Package ‘adabag’

October 14, 2015

Type Package
Title Applies Multiclass AdaBoost.M1, SAMME and Bagging
Version 4.1
Date 2015-10-14
Author Alfaro, Esteban; Gamez, Matias and Garcia, Noelia; with contributions from Li Guo
Maintainer Esteban Alfaro <Esteban.Alfaro@uclm.es>
Depends rpart, mlbench, caret
Description It implements Freund and Schapire's Adaboost.M1 algorithm and Breiman's Bagging
algorithm using classification trees as individual classifiers. Once these classifiers have been
trained, they can be used to predict on new data. Also, cross validation estimation of the error can
be done. Since version 2.0 the function margins() is available to calculate the margins for these
classifiers. Also a higher flexibility is achieved giving access to the rpart.control() argument
of 'rpart'. Four important new features were introduced on version 3.0, AdaBoost-SAMME (Zhu
et al., 2009) is implemented and a new function errorevol() shows the error of the ensembles as
a function of the number of iterations. In addition, the ensembles can be pruned using the option
'newmfinal' in the predict.bagging() and predict.boosting() functions and the posterior probabil-
ity of
each class for observations can be obtained. Version 3.1 modifies the relative importance measure
to take into account the gain of the Gini index given by a variable in each tree and the weights of
these trees. Version 4.0 includes the margin-based ordered aggregation for Bagging pruning (Guo
and Boukir, 2013) and a function to auto prune the 'rpart' tree. Moreover, three new plots are also
available importanceplot(), plot.errorevol() and plot.margins(). Version 4.1 allows to predict on
unlabeled data.
License GPL (>= 2)
LazyLoad yes
NeedsCompilation no
Repository CRAN
Date/Publication 2015-10-14 23:41:31

R topics documented:

   adabag-package ......................................................... 2
adabag-package

Description

It implements Freund and Schapire’s Adaboost.M1 algorithm and Breiman’s Bagging algorithm using classification trees as individual classifiers. Once these classifiers have been trained, they can be used to predict on new data. Also, cross validation estimation of the error can be done. Since version 2.0 the function margins() is available to calculate the margins for these classifiers. Also a higher flexibility is achieved giving access to the rpart.control() argument of ‘rpart’. Four important new features were introduced on version 3.0, AdaBoost-SAMME (Zhu et al., 2009) is implemented and a new function errorevol() shows the error of the ensembles as a function of the number of iterations. In addition, the ensembles can be pruned using the option ‘newmfinal’ in the predict.bagging() and predict.boosting() functions and the posterior probability of each class for observations can be obtained. Version 3.1 modifies the relative importance measure to take into account the gain of the Gini index given by a variable in each tree and the weights of these trees. Version 4.0 includes the margin-based ordered aggregation for Bagging pruning (Guo and Boukir, 2013) and a function to auto prune the ‘rpart’ tree. Moreover, three new plots are also available importanceplot(), plot.errorevol() and plot.margins(). Version 4.1 allows to predict on unlabeled data.

Details

Package: adabag
Type: Package
Version: 4.1
Date: 2015-10-14
License: GPL(>= 2)
LazyLoad: yes
Author(s)

Author: Esteban Alfaro-Cortes, Matias Gamez-Martinez and Noelia Garcia-Rubio, with contributions from Li Guo
Maintainer: Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>

References


Reverse cites: To the best of our knowledge this package has been cited by:


If you know any other work where this package is cited, please send us an email

See Also

autoprune, bagging, bagging.cv, boosting, boosting.cv, errorevol, importanceplot, margins, MarginOrderedPruning, Bagging.plot, errorevol.plot, margins, predict.bagging, predict.boosting

