Package ‘ada’

May 13, 2016

Version  2.0-5
Date     2016-04-12
Title    The R Package Ada for Stochastic Boosting
Author   Mark Culp, Kjell Johnson, and George Michailidis
Depends  R(>= 2.10),rpart
Description Performs discrete, real, and gentle boost under both exponential and logistic loss on a given data set. The package ada provides a straightforward, well-documented, and broad boosting routine for classification, ideally suited for small to moderate-sized data sets.
Maintainer Mark Culp <mvculp@mail.wvu.edu>
License   GPL
NeedsCompilation no
Repository CRAN
Date/Publication 2016-05-13 11:40:44

R topics documented:

ada ......................................................... 2
addtest .................................................... 5
pairs.ada .................................................. 6
plot.ada ................................................... 7
predict.ada .............................................. 8
print.ada ................................................ 9
soldat .................................................... 10
summary.ada .......................................... 11
update.ada ............................................. 11

Index 13
Fitting Stochastic Boosting Models

Description

'ada' is used to fit a variety stochastic boosting models for a binary response as described in Additive Logistic Regression: A Statistical View of Boosting by Friedman, et al. (2000).

Usage

ada(x,...)

## Default S3 method:
ada(x, y, test.x, test.y=NULL, loss=c("exponential","logistic"),
  type=c("discrete","real","gentle"),iter=50, nu=0.1, bag.frac=0.5,
  model.coef=TRUE,bag.shift=FALSE,max.iter=20,delta=10^(-10),
  verbose=FALSE,...,na.action=na.rpart)

## S3 method for class 'formula'
ada(formula, data, ..., subset, na.action=na.rpart)

Arguments

x matrix of descriptors.
y vector of responses. 'y' may have only two unique values.
test.x testing matrix of descriptors (optional)
test.y vector of testing responses (optional)
loss loss="exponential", "ada","e" or any variation corresponds to the default boosting under exponential loss. loss="logistic","l2","l" provides boosting under logistic loss.
iter number of boosting iterations to perform. Default = 50.
nu shrinkage parameter for boosting, default taken as 1.
bag.frac sampling fraction for samples taken out-of-bag. This allows one to use random permutation which improves performance.
model.coef flag to use stageweights in boosting. If FALSE then the procedure corresponds to epsilon-boosting.
bag.shift flag to determine whether the stageweights should go to one as nu goes to zero. This only makes since if bag.frac is small. The rationale behind this parameter is discussed in (Culp et al., 2006).
max.iter number of iterations to perform in the newton step to determine the coefficient.
delta tolerance for convergence of the newton step to determine the coefficient.
verbose  print the number of iterations necessary for convergence of a coefficient.
formula  a symbolic description of the model to be fit.
data  an optional data frame containing the variables in the model.
subset  an optional vector specifying a subset of observations to be used in the fitting process.
na.action  a function that indicates how to process ‘NA’ values. Default=na.rpart.
...  arguments passed to rpart.control. For stumps, use rpart.control(maxdepth=1,cp=-1,msplit=0).

maxdepth  controls the depth of trees, and cp controls the complexity of trees.
The priors should also be fixed through the parms argument as discussed in the second reference.

Details
This function directly follows the algorithms listed in “Additive Logistic Regression: A Statistical View of Boosting”.

When using usage ‘ada(x,y)’: x data can take the form data.frame or as.matrix. y data can take form data.frame, as.factor, as.matrix, as.array, or as.table. Missing values must be removed from the data prior to execution.

When using usage ‘ada(y~.)’: data must be in a data frame. Response can have factor or numeric values. Missing values can be present in the descriptor data, whenever na.action is set to any option other than na.pass.

After the model is fit, ‘ada’ prints a summary of the function call, the method used for boosting, the number of iterations, the final confusion matrix (observed classification vs predicted classification; labels for classes are same as in response), the error for the training set, and testing, training , and kappa estimates of the appropriate number of iterations.
A summary of this information can also be obtained with the command ‘print(x)’.

