Package ‘hyperSMURF’

April 30, 2018

Type Package
Title Hyper-Ensemble Smote Undersampled Random Forests
Version 2.0
Date 2018-04-28
Author Giorgio Valentini [aut, cre] - AnacletoLab, Dipartimento di Informatica, Universita' degli Studi di Milano;
Max Schubach [ctb] - Charite, Universitatsmedizin Berlin;
Matteo Re [ctb] - AnacletoLab, Dipartimento di Informatica, Universita' degli Studi di Milano;
Peter N Robinson [ctb] - The Jackson Laboratory for Genomic Medicine, Farmington CT, USA.
Maintainer Giorgio Valentini <valentini@di.unimi.it>
Description Machine learning supervised method to learn rare genomic features in imbalanced genetic data sets. This method can be also applied to classify or rank examples characterized by a high imbalance between the minority and majority class. hyperSMURF adopts a hyper-ensemble (ensemble of ensembles) approach, undersampling of the majority class and oversampling of the minority class to learn highly imbalanced data.
License GPL (>= 2)
LazyLoad yes
Imports unbalanced, randomForest
NeedsCompilation no
Repository CRAN
Date/Publication 2018-04-29 22:19:15 UTC

R topics documented:

hyperSMURF-package .................. 2
do.random.partition .................. 3
do.stratified.cv.data ................ 4
do.stratified.cv.data.from.folds .... 5
hyperSMURF.cv ........................ 6
hyperSMURF.test ..................... 7
hyperSMURF.test.thresh .............. 8
hyperSMURF.train ................... 9
Description

Machine learning supervised method to learn rare genomic features in imbalanced genetic data sets. This method can be also applied to classify or rank examples characterized by a high imbalance between the minority and majority class. hyperSMURF adopts a hyper-ensemble (ensemble of ensembles) approach, undersampling of the majority class and oversampling of the minority class to learn highly imbalanced data.

Details

The DESCRIPTION file:

Package: hyperSMURF
Type: Package
Title: Hyper-Ensemble Smote Undersampled Random Forests
Version: 2.0
Date: 2018-04-28
Author: Giorgio Valentini [aut, cre] - AnacletoLab, Dipartimento di Informatica, Universita' degli Studi di Milano; Max ... degli Studi di Milano; Peter N Robinson [ctb] - The Jackson Laboratory for Genomic Medicine, Farmington CT, USA.
Maintainer: Giorgio Valentini <valentini@di.unimi.it>
Description: Machine learning supervised method to learn rare genomic features in imbalanced genetic data sets. This method can be also applied to classify or rank examples characterized by a high imbalance between the minority and majority class. hyperSMURF adopts a hyper-ensemble (ensemble of ensembles) approach, undersampling of the majority class and oversampling of the minority class to learn highly imbalanced data.
License: GPL (>= 2)
LazyLoad: yes
Imports: unbalanced, randomForest
NeedsCompilation: no

Index of help topics:

do.random.partition Random partition of the data
do.stratified.cv.data Construction of random folds for cross-validation
do.stratified.cv.data.from.folds Construction of folds for cross-validation from predefined folds
hyperSMURF-package Hyper-Ensemble Smote Undersampled Random Forests
hyperSMURF.cv hyperSMURF cross-validation
hyperSMURF.test Test of a hyperSMURF model
hyperSMURF.test.thresh
do.random.partition

Test of a thresholded hyperSMURF model

hyperSMURF.train hyperSMURF training
imbalanced.data.generator Synthetic imbalanced data generator
smote SMOTE oversampling
smote_and_undersample SMOTE oversampling and undersampling

Author(s)
Giorgio Valentini [aut, cre] - AnacletoLab, Dipartimento di Informatica, Universita’ degli Studi di Milano; Max Schubach [ctb] - Charite, Universitatsmedizin Berlin; Matteo Re [ctb] - Anacleto-Lab, Dipartimento di Informatica, Universita’ degli Studi di Milano; Peter N Robinson [ctb] - The Jackson Laboratory for Genomic Medicine, Farmington CT, USA.
Maintainer: Giorgio Valentini <valentini@di.unimi.it>

References

do.random.partition  Random partition of the data

Description
Performs a random partition of the indices that refer to a given data set (data frame or matrix)

Usage
do.random.partition(n.ex, n.partitions, seed = 0)

Arguments
n.ex  number of indices to be partitioned
n.partitions  number of partitions
seed  seed for the random generator

Details
The partition of the data is performed using only the indices of the data not the data itself

Value
a list with n.partitions elements. Each element stores the indices of the partition.