Examples

```r
## rpart library should be loaded
data(iris)
iris.adaboost <- boosting(Species~., data=iris, boos=TRUE,
mfinal=6)
importanceplot(iris.adaboost)

sub <- c(sample(1:50, 35), sample(51:100, 35), sample(101:150, 35))
iris.bagging <- bagging(Species~., data=iris[sub,], mfinal=10)
#Predicting with labeled data
iris.predbagging<- predict.bagging(iris.bagging, newdata=iris[-sub,])
iris.predbagging

#Predicting with unlabeled data
iris.predbagging<- predict.bagging(iris.bagging, newdata=iris[-sub,-5])
iris.predbagging
```

autoprune Build automatically a pruned tree of class rpart

Description

Builds automatically a pruned tree of class rpart looking in the cptable for the minimum cross validation error plus a standard deviation

Usage

```r
autoprune(formula, data, subset=1:length(data[,1]), ...)
```
Arguments

- **formula**: a formula, as in the `lm` function.
- **data**: a data frame in which to interpret the variables named in the formula.
- **subset**: optional expression saying that only a subset of the rows of the data should be used in the fit, as in the `rpart` function.
- **...**: further arguments passed to or from other methods.

Details

The cross validation estimation of the error (xerror) has a random component. To avoid this randomness the 1-SE rule (or 1-SD rule) selects the simplest model with a xerror equal or less than the minimum xerror plus the standard deviation of the minimum xerror.

Value

An object of class `rpart`

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

References


See Also

`rpart`

Examples

```r
## rpart library should be loaded
library(rpart)
data(iris)
iris.prune<-autopruner(Species~., data=iris)
iris.prune

## Comparing the test error of rpart and autopruner
library(mlbench)
data(BreastCancer)
l <- length(BreastCancer[,1])
sub <- sample(1:l,2*l/3)

BC.rpart <- rpart(Class~.,data=BreastCancer[sub,-1],cp=1, maxdepth=5)
BC.rpart.pred <- predict(BC.rpart,newdata=BreastCancer[-sub,-1],type="class")
```
Applies the Bagging algorithm to a data set

Description

Fits the Bagging algorithm proposed by Breiman in 1996 using classification trees as single classifiers.

Usage

```
bagging(formula, data, mfinal = 100, control, ...)
```

Arguments

- `formula`: a formula, as in the `lm` function.
- `data`: a data frame in which to interpret the variables named in the `formula`.
- `mfinal`: an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal = 100` iterations.
- `control`: options that control details of the `rpart` algorithm. See `rpart.control` for more details.
- `...`: further arguments passed to or from other methods.

Details

Unlike boosting, individual classifiers are independent among them in bagging.

Value

An object of class `bagging`, which is a list with the following components:

- `formula`: the formula used.
- `trees`: the trees grown along the iterations.
votes a matrix describing, for each observation, the number of trees that assigned it to each class.

prob a matrix describing, for each observation, the posterior probability or degree of support of each class. These probabilities are calculated using the proportion of votes in the final ensemble.

class the class predicted by the ensemble classifier.
samples the bootstrap samples used along the iterations.
importance returns the relative importance of each variable in the classification task. This measure takes into account the gain of the Gini index given by a variable in each tree.

Author(s)
Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

References

See Also
predict.bagging, bagging.cv

Examples
```r
# rpart library should be loaded
#This example has been hidden to fulfill execution time <5s
library(rpart)
data(iris)
iris.bagging <- bagging(Species ~ , data = iris, mfinal = 10)

# Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
l <- length(Vehicle[,1])
sub <- sample(1:1,2*l/3)
Vehicle.bagging <- bagging(Class ~ , data = Vehicle[sub,], mfinal = 15,
control = rpart.control(maxdepth = 5, msplit = 15))
#Using the pruning option
Vehicle.bagging.pred <- predict.bagging(Vehicle.bagging, newdata = Vehicle[-sub,], newmfinal = 10)
Vehicle.bagging.pred$confusion
Vehicle.bagging.pred$error
```
Description

The data are divided into \( v \) non-overlapping subsets of roughly equal size. Then, bagging is applied on \((v-1)\) of the subsets. Finally, predictions are made for the left out subsets, and the process is repeated for each of the \( v \) subsets.

