Title  Applies Adaboost.M1 and Bagging

Version  1.0

Author  Esteban Alfaro Cortés, Matías Gámez Martínez y Noelia García Rubio

Maintainer  Esteban Alfaro Cortés <Esteban.Alfaro@uclm.es>

Depends  R, rpart

Suggests  mlbench

Description  This package 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,

License  GPL version 2 o newer

R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adaboost.M1</td>
<td>Applies the Adaboost.M1 algorithm to a data set</td>
</tr>
<tr>
<td>bagging.cv</td>
<td></td>
</tr>
<tr>
<td>bagging</td>
<td></td>
</tr>
<tr>
<td>boosting.cv</td>
<td></td>
</tr>
<tr>
<td>predict.bagging</td>
<td></td>
</tr>
<tr>
<td>predict.boosting</td>
<td></td>
</tr>
</tbody>
</table>

Index  13

adaboost.M1  Applies the Adaboost.M1 algorithm to a data set

Description

Fits the Adaboost.M1 algorithm proposed by Freund and Schapire in 1996 using classification trees as single classifiers.

Usage

adaboost.M1(formula, data, boos = TRUE, mfinal = 100, coeflearn = 'Breiman', ...)
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. Where $\alpha$ is the weight updating coefficient.
- **minsplit**: the minimum number of observations that must exist in a node in order for a split to be attempted.
- **cp**: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted.
- **maxdepth**: set the maximum depth of any node of the final tree, with the root node counted as depth 0 (past 30 `rpart` will give nonsense results on 32-bit machines). Defaults to the number of classes.

Details

Adaboost.M1 is a simple generalization of Adaboost for more than two classes.

Value

An object of class `adaboost.M1`, 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.
- **class**: the class predicted by the ensemble classifier.
- **importance**: returns the relative importance of each variable in the classification task. This measure is the number of times each variable is selected to split.

Author(s)

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es), Matías Gámez Martínez (Matias.Gamez@uclm.es) and Noelia García Rubio (Noelia.Garcia@uclm.es)

References


See Also

predict.boosting, boosting.cv

Examples

```r
## rpart library should be loaded
library(rpart)
data(iris)
names(iris)<-c("LS","AS","LP","AP","Especies")
iris.adaboost <- adaboost.M1(Especies~LS +AS +LP + AP, data=iris, boos=TRUE, mfinal=10)

## rpart and mlbench libraries should be loaded
## Comparing the test error of rpart and adaboost.M1
library(rpart)
library(mlbench)
data(BreastCancer)
l <- length(BreastCancer[,1])
sub <- sample(1:l,2*l/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 <- adaboost.M1(Class ~.,data=BreastCancer[-1],mfinal=25, maxdepth=3)
BC.adaboost.pred <- predict.boosting(BC.adaboost,newdata=BreastCancer[-sub,-1])
BC.adaboost.pred[-1]

## Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
l <- length(Vehicle[,1])
sub <- sample(1:l,2*l/3)
mfinal <- 25
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
error.rpart

Vehicle.adaboost <- adaboost.M1(Class ~.,data=Vehicle[sub, ],mfinal=mfinal, maxdepth=maxdepth)
Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost,newdata=Vehicle[-sub, ])
Vehicle.adaboost.pred[-1]
```

bagging.cv   Runs v-fold cross validation with Bagging
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

\[
\text{bagging.cv}(\text{formula, data, } v = 10, \text{ mfinal} = 100, \text{ msplit} = 5, \text{ cp} = 0.01, \text{ maxdepth})
\]

Arguments

- **formula**: a formula, as in the `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 \( \text{mfinal}=100 \) iterations.
- **msplit**: the minimum number of observations that must exist in a node in order for a split to be attempted.
- **cp**: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of \( \text{cp} \) is not attempted.
- **maxdepth**: set the maximum depth of any node of the final tree, with the root node counted as depth 0 (past 30 rpart will give nonsense results on 32-bit machines). Defaults to the number of classes.