Corresponding functions (Use help with summary.ada, predict.ada, … varplot for additional information on these commands):

summary : function to print a summary of the original function call, method used for boosting, number of iterations, final confusion matrix, accuracy, and kappa statistic (a measure of agreement between the observed classification and predicted classification). ‘summary’ can be used for training, testing, or validation data.
predict : function to predict the response for any data set (train, test, or validation).
plot : function to plot performance of the algorithm across boosting iterations. Default plot is iteration number (x-axis) versus prediction error (y-axis) for the data set used to build the model. Function can also simultaneously produce an error plot for an external test set and a kappa plot for training and test sets.
pairs : function to produce pairwise plots of descriptors. Descriptors are arranged by decreasing frequency of selection by boosting (upper left = most frequently chosen). The color of the marker in the plot represents class membership; the Size of the marker represents predicted class probability. The larger the marker, the higher the probability of classification.
varplot : plot of variables ordered by the variable importance measure (based on improvement).
addtest : add a testing data set to the ada object, therefore the testing errors only have to be computed once.
update : add more trees to the ada object.
Value

model The following items are the different components created by the algorithms:
trees: ensembl of rpart trees used to fit the model alpha: the weights of the
trees used in the final aggregate model (AdaBoost only; see references for more
information) F : F[[1]] corresponds to the training sum, F[[2]], . . . corresponds
to testing sums. errs : matrix of errs, training, kappa, testing 1, kappa 1, . . . lw :
last weights calculated, used by update routine

fit The predicted classification for each observation in the orginal level of the re-
sponse.
call The function call.
u The shrinkage parameter

type The type of adaboost performed: ‘discrete’, ‘real’, ‘logit’, and ‘gentle’.
confusion The confusion matrix (True value vs. Predicted value) for the training data.
iter The number of boosting iterations that were performed.
actual The original response vector.

Warnings

For LogitBoost and Gentle Boost, under certain circumstances, the methods will fail to classify the
data into more than one category. If this occurs, try modifying the rpart.control options such as
‘minsplit’, ‘cp’, and ‘maxdepth’.

‘ada’ does not currently handle multiclass problems. However, there is an example in (Culp et al.,
2006) that shows how to use this code in that setting. Plots and other functions are not set up for
this analysis.

Author(s)

Mark Culp, University of Michigan Kjell Johnson, Pfizer, Inc. George Michailidis, University of
Michigan

Special thanks goes to: Zhiguang Qian, Georgia Tech University Greg Warnes, Pfizer, Inc.

References

Report, Department of Statistics, Standford University.


of Statistical Software, 16.

See Also

print.ada, summary.ada, predict.ada, plot.ada, pairs.ada, update.ada, addtest
**Examples**

```r
## fit discrete ada boost to a simple example
data(iris)
##drop setosa
iris[iris$Species!="setosa",]->iris
##set up testing and training data (60% for training)
n<-dim(iris)[1]
trind<-sample(1:n,floor(.6*n),FALSE)
teind<-setdiff(1:n,trind)
iris[,5]<- as.factor(levels(iris[,5])[2:3][as.numeric(iris[,5])-1])
##fit 8-split trees
gdis<-ada(Species~.,data=iris[trind,],iter=20,nu=1,type="discrete")
##add testing data set
gdis=addtest(gdis,iris[teind,-5],iris[teind,5])
#plot gdis
plot(gdis,TRUE,TRUE)
#variable selection plot
varplot(gdis)
#pairwise plot
pairs(gdis,iris[trind,-5],maxvar=2)
```

## for many more examples refer to reference (Culp et al., 2006)

---

**addtest**  
*Add a test set to ada*

**Description**

`addtest` updates the `ada` object to have additional testing errors and testing kappa accuracies for each iteration.

**Usage**

```r
addtest(x,test.x,test.y, ...)
```

**Arguments**

- `x` object generated by the function `ada`.
- `test.x` new x data
- `test.y` the true labeling for this testing data
- `...` other arguments not used by this function.

**Value**

updated ada object.

**See Also**

`ada`, `update.ada`
Description

This command produces pairwise plots of the data. The data in the upper panel of pairwise plots colors the observations by observed class membership (if membership is provided). The lower panel of pairwise plots colors the observations by predicted classes. In addition, the plotting symbol is scaled by the class probability estimate from by adaboost.

The varplot command produces a variable importance plot using the improve criteria given in the reference (Hastie et al., 2001, pg332). This is a rather standard measure for determining variable importance.

