert Systems with Applications*, **41**(10), 4915–4928.

Dal Pozzolo A, Caelen O, Waterschoot S, Bontempi G (2013). "Racing for unbalanced methods selection." In *Proceedings of the 14th International Conference on Intelligent Data Engineering and Automated Learning*. IDEAL.

Drummond C, Holte R (2003). "C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling." In *Workshop on Learning from Imbalanced Datasets II*. Citeseer.

Hart PE (1968). "The Condensed Nearest Neighbor Rule." *IEEE Transactions on Information Theory*.

Kubat M, Matwin S, *et al.* (1997). "Addressing the curse of imbalanced training sets: one-sided selection." In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pp. 179–186. MORGAN KAUFMANN PUBLISHERS, INC.

Laurikkala J (2001). "Improving identification of difficult small classes by balancing class distribution." *Artificial Intelligence in Medicine*, pp. 63–66.

Maron O, Moore A (1993). "Hoeffding races: Accelerating model selection search for classification and function approximation." *Robotics Institute*, p. 263.

Tomek I (1976). "Two modifications of CNN." *IEEE Trans. Syst. Man Cybern.*, **6**, 769–772.

Wilson D (1972). "Asymptotic properties of nearest neighbor rules using edited data." *Systems, Man and Cybernetics*, (3), 408–421.

Affiliation:

Andrea Dal Pozzolo, Gianluca Bontempi
Machine Learning Group (MLG),
Computer Science Department,
Faculty of Sciences ULB,
Université Libre de Bruxelles,
Brussels, Belgium
E-mail: adalpozz@ulb.ac.be, gbonte@ulb.ac.be

Olivier Caelen,
Fraud Risk Management Analytics,
Worldline S.A.,
Brussels, Belgium
E-mail: olivier.caelen@worldline.com