Package 'unbalanced'

June 26, 2015

Type Package

Title Racing for Unbalanced Methods Selection

Version 2.0

Date 2015-06-25

Author Andrea Dal Pozzolo, Olivier Caelen and Gianluca Bontempi

Maintainer Andrea Dal Pozzolo <adalpozz@ulb.ac.be>

Description A dataset is said to be unbalanced when the class of interest (minority class) is much rarer than normal behaviour (majority class). The cost of missing a minority class is typically much higher that missing a majority class. Most learning systems are not prepared to cope with unbalanced data and several techniques have been proposed. This package implements some of most well-known techniques and propose a racing algorithm to select adaptively the most appropriate strategy for a given unbalanced task.

License GPL (>= 3)

URL http://mlg.ulb.ac.be

Depends mlr, foreach, doParallel

Imports FNN, RANN

Suggests randomForest, ROCR

NeedsCompilation no

Repository CRAN

Date/Publication 2015-06-26 13:34:37

R topics documented:

`unbalanced-package` *Racing for Unbalanced Methods Selection*

Description

A dataset is said to be unbalanced when the class of interest (minority class) is much rarer than normal behaviour (majority class). The cost of missing a minority class is typically much higher that missing a majority class. Most learning systems are not prepared to cope with unbalanced data and several techniques have been proposed to rebalance the classes. This package implements some of most well-known techniques and propose a racing algorithm [2] to select adaptively the most appropriate strategy for a given unbalanced task [1].

Details

Package:	unbalanced
Type:	Package
Version:	2.0
Date:	2015-06-17
License:	GPL (>= 3)

Author(s)

Andrea Dal Pozzolo <adalpozz@ulb.ac.be>, Olivier Caelen <olivier.caelen@worldline.com> and Gianluca Bontempi <gbonte@ulb.ac.be>

Maintainer: Andrea Dal Pozzolo

Andrea Dal Pozzolo and Gianluca Bontempi are with the Machine Learning Group, Computer Science Department, Faculty of Sciences ULB, Universite Libre de Bruxelles, Brussels, Belgium.
Olivier Caelen is with the Fraud Risk Management Analytics, Worldline, Belgium.
The work of Andrea Dal Pozzolo is supported by the Doctiris scholarship of Innoviris, Belgium.

References

1. Dal Pozzolo, Andrea, et al. "Racing for unbalanced methods selection." Intelligent Data Engineering and Automated Learning - IDEAL 2013. Springer Berlin Heidelberg, 2013. 24-31.
2. Birattari, Mauro, et al. "A Racing Algorithm for Configuring Metaheuristics."GECCO. Vol. 2. 2002.

See Also

ubBalance, ubRacing

Examples

```
#use Racing to select the best technique for an unbalanced dataset
library(unbalanced)
data(ubIonosphere)

#configure sampling parameters
ubConf <- list(type="ubUnder", percOver=200, percUnder=200, k=2, perc=50, method="percPos", w=NULL)

#load the classification algorithm that you intend to use inside the Race
#see 'mlr' package for supported algorithms
library(randomForest)
#use only 5 trees
results <- ubRacing(Class ~., ubIonosphere, "randomForest", positive=1, ubConf=ubConf, ntree=5)

# try with 500 trees
# results <- ubRacing(Class ~., ubIonosphere, "randomForest", positive=1, ubConf=ubConf, ntree=500)
# let's try with a different algorithm
# library(e1071)
# results <- ubRacing(Class ~., ubIonosphere, "svm", positive=1, ubConf=ubConf)
# library(rpart)
# results <- ubRacing(Class ~., ubIonosphere, "rpart", positive=1, ubConf=ubConf)
```

---

ubBalance *Balance wrapper*

Description

The function implements several techniques to re-balance or remove noisy instances in unbalanced datasets.