Usage

```r
## S3 method for class 'ada'
pairs(x, train.data = NULL, vars = NULL, maxvar = 10,
    test.x = NULL, test.y = NULL,
    test.only = FALSE, col = c(2, 4), pch = c(1, 2), ...)

varplot(x, plot.it = TRUE, type = c("none","scores"), max.var.show=30, ...)
```

Arguments

- `x` object generated by ‘ada’.
- `train.data` the ‘data.frame’ of the original data used to train the classifier. The names of this ‘data.frame’ must be the same as the variable names as the object generated by ‘ada’. `x.data` is used by both the ‘pairs’ command. Default = NULL.
- `vars` a vector of variables to include for this plot. The variable number must correspond to a specific column in ‘x’. For example, vars=c(1,2), generates a plot for the first two columns for ‘x.data’. Note: vars is only used for the ‘pairs’ command. Default = NULL.
- `maxvar` the maximum number of variables for the pairwise plot. If maxvar = 5, then ‘varplot’ chooses the five most important variables and places these in descending order in the plot. Maxvar is only used for the ‘pairs’ command. Default = 10.
- `test.x` an option to plot pairwise descriptors for a test data set. ‘test.data’ should be of type ‘data.frame’. ‘test.data’ is only used for the ‘pairs’ command. Default = NULL.
- `test.y` the corresponding response for the test data set. If ‘test.response’ is not specified, then the color of the symbols for the test data in the pairwise plots are black; training data are colored by class. ‘test.response’ is only used for the ‘pairs’ command. Default = NULL.
**plot.ada**

The `plot.ada` function produces plots of the overall classification error at each boosting iteration for both the training and test sets. In addition, the function can produce plots of the measure of agreement (kappa) between the predicted classification and actual classification at each boosting iteration for both the training and test sets.

**test.only**
- Provides pairwise plots for test data only (`test.only = TRUE`). Default = FALSE.
- If `test.response` is not specified, then `test.only` is ignored. `test.only` is only used for the ‘pairs’ command. Default = NULL.

**col**
- Color for plot symbols one for each class. Default `col=c(2,4)` (i.e. red and blue).

**pch**
- Pch for plot set two symbols. Default `pch=c(1,2)` (i.e. circle and triangle).

**plot.it**
- Provides a plot of frequencies for each variable (`plot.it = TRUE`). ‘plot.it’ is only used for the ‘varplot’ command. Default = NULL.

**type**
- If type="none" then nothing is returned. Default = “none”. If type="scores", the frequencies are returned.

**max.var.show**
- If plot.it is TRUE then this controls the number of variables shown for the plot.

**Details**

The ‘varplot’ command provides a sense of variable importance—the more frequently a variable is selected for boosting, the more likely the variable contains useful information for classification. Pairwise interactions of important variables can then be visualized using ‘varplot’. Note: The ‘pairs’ command calls the ‘varplot’ command.

**Value**

- `scores` If type="scores" then the frequencies for each variable is returned by the varplot command.

**Note**

This plot was designed as tool to use with adaboost. Please send any comments or suggestions for improvement to the authors.

**References**


---

**plot.ada**

*Plots for Ada*

---

**Description**

This function produces plots of the overall classification error at each boosting iteration for both the training and test sets. In addition, the function can produce plots of the measure of agreement (kappa) between the predicted classification and actual classification at each boosting iteration for both the training and test sets.
Usage

## S3 method for class 'ada'
plot(x, kappa = FALSE, test=FALSE, cols= rainbow(dim(x$modelerrs)[2]+1), tflag=TRUE, ...)

Arguments

- **x**
  - the object created by ada.
- **kappa**
  - option for a plot of Kappa values at each iteration. kappa = TRUE produces a plot of Kappa values. Default = FALSE.
- **test**
  - option for a plot of testing error values at each iteration. test=TRUE produces a plot of test values. Default=FALSE.
- **cols**
  - colors used for lines to be plotted
- **tflag**
  - indicates whether to include the title in the plot or not
- **...**
  - additional layout command parameter (see layout).

Value

- No value returned

See Also

- **ada**

---

### predict.ada

*Predict a data set using Ada*

Description

**predict** classifies a new set of observations from a previously built classifier. This function will provide either a vector of new classes, class probability estimates, or both.

Usage

## S3 method for class 'ada'
predict(object, newdata, type = c("vector", "probs", "both", "F"), n.iter=NULL, ...)

Arguments

- **object**
  - object generated by ada.
- **newdata**
  - new data set to predict. This data set must be of type ‘data.frame’ and prediction data set is required for this approach.
- **type**
  - choice for predictions. type="vector" returns the default class labels. type="prob" returns the probability class estimates. type="both" returns both the default class labels and probability class estimates. type="F" returns the ensemble average, where the class label is sign(F). This is mainly usefull for the multiclass case.
print.ada

n.iter number of iterations to consider for the prediction. By default this is iter from
the ada call (n.iter< iter)
...
other arguments not used by this function.

