

# Package ‘adabag’

April 17, 2009

**Title** Applies Adaboost.M1 and Bagging

**Version** 1.1

**Author** Esteban Alfaro Cortes, Matias Gamez Martinez and Noelia Garcia Rubio

**Maintainer** Esteban Alfaro Cortes <Esteban.Alfaro@uclm.es>

**Depends** R (>= 2.4.0), rpart, 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, they can be used to predict on new data. Also, cross validation predictions can be done.

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2007-10-25 19:15:06

## R topics documented:

adaboost.M1 . . . . .	2
bagging . . . . .	4
bagging.cv . . . . .	6
boosting.cv . . . . .	8
predict.bagging . . . . .	9
predict.boosting . . . . .	11

<b>Index</b>	<b>14</b>
--------------	-----------

---

 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',
            minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))
```

### Arguments

formula	a formula, as in the <code>lm</code> function.
data	a data frame in which to interpret the variables named in <code>formula</code> .
boos	if <code>TRUE</code> (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If <code>FALSE</code> , 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 <code>mfinal=100</code> iterations.
coeflearn	if 'Breiman' (by default), $\alpha = 1/2 \ln((1-\text{err})/\text{err})$ is used. If 'Freund' $\alpha = \ln((1-\text{err})/\text{err})$ 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 <code>cp</code> 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 Cortes (Esteban.Alfaro@uclm.es), Matias Gamez Martinez (Matias.Gamez@uclm.es) and Noelia Garcia Rubio (Noelia.Garcia@uclm.es)

**References**

- Alfaro, E., Gamez, M. and Garcia, N. (2007): “Multiclass corporate failure prediction by Adaboost.M1”. *International Advances in Economic Research*, Vol 13, 3, pp. 301–312.
- Freund, Y. and Schapire, R.E. (1996): “Experiments with a New Boosting Algorithm”. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148–156, Morgan Kaufmann.
- Breiman, L. (1998): “Arcing classifiers”. *The Annals of Statistics*, Vol 26, 3, pp. 801–849.

**See Also**

[predict.boosting](#), [boosting.cv](#)

**Examples**

```
## 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

*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, minsplit = 5, cp = 0.01,
        maxdepth = nlevels(vardep))

```

### Arguments

formula	a formula, as in the <code>lm</code> 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 <code>mfinal=100</code> 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 <code>cp</code> 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:

<code>formula</code>	the formula used.
<code>trees</code>	the trees grown along the iterations.
<code>votes</code>	a matrix describing, for each observation, the number of trees that assigned it to each class.
<code>class</code>	the class predicted by the ensemble classifier.
<code>samples</code>	the bootstrap samples used along the iterations.
<code>importance</code>	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 Cortes (Esteban.Alfaro@uclm.es), Matias Gamez Martinez (Matias.Gamez@uclm.es) and Noelia Garcia Rubio (Noelia.Garcia@uclm.es)

### References

Alfaro, E., Gamez, M. and Garcia, N. (2007): "Multiclass corporate failure prediction by Adaboost.M1". International Advances in Economic Research, Vol 13, 3, pp. 301–312.

Breiman, L. (1996): "Bagging predictors". Machine Learning, Vol 24, 2, pp.123–140.

Breiman, L. (1998). "Arcing classifiers". The Annals of Statistics, Vol 26, 3, pp. 801–849.

### See Also

[predict.bagging](#), [bagging.cv](#)

### Examples

```
## 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[,-1],mfinal=25, maxdepth=3)
BC.bagging.pred <- predict.bagging(BC.bagging,newdata=BreastCancer[-sub,-1])
BC.bagging.pred[-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]

```

---

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

```

bagging.cv(formula, data, v = 10, mfinal = 100, minsplit = 5,
           cp = 0.01, maxdepth = nlevels(vardep))

```

### Arguments

formula	a formula, as in the <code>lm</code> function.
data	a data frame in which to interpret the variables named in <code>formula</code>
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 <code>mfinal=100</code> 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 <code>cp</code> 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 <code>rpart</code> 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:

<code>class</code>	the class predicted by the ensemble classifier.
<code>confusion</code>	the confusion matrix which compares the real class with the predicted one.
<code>error</code>	returns the average error.

**Author(s)**

Esteban Alfaro Cortes [⟨Esteban.Alfaro@uclm.es⟩](mailto:Esteban.Alfaro@uclm.es), Matias Gamez Martinez [⟨Matias.Gamez@uclm.es⟩](mailto:Matias.Gamez@uclm.es)  
and Noelia Garcia Rubio [⟨Noelia.Garcia@uclm.es⟩](mailto:Noelia.Garcia@uclm.es)

**References**

Alfaro, E., Gamez, M. and Garcia, N. (2007): "Multiclass corporate failure prediction by Adaboost.M1". *International Advances in Economic Research*, Vol 13, 3, pp. 301–312.

Breiman, L. (1996): "Bagging predictors". *Machine Learning*, Vol 24, 2, pp. 123–140.

Breiman, L. (1998). "Arcing classifiers". *The Annals of Statistics*, Vol 26, 3, pp. 801–849.

**See Also**

[bagging](#), [predict.bagging](#)

**Examples**