Examples
do.random.partition(100, 10)
do.stratified.cv.data  Construction of random folds for cross-validation

Description
The function randomly generates fold data for cross-validation

Usage
do.stratified.cv.data(examples, positives, k = 10, seed = 0)

Arguments
examples  vector of integer: indices of the examples
positives  vector of integer: Indices of the positive examples. The indices refer to the
           indices of examples
k          number of folds (def = 10)
seed       seed of the random generator (def=0). If is set to 0 no initialization is performed

Details
The folds are separated for positive and negative examples. The elements included in each fold are
obtained by random sampling the data.

Value
a list with two components;
fold.non.positives  a list with k components. Each component is a vector with the indices of the
                   non positive elements of the fold
old.positives      a list with k components. Each component is a vector with the indices of the
                   positive elements of the fold

See Also
  do.stratified.cv.data.from.folds

Examples
do.stratified.cv.data(1:100, 1:10, k = 10)
do.stratified.cv.data.from.folds

Construction of folds for cross-validation from predefined folds

Description

The function generates data for cross-validation from pre-computed folds

Usage

do.stratified.cv.data.from.folds(examples, positives, folds, k = 10)

Arguments

examples
  vector of integer: indices of the examples
positives
  vector of integer: Indices of the positive examples. The indices refer to the indices of examples
folds
  vector of indices: its length is equal to examples, with values in the interval [0, kk). The value indicates the partition in the cross validation step of the class
k
  number of folds (def = 10)

Details

The folds are separated for positive and negative examples. The elements included in each fold are obtained from the vector of fold indices folds.

Value

a list with two components;
  fold.non.positives
    a list with k components. Each component is a vector with the indices of the non positive elements of the fold
  fold.positives
    a list with k components. Each component is a vector with the indices of the positive elements of the fold

See Also

  do.stratified.cv.data

Examples

do.stratified.cv.data.from.folds(1:100, 1:10, folds=sample(rep(0:4),20), k = 5)
Description

Automated cross validation of hyperSMURF (hyper-ensemble SMote Undersampled Random Forests)

Usage

hyperSMURF.cv(data, y, kk = 5, n.part = 10, fp = 1, ratio = 1,
               k = 5, ntree = 10, mtry = 5, cutoff = c(0.5, 0.5), thresh = FALSE,
               seed = 0, fold.partition = NULL, file = "")

Arguments

data a data frame or matrix with the data

y a factor with the labels. 0: majority class, 1: minority class.

kk number of folds (def: 5)

n.part number of partitions (def. 10)

fp multiplicative factor for the SMOTE oversampling of the minority class. If fp<1 no oversampling is performed.

cutoff ratio of the #majority/#minority

k number of the nearest neighbours for SMOTE oversampling (def. 5)

ntree number of the features to randomly selected by the decision tree of each base random forest (def. 10)

mtry cutoff a numeric vector of length 2. Cutoff for respectively the majority and minority class. This parameter is meaningful when used with the thresholded version of hyperSMURF parameter (thresh=TRUE)

thresh logical. If TRUE the thresholded version of hyperSMURF is executed (def: FALSE)

seed initialization seed for the random generator. If set to 0 (def.) no initialization is performed

fold.partition vector of size nrow(data) with values in interval [0, kk). The values indicate the fold of the cross validation of each example. If NULL (default) the folds are randomly generated.

file name of the file where the cross-validated hyperSMURF models will be saved. If file="" (def.) no model is saved.
Details

The cross-validation is performed by randomly constructing the folds (parameter `fold.partition = NULL`) or using a set of predefined folds listed in the parameter `fold.partition`. The cross validation is performed by training and testing in sequence the base random forests. More precisely for each training set constructed at each step of the cross validation a separated random forest is trained sequentially for each of the `n.part` partitions of the data, by oversampling the minority class (parameter `fp`) and undersampling the majority class (parameter `ratio`). The random forest parameters `ntree` and `mtry` are the same for all the random forest of the hyper-ensemble.