Usage

```
ubBalance(X, Y, type="ubSMOTE", positive=1, percOver=200, percUnder=200,
          k=5, perc=50, method="percPos", w=NULL, verbose=FALSE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset.
type	the balancing technique to use (ubOver, ubUnder, ubSMOTE, ubOSS, ubCNN, ubENN, ubNCL, ubTomek).
positive	the majority class of the response variable.
percOver	parameter used in ubSMOTE

percUnder	parameter used in ubSMOTE
k	parameter used in ubOver, ubSMOTE, ubCNN, ubENN, ubNCL
perc	parameter used in ubUnder
method	parameter used in ubUnder
w	parameter used in ubUnder
verbose	print extra information (TRUE/FALSE)

Details

The argument type can take the following values: "ubOver" (over-sampling), "ubUnder" (under-sampling), "ubSMOTE" (SMOTE), "ubOSS" (One Side Selection), "ubCNN" (Condensed Nearest Neighbor), "ubENN" (Edited Nearest Neighbor), "ubNCL" (Neighborhood Cleaning Rule), "ubTomek" (Tomek Link).

Value

The function returns a list:

X	input variables
Y	response variable
id.rm	index of instances removed if availble in the technique selected

References

Dal Pozzolo, Andrea, et al. "Racing for unbalanced methods selection." Intelligent Data Engineering and Automated Learning - IDEAL 2013. Springer Berlin Heidelberg, 2013. 24-31.

See Also

ubRacing, ubOver, ubUnder, ubSMOTE, ubOSS, ubCNN, ubENN, ubNCL, ubTomek

Examples

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

#balance the dataset
data<-ubBalance(X= input, Y=output, type="ubSMOTE", percOver=300, percUnder=150, verbose=TRUE)
balancedData<-cbind(data$X,data$Y)
```

ubCNN *Condensed Nearest Neighbor*

Description

Condensed Nearest Neighbor selects the subset of instances that are able to correctly classifing the original datasets using a one-nearest neighbor rule.

Usage

```
ubCNN(X, Y, k = 1, verbose = T)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
k	the number of neighbours to use
verbose	print extra information (TRUE/FALSE)

Details

In order to compute nearest neighbors, only numeric features are allowed.

Value

The function returns a list:

X	input variables
Y	response variable

References

P. E. Hart. The condensed nearest neighbor rule. IEEE Transactions on Informa- tion Theory, 1968.

See Also

ubBalance

Examples

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

data<-ubCNN(X=input, Y= output)
```

```
newData<-cbind(data$X, data$Y)
```

---

ubENN *Edited Nearest Neighbor*

Description

Edited Nearest Neighbor removes any example whose class label differs from the class of at least two of its three nearest neighbors.

Usage

```
ubENN(X, Y, k = 3, verbose = TRUE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
k	the number of neighbours to use
verbose	print extra information (TRUE/FALSE)

Details

In order to compute nearest neighbors, only numeric features are allowed.

Value

The function returns a list:

X	input variables
Y	response variable

References

D. Wilson. Asymptotic properties of nearest neighbor rules using edited data. Systems, Man and Cybernetics, IEEE Transactions on, 408-421, 1972.

See Also

ubBalance

Examples

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

data<-ubENN(X=input, Y= output)
newData<-cbind(data$X, data$Y)
```

---

ubIonosphere *Ionosphere dataset*

Description

The datasets is a modifcation of Ionosphere dataset cotained in "mlbench" package. It contains only numerical input variables, i.e. the first two variables are removed. The Class variable orginally taking values bad and good has been transformed into a factor where 1 denotes bad and 0 good.

Usage

```
data(ubIonosphere)
```

Format

A data frame with 351 observations on 33 independent variables (all numerical) and one last defining the class (1 or 0).

Source

http://cran.r-project.org/package=mlbench

Examples

```
data(ubIonosphere)
summary(ubIonosphere)
```

ubNCL *Neighborhood Cleaning Rule*

Description

Neighborhood Cleaning Rule modifies the Edited Nearest Neighbor method by increasing the role of data cleaning. Firstly, NCL removes negatives examples which are misclassified by their 3-nearest neighbors. Secondly, the neighbors of each positive examples are found and the ones belonging to the majority class are removed.

Usage

```
ubNCL(X, Y, k = 3, verbose = TRUE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
k	the number of neighbours to use
verbose	print extra information (TRUE/FALSE)

Details

In order to compute nearest neighbors, only numeric features are allowed.

Value

The function returns a list:

X	input variables
Y	response variable

References

J. Laurikkala. Improving identification of difficult small classes by balancing class distribution. Artificial Intelligence in Medicine, pages 63-66, 2001.

See Also

ubBalance

Examples

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

data<-ubNCL(X=input, Y= output)
newData<-cbind(data$X, data$Y)
```

---

ubOSS *One Side Selection*

Description

One Side Selection is an undersampling method resulting from the application of Tomek links followed by the application of Condensed Nearest Neighbor.