```
## 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.baggingcv <- bagging.cv(Class ~ ., data=Vehicle, mfinal=25, maxdepth=5)
Vehicle.baggingcv[-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

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

### Arguments

<code>formula</code>	a formula, as in the <code>lm</code> function.
<code>data</code>	a data frame in which to interpret the variables named in <code>formula</code>
<code>boos</code>	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.
<code>v</code>	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.
<code>mfinal</code>	an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to <code>mfinal=100</code> iterations.
<code>coeflearn</code>	if "Breiman"(by default), $\alpha=1/2\ln((1-\text{err})/\text{err})$ is used. If "Freund" $\alpha=\ln((1-\text{err})/\text{err})$ is used. Where $\alpha$ is the weight updating coefficient.
<code>minsplit</code>	the minimum number of observations that must exist in a node, in order for a split to be attempted.
<code>cp</code>	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of <code>cp</code> is not attempted.
<code>maxdepth</code>	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:

<code>class</code>	the class predicted by the ensemble classifier.
<code>confusion</code>	the confusion matrix which compares the real class with the predicted one.
<code>error</code>	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**

Alfaro, E., Gamez, M. and Garcia, N. (2007): "Multiclass corporate failure prediction by Adaboost.M1". International Advances in Economic Research, Vol 13, 3, pp. 301–312.

Freund, Y. and Schapire, R.E. (1996): "Experiments with a New Boosting Algorithm". In Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156, Morgan Kaufmann.

Breiman, L. (1998): "Arcing classifiers". The Annals of Statistics, Vol 26, 3, pp. 801–849.

**See Also**

[adaboost.Ml](#), [predict.boosting](#)

**Examples**

```
## rpart library should be loaded
library(rpart)
data(iris)
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**

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

**Arguments**

<code>object</code>	fitted model object of class <code>bagging</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>bagging</code> function.
<code>newdata</code>	data frame containing the values at which predictions are required. The predictors referred to in the right side of <code>formula(object)</code> must be present by name in <code>newdata</code> .
<code>...</code>	further arguments passed to or from other methods.

**Value**

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

<code>class</code>	the class predicted by the ensemble classifier.
<code>confusion</code>	the confusion matrix which compares the real class with the predicted one.
<code>error</code>	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**

- Alfaro, E., Gamez, M. and Garcia, N. (2007): "Multiclass corporate failure prediction by Adaboost.M1". *International Advances in Economic Research*, Vol 13, 3, pp. 301–312.
- Breiman, L. (1996): "Bagging predictors". *Machine Learning*, Vol 24, 2, pp. 123–140.
- Breiman, L. (1998). "Arcing classifiers". *The Annals of Statistics*, Vol 26, 3, pp. 801–849.

**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])
BC.bagging.pred[-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

```

## 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 Cortes (Esteban.Alfaro@uclm.es), Matias Gamez Martinez (Matias.Gamez@uclm.es) and Noelia Garcia Rubio (Noelia.Garcia@uclm.es)

**References**

Alfaro, E., Gamez, M. and Garcia, N. (2007): "Multiclass corporate failure prediction by Adaboost.M1". International Advances in Economic Research, Vol 13, 3, pp. 301–312.

Freund, Y. and Schapire, R.E. (1996): "Experiments with a New Boosting Algorithm". En Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156, Morgan Kaufmann.

Breiman, L. (1998): "Arcing classifiers". The Annals of Statistics, Vol 26, 3, pp. 801–849.

**See Also**

[adaboost.M1](#), [boosting.cv](#)

**Examples**

```
## 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])
BC.adaboost.pred[-1]

## Data Vehicle (four classes)
library(rpart)
library(mlbench)
data(Vehicle)
l <- length(Vehicle[,1])
```

```
sub <- sample(1:1, 2*1/3)
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, 4
- bagging.cv, 6
- boosting.cv, 7
- predict.bagging, 9
- predict.boosting, 11

## \*Topic **tree**

- adaboost.M1, 1
- bagging, 4
- bagging.cv, 6
- boosting.cv, 7
- predict.bagging, 9
- predict.boosting, 11

adaboost.M1, 1, 9, 12

bagging, 4, 7, 10

bagging.cv, 5, 6, 10

boosting.cv, 3, 7, 12

predict.bagging, 5, 7, 9

predict.boosting, 3, 9, 11