Usage

\[
\text{bagging.cv}(\text{formula, data, } v = 10, m\text{final} = 100, \text{control})
\]

Arguments

- formula: a formula, as in the \texttt{lm} function.
- data: a data frame in which to interpret the variables named in formula.
- \( v \): An integer, specifying the type of \( v \)-fold cross validation. Defaults to 10. If \( v \) is set as the number of observations, leave-one-out cross validation is carried out. Besides this, every value between two and the number of observations is valid and means that roughly every \( v \)-th observation is left out.
- mfinal: an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to \( m\text{final}=100 \) iterations.
- control: options that control details of the \texttt{rpart} algorithm. See \texttt{rpart.control} for more details.

Value

An object of class \texttt{bagging.cv}, which is a list with the following components:

- class: the class predicted by the ensemble classifier.
- confusion: the confusion matrix which compares the real class with the predicted one.
- error: returns the average error.

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>
References


See Also

`bagging`, `predict.bagging`

Examples

```r
## rpart library should be loaded
library(rpart)
data(iris)
iris.baggingcv <- bagging.cv(Species ~ ., data=iris, mfinal=10, control=rpart.control(cp=0.01))
iris.baggingcv[-1]

## rpart and mlbench libraries should be loaded
## Data Vehicle (four classes)
#This example has been hidden to keep execution time <5s
#data(Vehicle)
#Vehicle.bagging.cv <- bagging.cv(Class ~ ., data=Vehicle,v=5,mfinal=10,
#control=rpart.control(maxdepth=5))
#Vehicle.bagging.cv[-1]
```

---

**boosting**

*Applies the AdaBoost.M1 and SAMME algorithms to a data set*

**Description**

Fits the AdaBoost.M1 (Freund and Schapire, 1996) and SAMME (Zhu et al., 2009) algorithms using classification trees as single classifiers.

**Usage**

```r
boosting(formula, data, boos = TRUE, mfinal = 100, coeflearn = 'Breiman', control, ...)
```
Arguments

formula
  a formula, as in the `lm` function.
data
  a data frame in which to interpret the variables named in `formula`.
boos
  if TRUE (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If FALSE, every observation is used with its weights.
mfinal
  an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal=100` iterations.
coeflearn
  if 'Breiman' (by default), \( \alpha = \frac{1}{2} \ln \left( \frac{1 - err}{err} \right) \) is used. If 'Freund' \( \alpha = \ln \left( \frac{1 - err}{err} \right) \) is used. In both cases the AdaBoost.M1 algorithm is used and \( \alpha \) is the weight updating coefficient. On the other hand, if `coeflearn` is 'Zhu' the SAMME algorithm is implemented with \( \alpha = \ln \left( \frac{1 - err}{err} \right) + \ln \left( \frac{n \text{classes} - 1}{\text{classes}} \right) \).
control
  options that control details of the rpart algorithm. See `rpart.control` for more details.
...
  further arguments passed to or from other methods.

Details

AdaBoost.M1 and SAMME are simple generalizations of AdaBoost for more than two classes. In AdaBoost-SAMME the individual trees are required to have an error lower than \( 1 - \frac{1}{\text{classes}} \) instead of \( \frac{1}{2} \) of the AdaBoost.M1

Value

An object of class `boosting`, which is a list with the following components:

formula
  the formula used.
trees
  the trees grown along the iterations.
weights
  a vector with the weighting of the trees of all iterations.
votes
  a matrix describing, for each observation, the number of trees that assigned it to each class, weighting each tree by its \( \alpha \) coefficient.
prob
  a matrix describing, for each observation, the posterior probability or degree of support of each class. These probabilities are calculated using the proportion of votes in the final ensemble.
class
  the class predicted by the ensemble classifier.
importance
  returns the relative importance of each variable in the classification task. This measure takes into account the gain of the Gini index given by a variable in a tree and the weight of this tree.