Details
This function was modeled after predict.rpart. Furthermore, predict.rpart will be invoked to
handle predictions by each tree in the ensamble.

Value
fit a vector of fitted responses. Fit will be returned if type="vector".
probs a matrix of class probability estimates. The first column corresponds to the
first label in the 'levels' of the response. The second column corresponds to
the second label in the 'levels' of the response. Probs are returned whenever
type="probs".
both returns both the vector of fitted responses and class probability estimates. The
first element returns the fitted responses and will be labeled as 'class'. The
second element returns the class probability estimates and will be labeled as
'probs'.
F this is used in the multiclass case when one uses the package to perform 1 v.s.
all.

Note
This function is invoked by the summary, pairs, and plot S3 generics invoked with an ada object.
If an error occurs in one of the above commands then try using this command directly to track
possible errors. Also, the newdata data set must be of type 'data.frame' when invoking summary,
pairs, and plot.

See Also
ada.default, summary.ada, print.ada, plot.ada, pairs.ada, update.ada, addtest

____________________________
print.ada  Model Information for Ada

Description
print lists the model information and final confusion matrix for submitted data.

Usage
## S3 method for class 'ada'
print(x, ...)

Arguments

  x object generated by the function ada.
  ... other arguments not used by this function.

Details

  print produces a summary of the original function call, method used for boosting, number of
  iterations, final confusion matrix, error from data used to build the model, and estimates of M.
  Note: any object of class ada invokes print, when printed to the screen.

Value

  No value returned.

See Also

  ada.default, summary.ada, predict.ada, plot.ada, pairs.ada, update.ada, addtest

---

soldat  

Solvability Data

Description

  A data set that contains information about compounds used in drug discovery. Specifically, this data
  set consists of 5631 compounds on which an in-house solubility screen (ability of a compound to
dissolve in a water/solvent mixture) was performed.
  Based on this screen, compounds were categorized as either insoluble (n=3493) or soluble (n=2138).
  Then, for each compound, 72 continuous, noisy structural descriptors were computed.

Usage

  data(soldat)

Format

  A data frame with 5631 observations on the following 73 variables. Some rows have missing data.

Examples

  data(soldat)
summary.ada

Summary of model fit for arbitrary data (test, validation, or training)

Description

summary lists the model information for fitted model and final confusion matrix.

Usage

## S3 method for class 'ada'
summary(object, n.iter=NULL, ...)

Arguments

- **object**: object generated by 'ada'.
- **n.iter**: specific iteration to obtain the training and testing information at.
- **...**: other arguments not used by this function.

Details

summary produces a summary of the original function call, method used for boosting for a specific iteration, accuracy, and kappa statistic (a measure of agreement between the observed classification and predicted classification) for the training data.

In addition, if any other data set (i.e. test or validation) has been incorporated to the ada object (see addtest), summary produces analogous information.

See Also

- ada.predict.ada
- plot.ada
- pairs.ada

update.ada

Add more trees to an ada object

Description

ada.update updates the ada object to have additional trees given a new number of iterations.

Usage

## S3 method for class 'ada'
update(object, x, y, test.x, test.y = NULL, n.iter, ...)

Arguments

- object: object generated by the function ada.
- x: x training data
- y: training response
- test.x: x testing data (optional)
- test.y: the true labeling for this testing data (optional)
- n.iter: new number of iterations, must be provided and n.iter>iter
- ... other arguments not used by this function.

Value

updated ada object.

See Also

- ada.default
- summary.ada
- predict.ada
- plot.ada
- pairs.ada
Index

*Topic classes
   ada, 2

*Topic datasets
   soldat, 10

*Topic methods
   ada, 2
       addtest, 5
       pairs.ada, 6
       plot.ada, 7
       predict.ada, 8
       print.ada, 9
       summary.ada, 11
       update.ada, 11

*Topic models
   ada, 2

   ada, 2, 5, 8, 11
   ada.default, 9, 10, 12
   addtest, 4, 5, 9, 10

   pairs.ada, 4, 6, 9–12
   plot.ada, 4, 7, 9–12
   predict.ada, 4, 8, 10–12
   print.ada, 4, 9, 9

   soldat, 10
   summary.ada, 4, 9, 10, 11, 12

   update.ada, 4, 5, 9, 10, 11

   varplot(pairs.ada), 6