Value

An object of class `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 Cortés (Esteban.Alfaro@uclm.es), Matías Gámez Martínez (Matias.Gamez@uclm.es) and Noelia García Rubio (Noelia.Garcia@uclm.es)

References


See Also

`bagging`, `predict.bagging`
**Examples**

```r
## rpart library should be loaded
library(rpart)
data(iris)
names(iris) <- c("LS", "AS", "LP", "AP", "Especies")
iris.baggingcv <- bagging.cv(Especies ~ ., v=10, data=iris, mfinal=10, maxdepth=3)

data(kyphosis)
kyphosis.baggingcv <- bagging.cv(Kyphosis ~ Age + Number + Start, data=kyphosis, mfinal=15)

## rpart and mlbench libraries should be loaded
## Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
Vehicle.bagging.cv <- bagging.cv(Class ~ ., data=Vehicle, mfinal=25, maxdepth=5)
Vehicle.bagging.cv[-1]
```

---

**bagging**

*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**

```r
bagging(formula, data, mfinal = 100, minsplit = 5, cp = 0.01, maxdepth = nlevels)
```

**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.
- `minsplit`: the minimum number of observations that must exist in a node, in order for a split to be attempted.
- `cp`: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted.
- `maxdepth`: set the maximum depth of any node of the final tree, with the root node counted as depth 0 (past 30 rpart will give nonsense results on 32-bit machines). Defaults to the number of classes.

**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.
- **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 is the number of times each variable is selected to split.

Author(s)

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es), Matías Gámez Martínez (Matias.Gamez@uclm.es) and Noelia García Rubio (Noelia.Garcia@uclm.es)

References

doost.M1". To appear in International Advances in Economic Research.


See Also

`predict.bagging, bagging.cv`

Examples

```r
## rpart library should be loaded
library(rpart)
data(iris)
names(iris)<-c("LS","AS","LP","AP","Especies")
lirios.bagging <- bagging(Especies~LS +AS +LP+ AP, data=iris, mfinal=10)

## rpart and mlbench libraries should be loaded
library(rpart)
library(mlbench)
data(BreastCancer)
l <- length(BreastCancer[,1])
sub <- sample(1:l,2*l/3)
BC.bagging <- bagging(Class ~.,data=BreastCancer[-sub,-1],mfinal=25, maxdepth=3)
BC.bagging.pred <- predict.bagging(BC.bagging,newdata=BreastCancer[-sub,-1])

## Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
l <- length(Vehicle[,1])
sub <- sample(1:l,2*l/3)
Vehicle.bagging <- bagging(Class ~.,data=Vehicle[sub, ],mfinal=50, maxdepth=5)
```
Vehicle.bagging.pred <- predict.bagging(Vehicle.bagging,newdata=Vehicle[-sub,])
Vehicle.bagging.pred[-1]

**boosting.cv**

*Runs v-fold cross validation with adaboost.M1*

**Description**

The data are divided into *v* non-overlapping subsets of roughly equal size. Then, adaboost.M1 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**

```r
boosting.cv(formula, data, v = 10, boos = TRUE, mfinal = 100, coeflearn = "Breiman", minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))
```

**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), \( \alpha = \frac{1}{2} \ln\left(\frac{1-err}{err}\right) \) is used. If "Freund" \( \alpha = \ln\left(\frac{1-err}{err}\right) \) is used. Where \( \alpha \) is the weight updating coefficient.
- `minsplit`: the minimum number of observations that must exist in a node, in order for a split to be attempted.
- `cp`: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of *cp* is not attempted.
- `maxdepth`: set the maximum depth of any node of the final tree, with the root node counted as depth 0 (past 30 rpart will give nonsense results on 32-bit machines). Defaults to the number of classes.

**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.
predict.bagging

Author(s)

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es), Matías Gámez Martínez (Matias.Gamez@uclm.es) and Noelia García Rubio (Noelia.Garcia@uclm.es)

References


See Also

adaboost.M1, predict.boosting

Examples

```r
## rpart library should be loaded
library(rpart)
names(iris)<-c("LS","AS","LP","AP","Especies")
iris.boostcv <- boosting.cv(Especies ~ ., v=10, data=iris, mfinal=10, maxdepth=3)
data(kyphosis)
kyphosis.boostcv <- boosting.cv(Kyphosis ~ Age + Number + Start, data=kyphosis, mfinal=15)

## rpart and mlbench libraries should be loaded
## Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
Vehicle.boost.cv <- boosting.cv(Class ~.,data=Vehicle,mfinal=25, maxdepth=5)
Vehicle.boost.cv[-1]
```

predict.bagging Predicts from a fitted bagging object.