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>
References


See Also

predict.boosting, boosting.cv

Examples

```r
## rpart library should be loaded
data(iris)
iris.adaboost <- boosting(Species~., data=iris, boos=TRUE, mfinal=5)
iris.adaboost

## Data Vehicle (four classes)
data(Vehicle)
1 <- length(Vehicle[,1])
sub <- sample(1:1,2*1/3)
mfinal <- 10
maxdepth <- 5
Vehicle.rpart <- rpart(Class~.,data=Vehicle[sub,],maxdepth=maxdepth)
Vehicle.rpart.pred <- predict(Vehicle.rpart,newdata=Vehicle[-sub, ],type="class")
tb <- table(Vehicle.rpart.pred,Vehicle$Class[-sub])
error.rpart <- 1-(sum(diag(tb))/sum(tb))
tb
tb

Vehicle.adaboost <- boosting(Class~.,data=Vehicle[sub, ],mfinal=mfinal, coeflearn="Zhu",
control=rpart.control(maxdepth=maxdepth))
Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost,newdata=Vehicle[-sub, ])
Vehicle.adaboost.pred$Confusion
Vehicle.adaboost.pred$Error

#comparing error evolution in training and test set
errorevol(Vehicle.adaboost,newdata=Vehicle[sub, ])->evol.train
errorevol(Vehicle.adaboost,newdata=Vehicle[-sub, ])->evol.test
plot.errorevol(evol.test, evol.train)
```
boosting.cv

Runs v-fold cross validation with AdaBoost.M1 or SAMME.

Description

The data are divided into \( v \) non-overlapping subsets of roughly equal size. Then, boosting is applied on \((v-1)\) of the subsets. Finally, predictions are made for the left out subsets, and the process is repeated for each of the \( v \) subsets.

Usage

boosting.cv(formula, data, v = 10, boos = TRUE, mfinal = 100, coeflearn = "Breiman", control)

Arguments

- **formula** a formula, as in the `lm` function.
- **data** a data frame in which to interpret the variables named in `formula`.
- **boos** if TRUE (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If FALSE, every observation is used with its weights.
- **v** An integer, specifying the type of v-fold cross validation. Defaults to 10. If \( v \) is set as the number of observations, leave-one-out cross validation is carried out. Besides this, every value between two and the number of observations is valid and means that roughly every \( v \)-th observation is left out.
- **mfinal** an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to \( mfinal = 100 \) iterations.
- **coeflearn** if 'Breiman'(by default), \( \alpha = 1/2 \ln((1-err)/err) \) is used. If 'Freund' \( \alpha = \ln((1-err)/err) \) is used. In both cases the AdaBoost.M1 algorithm is used and \( \alpha \) is the weight updating coefficient. On the other hand, if coeflearn is 'Zhu' the SAMME algorithm is implemented with \( \alpha = \ln((1-err)/err) + \ln(nclasses-1) \).
- **control** options that control details of the rpart algorithm. See rpart.control for more details.

Value

An object of class boosting.cv, which is a list with the following components:

- **class** the class predicted by the ensemble classifier.
- **confusion** the confusion matrix which compares the real class with the predicted one.
- **error** returns the average error.
Author(s)

Esteban Alfaro-Cortes <esteban.alfaro@uclm.es>, Matias Gamez-Martinez <matias.gamez@uclm.es> and Noelia Garcia-Rubio <noelia.garcia@uclm.es>