Usage

```
ubOSS(X, Y, verbose = TRUE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
verbose	print extra information (TRUE/FALSE)

Details

In order to compute nearest neighbors, only numeric features are allowed.

Value

The function returns a list:

X	input variables
Y	response variable

References

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

See Also

ubBalance

Examples

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

data<-ubOSS(X=input, Y= output)
newData<-cbind(data$X, data$Y)
```

---

ubOver *Over-sampling*

Description

The function replicates randomly some instances from the minority class in order to obtain a final dataset with the same number of instances from the two classes.

Usage

```
ubOver(X, Y, k = 0, verbose=TRUE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
k	defines the sampling method.
verbose	print extra information (TRUE/FALSE)

Details

If K=0: sample with replacement from the minority class until we have the same number of instances in each class. If K>0: sample with replacement from the minority class until we have k-times the orginal number of minority instances.

Value

The function returns a list:

X	input variables
Y	response variable

See Also

ubBalance

Examples

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

data<-ubOver(X=input, Y= output)
newData<-cbind(data$X, data$Y)
```

---

ubRacing *Racing*

Description

The function implementes the Racing algorithm [2] for selecting the best technique to re-balance or remove noisy instances in unbalanced datasets [1].

Usage

```
ubRacing(formula, data, algo, positive=1, ncore=1, nFold=10, maxFold=10, maxExp=100,
          stat.test="friedman", metric="f1", ubConf, verbose=FALSE, ...)
```

Arguments

formula	formula describing the model to be fitted.
data	the unbalanced dataset
algo	the classification algorithm to use with the mlr package.
positive	label of the positive (minority) class.
ncore	the number of core to use in the Race. Race is performed with parallel exectuion when ncore > 1.
nFold	number of folds in the cross-validation that provides the subset of data to the Race
maxFold	maximum number of folds to use in the Race
maxExp	maximum number of experiments to use in the Race
stat.test	statistical test to use to remove candidates which perform significantly worse than the best.
metric	metric used to asses the classification.
ubConf	configuration of the balancing techniques used in the Race.
verbose	print extra information (TRUE/FALSE)
...	additional arguments pass to train function in mlr package.

Details

The argument metric can take the following values: "gmean", "f1" (F-score or F-measure), "auc" (Area Under ROC curve). Argument stat.test defines the statistical test used to remove candidates during the race. It can take the following values: "friedman" (Friedman test), "t.bonferroni" (t-test with bonferroni correction), "t.holm" (t-test with holm correction), "t.none" (t-test without correction), "no" (no test, the Race continues until new subsets of data are provided by the cross validation). Argument balanceConf is a list passed to function ubBalance that is used for configuration.

Value

The function returns a list:

`Race`	matrix containing accuracy results for each technique in the Race.
`best`	best technique selected in the Race.
`avg`	average of the metric used in the Race for the technique selected.
`sd`	standard deviation of the metric used in the Race for the technique selected.
`N.test`	number of experiments used in the Race.
`Gain`	% of computational gain with resepct to the maximum number of experiments given by the cross validation.

Note

The function ubRacing is a modified version of the race function availble in the race package: http://cran.r-project.org/package=race.

References

1. Dal Pozzolo, Andrea, et al. "Racing for unbalanced methods selection." Intelligent Data Engineering and Automated Learning - IDEAL 2013. Springer Berlin Heidelberg, 2013. 24-31.
2. Birattari, Mauro, et al. "A Racing Algorithm for Configuring Metaheuristics."GECCO. Vol. 2. 2002.