Value

A vector with the cross-validated hyperSMURF probabilities (hyperSMURF scores).

References


See Also

`hyperSMURF.train`, `hyperSMURF.test`

Examples

d <- imbalanced.data.generator(n.pos=10, n.neg=300, sd=0.3);
res <- hyperSMURF.cv (d$data, d$labels, kk=2, n.part=3, fp=1, ratio=1, k=3, ntree=7, mtry=2, seed = 1, fold.partition=NULL);

Test of a hyperSMURF model

Description

A hyperSMURF model is tested on a given data set. Predictions of each RF of the hyperensemble are performed sequentially and the scores of each ensemble are finally averaged.

Usage

`hyperSMURF.test(data, HSmodel)`

Arguments

data a data frame or matrix with the test data. Rows: examples; columns: features

HSmodel a list including the trained random forest models. The models have been trained with `hyperSMURF.train`
Value

a named vector with the computed probabilities for each example (hyperSMURF score)

References


See Also

hyperSMURF.train

Examples

```r
train <- imbalanced.data.generator(n.pos=20, n.neg=1000,
   n.features=10, n.inf.features=2, sd=0.1, seed=1);
hsmodel <- hyperSMURF.train(train$data, train$label,
   n.part = 5, fp = 1, ratio = 2, k = 5);
test <- imbalanced.data.generator(n.pos=20, n.neg=1000,
   n.features=10, n.inf.features=2, sd=0.1, seed=2);
res <- hyperSMURF.test(test$data, HSmodel);
y <- ifelse(test$labels==1,1,0);
pred <- ifelse(res>0.5,1,0);
table(pred,y);
```
Value

a named vector with the computed probabilities for each example (HyperSMURF thresholded score)

See Also

hyperSMURF.test, hyperSMURF.train

Examples

```r
train <- imbalanced.data.generator(n.pos=20, n.neg=500,
                                  n.features=10, n.inf.features=2, sd=0.1, seed=1);
HSmodel <- hyperSMURF.train(train$data, train$label, n.part = 5,
                           fp = 1, ratio = 2, k = 5, cutoff=c(0.3, 0.7));
test <- imbalanced.data.generator(n.pos=20, n.neg=500,
                                   n.features=10, n.inf.features=2, sd=0.1, seed=2);
res <- hyperSMURF.test.thresh(test$data, HSmodel);
```

Description

A hyperSMURF model is trained on a given data set. Training data are partitioned, and each RF is separately trained on each partition by SMOTE oversampling of the positives (minority class examples) and undersampling of the negatives (majority class examples). Each RF is trained sequentially.

Usage

```r
hyperSMURF.train(data, y, n.part = 10, fp = 1, ratio = 1, k = 5, ntree = 10,
                   mtry = 5, cutoff = c(0.5, 0.5), seed = 0, file = "")
```

Arguments

data a data frame or matrix with the train data. Rows: examples; columns: features
y a factor with the labels. 0: majority class, 1: minority class.
n.part number of partitions (def. 10)
fp multiplicative factor for the SMOTE oversampling of the minority class. If fp<1 no oversampling is performed.
ratio ratio of the #majority/#minority
k number of the nearest neighbors for SMOTE oversampling (def. 5)
tree number of trees of the base learner random forest (def. 10)
mtry number of the features to randomly selected by the decision tree of each base random forest (def. 5)
cutoff a numeric vector of length 2. Cutoff for respectively the majority and minority class. This parameter is meaningful when used with the thresholded version of hyperSMURF (parameter thresh=TRUE)
seed

initialization seed for the random generator. If set to 0 (def.) no initialization is performed.

file

name of the file where the cross-validated hyperSMURF models will be saved. If file="" (def.) no model is saved.