References


See Also

boosting, predict.boosting

Examples

```r
## rpart library should be loaded
data(iris)
iris.boostcv <- boosting.cv(Species ~ ., v=2, data=iris, mfinal=10,
control=rpart.control(cp=0.01))
iris.boostcv[-1]

## rpart and mlbench libraries should be loaded
## Data Vehicle (four classes)
## This example has been hidden to fulfill execution time <5s
#data(Vehicle)
#Vehicle.boost.cv <- boosting.cv(Class ~ ., data=Vehicle,v=5, mfinal=10, coeflearn="Zhu",
#control=rpart.control(maxdepth=5))
#Vehicle.boost.cv[-1]
```

errorevol

Shows the error evolution of the ensemble

Description

Calculates the error evolution of an AdaBoost.M1, AdaBoost-SAMME or Bagging classifier for a data frame as the ensemble size grows
Usage

```r
errorevol(object, newdata)
```

Arguments

- **object**: This object must be the output of one of the functions `bagging` or `boosting`. This is assumed to be the result of some function that produces an object with two components named `formula` and `trees`, as those returned for instance by the `bagging` function.
- **newdata**: Could be the same data frame used in `object` or a new one.

Details

This can be useful to see how fast `bagging`, `boosting` reduce the error of the ensemble. In addition, it can detect the presence of overfitting and, therefore, the convenience of pruning the ensemble using `predictNbagging` or `predictNboosting`.

Value

An object of class `errorevol`, which is a list with only one component:

- **error**: a vector with the error evolution.

Author(s)

Esteban Alfaro-Cortes `<Esteban.Alfaro@uclm.es>`, Matias Gamez-Martinez `<Matias.Gamez@uclm.es>` and Noelia Garcia-Rubio `<Noelia.Garcia@uclm.es>`

References


See Also

`boosting`, `predictNboosting`, `bagging`, `predictNbagging`
importanceplot

Examples

data(BreastCancer)
1 <- length(BreastCancer[,1])
sub <- sample(1:1,2*1/3)
cntrl <- rpart.control(maxdepth = 3, msplit = 0, cp = -1)

BC.adaboost <- boosting(Class ~., data=BreastCancer[sub,-1], mfinal=15, control=cntrl)
BC.adaboost.pred <- predict.boosting(BC.adaboost, newdata=BreastCancer[-sub,-1])

errorevol(BC.adaboost, newdata=BreastCancer[-sub,-1])->evol.test
errorevol(BC.adaboost, newdata=BreastCancer[sub,-1])->evol.train

plot errorevol(evol.test, evol.train)
abline(h=min(evol.test[[1]]), col="red", lty=2, lwd=2)
abline(h=min(evol.train[[1]]), col="blue", lty=2, lwd=2)

importanceplot

Plots the variables relative importance

Description

Plots the relative importance of each variable in the classification task. This measure takes into account the gain of the Gini index given by a variable in a tree and, in the boosting case, the weight of this tree.

Usage

importanceplot(object, ...)

Arguments

object fitted model object of class boosting or bagging. This is assumed to be the result of some function that produces an object with a component named importance as that returned by the boosting and bagging functions.

...

further arguments passed to or from other methods.

Details

For this goal, the varImp function of the caret package is used to get the gain of the Gini index of the variables in each tree.

Value

A labeled plot is produced on the current graphics device (one being opened if needed).
MarginOrderedPruning.Bagging

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

References


See Also

boosting, bagging.

Examples

#Examples
#Iris example
library(rpart)
data(iris)
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
iris.adaboot <- boosting(Species ~ ., data=iris[sub,], mfinal=10)
importanceplot(iris.adaboot)

#Examples with bagging
iris.bagging <- bagging(Species ~ ., data=iris[sub,], mfinal=10)
importanceplot(iris.bagging, horiz=TRUE, cex.names=.6)

MarginOrderedPruning.Bagging

MarginOrderedPruning.Bagging

Description

Margin-based ordered aggregation for bagging pruning

Usage

MarginOrderedPruning.Bagging(baggingObject, trainingset, pruningset, marginType = "unsupervised", doTrace = TRUE)
MarginedOrderedPruning.Bagging