Description

Classifies a dataframe using a fitted bagging object.

Usage

```r
## S3 method for class 'bagging':
predict(object, newdata, ...)
```
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.

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.

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

Value

An object of class predict.bagging, 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 Cortés ⟨Esteban.Alfaro@uclm.es⟩, Matías Gámez Martínez ⟨Matias.Gamez@uclm.es⟩ and Noelia García Rubio ⟨Noelia.Garcia@uclm.es⟩

References


See Also

bagging bagging.cv

Examples

library(rpart)
data(iris)
names(iris)<-c("LS","AS","LP","AP","Especies")
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
iris.bagging <- bagging(Especies ~ ., data=iris[sub,], mfinal=10)
iris.predbagging<- predict.bagging(iris.bagging, newdata=iris[-sub,])

## rpart and mlbench libraries should be loaded
library(rpart)
library(mlbench)
data(BreastCancer)
l <- length(BreastCancer[,1])
sub <- sample(1:l,2*l/3)
BC.bagging <- bagging(Class ~.,data=BreastCancer[-1],mfinal=25, maxdepth=3)
BC.bagging.pred <- predict.bagging(BC.bagging,newdata=BreastCancer[-sub,-1])

# Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
l <- length(Vehicle[,1])
sub <- sample(1:l,2*l/3)
Vehicle.bagging <- bagging(Class ~.,data=Vehicle[sub,],mfinal=50, maxdepth=5)
Vehicle.bagging.pred <- predict.bagging(Vehicle.bagging,newdata=Vehicle[-sub,])
Vehicle.bagging.pred[-1]

**predict.boosting**

Predicts from a fitted Adaboost.M1 object.

**Description**

Classifies a dataframe using a fitted adaboost.M1 object.

**Usage**

```r
## S3 method for class 'boosting':
predict(object, newdata, ...)
```

**Arguments**

- **object**: fitted model object of class adaboost.M1. This is assumed to be the result of some function that produces an object with the same named components as that returned by the adaboost.M1 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`.
- **...**: further arguments passed to or from other methods.

**Value**

An object of class predict.boosting, 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 Cortés (Esteban.Alfaro@uclm.es), Matías Gámez Martínez (Matias.Gamez@uclm.es) and Noelia García Rubio (Noelia.Garcia@uclm.es)
References


See Also

adaboost.M1, boosting.cv

Examples

```r
## rpart library should be loaded
library(rpart)
data(iris)
names(iris)<-c("LS","AS","LP","AP","Especies")
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
iris.adaboost <- adaboost.M1(Especies ~ ., data=iris[sub,], mfinal=10)
iris.predboosting<- predict.boosting(iris.adaboost, newdata=iris[-sub,])

## rpart and mlbench libraries should be loaded
## Comparing the test error of rpart and adaboost.M1
library(rpart)
library(mlbench)
data(BreastCancer)
l <- length(BreastCancer[,1])
sub <- sample(1:l,2*l/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 <- adaboost.M1(Class ~.,data=BreastCancer[-1],mfinal=25, maxdepth=3)
BC.adaboost.pred <- predict.boosting(BC.adaboost,newdata=BreastCancer[-sub,-1])

## Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
l <- length(Vehicle[,1])
sub <- sample(1:l,2*l/3)
mfinal <- 25
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
```
error.rpart

Vehicle.adaboost <- adaboost.M1(Class ~., data=Vehicle[,, sub], mfinal=mfinal, maxdepth=maxdepth)
Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost, newdata=Vehicle[-, sub])
Vehicle.adaboost.pred[-1]
Index

*Topic **classif**
  adaboost.M1, 1
  bagging, 5
  bagging.cv, 3
  boosting.cv, 7
  predict.bagging, 8
  predict.boosting, 10

*Topic **tree**
  adaboost.M1, 1
  bagging, 5
  bagging.cv, 3
  boosting.cv, 7
  predict.bagging, 8
  predict.boosting, 10

adaboost.M1, 1, 8, 11

bagging, 4, 5, 9
bagging.cv, 3, 6, 9
boosting.cv, 2, 7, 11

predict.bagging, 4, 6, 8
predict.boosting, 2, 8, 10