See Also

ubBalance, ubOver, ubUnder, ubSMOTE, ubOSS, ubCNN, ubENN, ubNCL, ubTomek

Examples

```
#use Racing to select the best technique for an unbalanced dataset
library(unbalanced)
data(ubIonosphere)

#configure sampling parameters
ubConf <- list(type="ubUnder", percOver=200, percUnder=200, k=2, perc=50, method="percPos", w=NULL)

#load the classification algorithm that you intend to use inside the Race
#see 'mlr' package for supported algorithms
library(randomForest)
#use only 5 trees
```

```
results <- ubRacing(Class ~., ubIonosphere, "randomForest", positive=1, ubConf=ubConf, ntree=5)

# try with 500 trees
# results <- ubRacing(Class ~., ubIonosphere, "randomForest", positive=1, ubConf=ubConf, ntree=500)
# let's try with a different algorithm
# library(e1071)
# results <- ubRacing(Class ~., ubIonosphere, "svm", positive=1, ubConf=ubConf)
# library(rpart)
# results <- ubRacing(Class ~., ubIonosphere, "rpart", positive=1, ubConf=ubConf)
```

---

ubSMOTE *SMOTE*

Description

Function that implements SMOTE (synthetic minority over-sampling technique)

Usage

```
ubSMOTE(X, Y, perc.over = 200, k = 5, perc.under = 200, verbose = TRUE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
perc.over	per.over/100 is the number of new instances generated for each rare instance. If perc.over < 100 a single instance is generated.
k	the number of neighbours to consider as the pool from where the new examples are generated
perc.under	perc.under/100 is the number of "normal" (majority class) instances that are randomly selected for each smoted observation.
verbose	print extra information (TRUE/FALSE)

Details

Y must be a factor.

Value

The function returns a list:

X	input variables
Y	response variable

Note

Original code from DMwR package

References

Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." arXiv preprint arXiv:1106.1813 (2011).

See Also

ubBalance

Examples

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

data<-ubSMOTE(X=input, Y= output)
newData<-cbind(data$X, data$Y)
```

---

ubSmoteExs *ubSmoteExs*

Description

Function used in SMOTE to generate new minority examples.

Usage

```
ubSmoteExs(data, tgt, N = 200, k = 5)
```

Arguments

data	the data.frame
tgt	the index of the target/response variables
N	N/100 is the number of new instances generated for each rare instance. If N < 100 a single instance is generated
k	the number of neighbours to consider as the pool from where the new examples are generated

Details

This function does not handle vectors

Value

newCases

See Also

ubSMOTE

---

ubTomek *Tomek Link*

Description

The function finds the points in the dataset that are tomek link using 1-NN and then removes only majority class instances that are tomek links.

Usage

```
ubTomek(X, Y, verbose = TRUE)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
verbose	print extra information (TRUE/FALSE)

Details

In order to compute nearest neighbors, only numeric features are allowed.

Value

The function returns a list:

X	input variables
Y	response variable
id.rm	index of instances removed

References

I. Tomek. Two modifications of cnn. IEEE Trans. Syst. Man Cybern., 6:769-772, 1976.

See Also

ubBalance

Examples

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

data<-ubTomek(X=input, Y= output)
newData<-cbind(data$X, data$Y)
```

---

ubUnder *Under-sampling*

Description

The function removes randomly some instances from the majority (negative) class and keeps all instances in the minority (positive) class in order to obtain a more balanced dataset. It allows two ways to perform undersampling: i) by setting the percentage of positives wanted after undersampling (percPos method), ii) by setting the sampling rate on the negatives, (percUnder method). For percPos, "perc"has to be (N.1/N * 100) <= perc <= 50, where N.1 is the number of positive and N the total number of instances. For percUnder, "perc"has to be (N.1/N.0 * 100) <= perc <= 100, where N.1 is the number of positive and N.0 the number of negative instances.

Usage

```
ubUnder(X, Y, perc = 50, method = "percPos", w = NULL)
```

Arguments

X	the input variables of the unbalanced dataset.
Y	the response variable of the unbalanced dataset. It must be a binary factor where the majority class is coded as 0 and the minority as 1.
perc	percentage of sampling.
method	method to perform under sampling ("percPos", "percUnder").
w	weights used for sampling the majority class, if NULL all majority instances are sampled with equal weights

Value

The function returns a list:

X	input variables
Y	response variable
id.rm	index of instances removed

See Also

ubBalance

Examples

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

data<-ubUnder(X=input, Y= output, perc = 40,  method = "percPos")
newData<-cbind(data$X, data$Y)
```

Index