Arguments

- **baggingObject**: fitted model object of class bagging
- **trainingset**: the training set of the bagging object
- **pruningset**: a set aside dataset for bagging pruning
- **marginType**: if "unsupervised" (by default) the margin is the difference between the proportions of votes of the first and second most popular classes. Else the margin is calculated as the difference between the proportion of votes of the correct class and the most popular among the other classes
- **doTrace**: If set to TRUE, give a more verbose output as MarginOrderedPruning.Bagging is running

Value

Returns a list with the following components:

- **prunedBagging**: a pruned bagging object
- **AccuracyOrderedEnsemblePruningSet**: Accuracy of each ordered ensemble on pruning set

Note

Questions about this function should be sent to Li Guo

Author(s)

Li Guo <guoli84@hotmail.com>

References


See Also

- **bagging**, **predict.bagging**

Examples

```r
# mlbench package should be loaded
library(mlbench)
data(Satellite)
# Separate data into 3 parts: training set, pruning set and test set
ind <- sample(3, nrow(Satellite), replace = TRUE, prob = c(0.3, 0.2, 0.5))

# create bagging with training set
#increase mfinal in your own execution of this example to see
#the real usefulness of this function
Satellite.bagging <- bagging(classes ~ ., data = Satellite[ind == 1, ], mfinal = 8)
#Satellite.bagging.pred <- predict(Satellite.bagging, Satellite[ind == 3, ])
```
margins

Description

Calculates the margins of an AdaBoost.M1, AdaBoost-SAMME or Bagging classifier for a data frame

Usage

margins(object, newdata)

Arguments

table

Arguments

table

Details

Intuitively, the margin for an observation is related to the certainty of its classification. It is calculated as the difference between the support of the correct class and the maximum support of an incorrect class.

Value

An object of class margins, which is a list with only one component:

margins a vector with the margins.

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>
plot.ererevol

Plots the error evolution of the ensemble

Description

Plots the previously calculated error evolution of an AdaBoost.M1, AdaBoost-SAMME or Bagging classifier for a data frame as the ensemble size grows.

References


See Also

`bagging`, `boosting`, `plot.margins`, `predict.boosting`, `predict.bagging`

Examples

```r
# Iris example
library(rpart)
data(iris)
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
iris.adaboost <- boosting(Species ~ ., data=iris[sub,], mfinal=10)
margins(iris.adaboost,iris[sub,]) -> iris.margins # training set
plot.margins(iris.margins)

# test set
iris.predboosting <- predict.boosting(iris.adaboost, newdata=iris[-sub,])
margins(iris.predboosting,iris[-sub,]) -> iris.premargins
plot.margins(iris.premargins,iris.margins)

# Examples with bagging
iris.bagging <- bagging(Species ~ ., data=iris[sub,], mfinal=5)
margins(iris.bagging,iris[sub,]) -> iris.bagging.margins # training set
iris.predbagging <- predict.bagging(iris.bagging, newdata=iris[-sub,])
margins(iris.predbagging,iris[-sub,]) ->iris.bagging.premargins # test set
par(bg="lightyellow")
plot.margins(iris.bagging.premargins,iris.bagging.margins)
```
Usage

```r
## S3 method for class 'errorevol'
plot(x, y = NULL, ...)
```

Arguments

- `x` An object of class `errorevol`. This is assumed to be the result of some function that produces an object with a component named `error` as that returned by the `errorevol` function.
- `y` This argument can be used to represent in the same plot the evolution of the test and train errors, `x` and `y`, respectively. Should be `NULL` (by default) or an object of class `errorevol`.
- `...` further arguments passed to or from other methods.

Details

This can be useful to see how fast bagging or boosting reduce the error of the ensemble. In addition, it can detect the presence of overfitting and, therefore, the convenience of pruning the ensemble using `predict.boosting` or `predict.bagging`.

Value

A labeled plot is produced on the current graphics device (one being opened if needed).

