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 than 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 .................................................. 2
  ubBalance ............................................................... 3
  ubCNN ................................................................. 5
  ubENN ................................................................. 6
  ubIonosphere .......................................................... 7
  ubNCL ................................................................. 8
  ubOSS ................................................................. 9
  ubOver ............................................................... 10
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 than 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

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
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).

The function returns a list:

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

References


See Also

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

Examples

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

Usage

ubCNN(X, Y, k = 1, 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


See Also

ubbalance

Examples

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

data<-ubCNN(X=input, Y=output)
Edited Nearest Neighbor

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

Usage

\[
\text{ubENN}(X, Y, k = 3, \text{verbose} = \text{TRUE})
\]

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


See Also

**ubBalance**
Examples

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

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

---

**ubIonosphere**  
*Ionosphere dataset*

**Description**

The dataset is a modification of Ionosphere dataset contained in “mlbench” package. It contains 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 bad and 0 good.

**Usage**

```r
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](http://cran.r-project.org/package=mlbench)

**Examples**

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


See Also

ubBalance
**Examples**

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

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**

```r
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**

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

```r
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 original number of minority instances.

Value

The function returns a list:

- `X`: input variables
- `Y`: response variable
**ubRacing**

**See Also**

ubBalance

**Examples**

```r
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)
```

**Description**

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

**Usage**

```r
ubRacing(formula, data, algo, positive=1, ncore=1, nfold=10, maxfold=10, maxexp=100, statNtest="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 execution 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 assess 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 available in the race package:

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

References


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
ubSMOTE

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>the input variables of the unbalanced dataset.</td>
</tr>
<tr>
<td>Y</td>
<td>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.</td>
</tr>
<tr>
<td>perc.over</td>
<td>per.over/100 is the number of new instances generated for each rare instance. If perc.over &lt; 100 a single instance is generated.</td>
</tr>
<tr>
<td>k</td>
<td>the number of neighbours to consider as the pool from where the new examples are generated</td>
</tr>
<tr>
<td>perc.under</td>
<td>perc.under/100 is the number of &quot;normal&quot; (majority class) instances that are randomly selected for each smoted observation.</td>
</tr>
<tr>
<td>verbose</td>
<td>print extra information (TRUE/FALSE)</td>
</tr>
</tbody>
</table>

Details

Y must be a factor.

Value

The function returns a list:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>input variables</td>
</tr>
<tr>
<td>Y</td>
<td>response variable</td>
</tr>
</tbody>
</table>

Note

Original code from DMwR package
References


See Also

ubBalance

Examples

library(unbalanced)
data(ubIonosphere)n<-ncol(ubIonosphere)output<-ubIonosphere$Classinput<-ubIonosphere[, -n]data<-ubSMOTE(X=input, Y = output)newData<-cbind(data$X, data$Y)

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
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

Examples

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

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

---

**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 \((N.1/N * 100) \leq perc \leq 50\), where \(N.1\) is the number of positive and \(N\) the total number of instances. For percUnder, "perc" has to be \((N.1/N.0 * 100) \leq perc \leq 100\), where \(N.1\) is the number of positive and \(N.0\) the number of negative instances.

**Usage**

```r
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
**ubUnder**

**See Also**

`ubBalance`

**Examples**

```r
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

*Topic datasets
   ubIonosphere, 7

*Topic unbalanced datasets, imbalanced learning
   unbalanced-package, 2
   ubBalance, 3, 5, 6, 8, 10–12, 14, 15, 17
   ubCNN, 4, 5, 12
   ubENN, 4, 6, 12
   ubIonosphere, 7
   ubNCL, 4, 8, 12
   ubOSS, 4, 9, 12
   ubOver, 4, 10, 12
   ubRacing, 3, 4, 11
   ubSMOTE, 4, 12, 13, 15
   ubSmoteExs, 14
   ubTomek, 4, 12, 15
   ubUnder, 4, 12, 16
   unbalanced (unbalanced-package), 2
   unbalanced-package, 2

18