Details

A different random forest is trained on each partition of the training set. If npos and nneg are the number of respectively the positive and negative examples, for each partition of the training data fp*npos new synthetic positives constructed by the SMOTE algorithm are added to the training set. The number of negatives is set to ratio*(fp*npos + npos). If no enough negatives are available in the partition, then all the negatives in the partition are used to train the base RF associated to the partition.

Value

A list of trained RF models. Each element of the list is a randomForest objects of the homonymous package.

References


See Also

hyperSMURF.test

Examples

```r
train <- imbalanced.data.generator(n.pos=20, n.neg=1000,
    n.features=10, n.inf.features=2, sd=1, seed=1);
HSmodel <- hyperSMURF.train(train$data, train$label, n.part = 5, fp = 1, ratio = 2);
```

Usage

imbalanced.data.generator(n.pos=100, n.neg=2000, 
n.features=10, n.inf.features=2, sd=1, seed=0)

Arguments

  n.pos    number of positive (minority class) examples (def. 100)
  n.neg    number of negative (majority class) examples (def. 2000)
  n.features total number of features (def. 10)
  n.inf.features number of informative features (def. 2)
  sd        standard deviation of the informative features (def.1)
  seed      initialization seed for the random number generator. If 0 (def) current clock
time is used.

Value

A list with two elements:

data    the matrix of the synthetic data having pos+n.neg rows and n.features columns
labels  a factor with the labels of the examples: 1 for minority and 0 for majority class

Examples

imbalanced.data.generator(n.pos=10, n.neg=200, n.features=6, n.inf.features=2, sd=1)

smote                   SMOTE oversampling

Description

Function to oversample by SMOTE the minority class

Usage

smote(data, fp = 1, k = 5)

Arguments

data    data frame or matrix of data including only the minority class. Rows: examples; 
columns: features
fp       multiplicative factor for the SMOTE oversampling of the minority class (def=1). 
          If fp<1 no oversampling is performed.
k        number of the nearest neighbours for SMOTE oversampling (def. 5)
Details

If \( n \) is the number of examples of the minority class, then \( fp \cdot n \) new synthetic examples are generated according to the SMOTE algorithm and returned in addition to the original set of positives. If \( fp < 1 \) no new data are generated and the original data set is returned.

Value

a data frame including the original minority class examples plus the SMOTE oversampled data

See Also

smote_and_undersample

Examples

d <- imbalanced.data.generator(n.pos=20, n.neg=1000, n.features=12, n.inf.features=2, sd=1, seed=1);
res <- smote(d$data[d$label==1,], fp = 2, k = 3);

smote_and_undersample

SMOTE oversampling and undersampling

Description

Function to both oversample by SMOTE the minority class and undersample the majority class

Usage

smote_and_undersample(data, y, fp = 1, ratio = 1, k = 5)

Arguments

data a data frame or matrix. Rows: examples; columns: features
y a factor with the labels. 0: majority class, 1: minority class.
fp multiplicative factor for the SMOTE oversampling of the minority class. If \( fp < 1 \) no oversampling is performed.
ratio ratio of the #majority/#minority
k number of the nearest neighbours for SMOTE oversampling (def. 5)

Details

If \( n \) is the number of examples of the minority class, then \( fp \cdot n \) new synthetic examples are generated according to the SMOTE algorithm and \( \text{ratio} \cdot (fp \cdot n + n) \) negative examples are undersampled form the majority class.
Value

A list with two entries:

- **X**  
  a data frame including the original minority class examples plus the SMOTE oversampled and undersampled data

- **Y**  
  a factor with the labels of the data frame

See Also

`smote`

Examples

d <- imbalanced.data.generator(n.pos=20, n.neg=1000, n.features=12, n.inf.features=2, sd=1, seed=1);
res <- smote_and_undersample(d$data, d$label, fp = 2, ratio = 3);
Index

do.random.partition, 3
do.stratified.cv.data, 4, 5
do.stratified.cv.data.from.folds, 4, 5

hypermURF (hypermURF-package), 2
hypermURF-package, 2
hypermURF.cv, 6
hypermURF.test, 7, 7, 9, 10
hypermURF.test.thresh, 8
hypermURF.train, 7–9, 9

imbalanced.data.generator, 10

smote, 11, 13
smote_and_undersample, 12, 12