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

References


See Also

`boosting`, `predict.boosting`, `bagging`, `predict.bagging`, `errorevol`
Examples

data(iris)
  train <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
  cntrl <- rpart.control(maxdepth=1)
  iris.adaboost <- boosting(Species ~ ., data=iris[train,], mfinal=10, control=cntrl)
  errorevol(iris.adaboost, iris[train,]) -> evol.train
  plot.errorevol(evol.train)
  errorevol(iris.adaboost, iris[-train,]) -> evol.test
  plot.errorevol(evol.test, evol.train)

  # Error evolution along the iterations in training set
  # Comparing error evolution in training and test set
  # See the help of the functions errorevol and boosting

plot.margins

Plots the margins of the ensemble

Description

Plots the previously calculated margins of an AdaBoost.M1, AdaBoost-SAMME or Bagging classifier for a data frame

Usage

## S3 method for class 'margins'
plot(x, y = NULL, ...)

Arguments

x
  An object of class margins. This is assumed to be the result of some function that produces an object with a component named margins as that returned by the margins function.

y
  This argument can be used to represent in the same plot the margins in the test and train sets, x and y, respectively. Should be NULL (by default) or an object of class margins.

... further arguments passed to or from other methods.

Details

Intuitively, the margin for an observation is related to the certainty of its classification. It is calculated as the difference between the support of the correct class and the maximum support of an incorrect class.
Value

A labeled plot is produced on the current graphics device (one being opened if needed).

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

References


See Also

margins, boosting, predict.boosting, bagging, predict.bagging

Examples

data(BreastCancer)
l <- length(BreastCancer[,1])
sub <- sample(1:l, 2*l/3)
cntrl <- rpart.control(maxdepth = 3, minsplit = 0, cp = -1)

BC.adaboost <- boosting(Class ~., data=BreastCancer[sub,-1], mfinal=15, control=cntrl)
BC.adaboost.pred <- predict.bagging(BC.adaboost, newdata= BreastCancer[-sub,-1])

BC.margins<-margins(BC.adaboost,BreastCancer[sub,-1]) # training set
BC.predmargins<-margins(BC.adaboost.pred,BreastCancer[-sub,-1]) # test set
plot.margins(BC.predmargins, BC.margins)

predict.bagging Predicts from a fitted bagging object

Description

Classifies a dataframe using a fitted bagging object.

Usage

# S3 method for class 'bagging'
predict(object, newdata, newmfinal=length(object$trees), ...)

predict.bagging

Arguments

object: fitted model object of class bagging. This is assumed to be the result of some function that produces an object with the same named components as that returned by the bagging function.

ewdata: data frame containing the values at which predictions are required. The predictors referred to in the right side of formula(object) must be present by name in newdata.

newmfinal: The number of trees of the bagging object to be used in the prediction. This argument allows the user to prune the ensemble. By default all the trees in the bagging object are used.

... further arguments passed to or from other methods.

Value

An object of class predict.bagging, which is a list with the following components:

formula: the formula used.

votes: a matrix describing, for each observation, the number of trees that assigned it to each class.

prob: a matrix describing, for each observation, the posterior probability or degree of support of each class. These probabilities are calculated using the proportion of votes in the final ensemble.

class: the class predicted by the ensemble classifier.

confusion: the confusion matrix which compares the real class with the predicted one.

error: returns the average error.

Author(s)

Esteban Alfaro-Cortes <Esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

References


See Also

bagging, bagging.cv
predict.boosting

Predicts from a fitted boosting object

Description

Classifies a dataframe using a fitted boosting object.

Usage

## S3 method for class 'boosting'
predict(object, newdata, newmfinal=length(object$trees), ...)  

Arguments

object 
fitted model object of class boosting. This is assumed to be the result of some function that produces an object with the same named components as that returned by the boosting function.

newdata 
data frame containing the values at which predictions are required. The predictors referred to in the right side of formula(object) must be present by name in newdata.

newmfinal 
The number of trees of the boosting object to be used in the prediction. This argument allows the user to prune the ensemble. By default all the trees in object are used

... 
further arguments passed to or from other methods.
**Value**

An object of class predict.boosting, which is a list with the following components:

- **formula**: the formula used.
- **votes**: a matrix describing, for each observation, the number of trees that assigned it to each class, weighting each tree by its alpha coefficient.
- **prob**: a matrix describing, for each observation, the posterior probability or degree of support of each class. These probabilities are calculated using the proportion of votes in the final ensemble.
- **class**: the class predicted by the ensemble classifier.
- **confusion**: the confusion matrix which compares the real class with the predicted one.
- **error**: returns the average error.

**Author(s)**

Esteban Alfaro-Cortes <esteban.Alfaro@uclm.es>, Matias Gamez-Martinez <Matias.Gamez@uclm.es> and Noelia Garcia-Rubio <Noelia.Garcia@uclm.es>

**References**


**See Also**

boosting, boosting.cv

**Examples**

```r
## rpart library should be loaded
#This example has been hidden to fulfill execution time <5s
#library(rpart)
data(iris)
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
iris.adaboost <- boosting(Species ~ ., data=iris[sub, ], mfinal=10)
iris.preditboosting <- predict.boosting(iris.adaboost, newdata=iris[-sub,])
iris.preditboosting$prob
```

## rpart and mlbench libraries should be loaded
## Comparing the test error of rpart and adaboost.M1

```r
library(rpart)
library(mlbench)
data(BreastCancer)
1 <- length(BreastCancer[,1])
sub <- sample(1:1, 2*1/3)

BC.rpart <- rpart(Class~., data=BreastCancer[sub,-1], maxdepth=3)
BC.rpart.pred <- predict(BC.rpart, newdata=BreastCancer[-sub,-1], type="class")
tb <- table(BC.rpart.pred, BreastCancer$Class[-sub])
error.rpart <- 1-(sum(diag(tb))/sum(tb))
tb
error.rpart

BC.adaboost <- boosting(Class ~., data=BreastCancer[, -1], mfinal=20, coeflearn="Freund",
                        boos=FALSE, control=rpart.control(maxdepth=3))

# Using the pruning option
BC.adaboost.pred <- predict.boosting(BC.adaboost, newdata=BreastCancer[-sub,-1], newmfinal=10)
BC.adaboost.pred$confusion
BC.adaboost.pred$error
```
## Index

**Topic** `classif`
- adabag-package, 2
- autoprun, 4
- bagging, 6
- bagging.cv, 8
- boosting, 9
- boosting.cv, 12
- errorevol, 13
- importanceplot, 15
- MarginOrderedPruning.Bagging, 16
- margins, 18
- plot.errorevol, 19
- plot.margins, 21
- predict.bagging, 22
- predict.boosting, 24

**Topic** `tree`
- adabag-package, 2
- autoprun, 4
- bagging, 6
- bagging.cv, 8
- boosting, 9
- boosting.cv, 12
- errorevol, 13
- importanceplot, 15
- MarginOrderedPruning.Bagging, 16
- margins, 18
- plot.errorevol, 19
- plot.margins, 21
- predict.bagging, 22
- predict.boosting, 24

- adabag (adabag-package), 2
- adabag-package, 2
- adaboost.M1 (boosting), 9
- autoprun, 4, 4

- bagging, 4, 6, 9, 14, 16, 17, 19, 20, 22, 23
- bagging.cv, 4, 7, 8, 23
- boosting, 4, 9, 13, 14, 16, 19, 20, 22, 25
- boosting.cv, 4, 11, 12, 25

- errorevol, 4, 13, 20
- importanceplot, 4, 15

- MarginOrderedPruning.Bagging, 4, 16
- margins, 4, 18, 22
- plot.errorevol, 4, 19
- plot.margins, 4, 19, 21
- predict.bagging, 4, 7, 9, 14, 17, 19, 20, 22, 22
- predict.boosting, 4, 11, 13, 14, 19, 20, 22, 24

- rpart, 5