Package 'bst'

July 2, 2014

Type Package

Title Gradient Boosting

Version 0.3-4

Date 2014-05-8

Author Zhu Wang

Maintainer Zhu Wang <zwang@connecticutchildrens.org>

Description The package contains HingeBoost for binary and multi-class classification, with unequal misclassification costs for binary case. Functional gradient descent algorithm to optimize the hinge loss. The algorithm can fit linear and nonlinear classifiers.

Imports rpart, gbm

Suggests hdi

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-24 01:03:27

R topics documented:

`bst-package` *Boosting for Classification and Regression*

Description

Gradient descent boosting for hinge loss and square error loss.

Details

Package:	bst
Type:	Package
Version:	0.1
Date:	2010-04-15
License:	GPL-2
LazyLoad:	yes

Author(s)

Zhu Wang

`bst` *Boosting for Classification and Regression*

Description

Gradient boosting for optimizing hinge or squared error loss functions with componentwise linear, smoothing splines, tree models as base learners.

Usage

```
bst(x, y, cost = 0.5, family = c("hinge", "gaussian"), ctrl = bst_control(),
control.tree = list(maxdepth = 1), learner = c("ls", "sm", "tree"))
## S3 method for class 'bst'
print(x, ...)
```

```
## S3 method for class 'bst'
predict(object, newdata=NULL, newy=NULL, mstop=NULL,
type=c("response", "all.res", "class", "loss", "error"), ...)
## S3 method for class 'bst'
plot(x, type = c("step", "norm"),...)
## S3 method for class 'bst'
coef(object, ...)
## S3 method for class 'bst'
fpartial(object, mstop=NULL, newdata=NULL)
```

Arguments

x	a data frame containing the variables in the model.
y	vector of responses. y must be in {1, -1} for family = "hinge".
cost	price to pay for false positive, 0 < cost < 1; price of false negative is 1-cost.
family	family = "hinge" for hinge loss and family="gaussian" for squared error loss. Implementing the negative gradient corresponding to the loss function to be minimized. By default, hinge loss for +1/-1 binary responses is used.
ctrl	an object of class bst_control.
control.tree	control parameters of rpart.
learner	a character specifying the component-wise base learner to be used: ls linear models, sm smoothing splines, tree regression trees.
type	in predict a character indicating whether the response, all responses across the boosting iterations, classes, loss or classification errors should be predicted in case of hinge problems. in plot, plot of boosting iteration or L_1 norm.
object	class of bst.
newdata	new data for prediction with the same number of columns as x.
newy	new response.
mstop	boosting iteration for prediction.
...	additional arguments.

Details

A linear or nonlinear classifier is fitted using a boosting algorithm for +1/-1 responses.

Value

An object of class bst with print, coef, plot and predict methods are available for linear models. For nonlinear models, methods print and predict are available.

Author(s)

Zhu Wang

References

Zhu Wang (2011), HingeBoost: ROC-Based Boost for Classification and Variable Selection. *The International Journal of Biostatistics*, **7**(1), Article 13.

See Also

`cv.bst` for cross-validated stopping iteration. Furthermore see `bst_control`

Examples

```
x <- matrix(rnorm(100*5),ncol=5)
c <- 2*x[,1]
p <- exp(c)/(exp(c)+exp(-c))
y <- rbinom(100,1,p)
y[y != 1] <- -1
x <- as.data.frame(x)
dat.m <- bst(x, y, ctrl = bst_control(mstop=50), family = "hinge", learner = "ls")
dat.m1 <- bst(x, y, ctrl = bst_control(twinboost=TRUE,
f.init=predict(dat.m), xselect.init = dat.m$xselect, mstop=50))
```

---

bst.sel *Function to select number of predictors*

Description

Function to determine the first q predictors in the boosting path, or perform (10-fold) cross-validation and determine the optimal set of parameters

Usage

```
bst.sel(x, y, q, type=c("firstq", "cv"), ...)
```

Arguments

`x`	Design matrix (without intercept).
`y`	Continuous response vector for linear regression
`q`	Maximum number of predictors that should be selected.
`type`	if `type="firstq"`, return the first `q` predictors in the boosting path. if `type="cv"`, perform (10-fold) cross-validation and determine the optimal set of parameters
`...`	Further arguments to be passed to `cv.bst`.

Details

Function to determine the first q predictors in the boosting path, or perform (10-fold) cross-validation and determine the optimal set of parameters. This may be used for p-value calculation. See below.

Value

Vector of selected predictors.

Author(s)

Zhu Wang

Examples

```
## Not run:
x <- matrix(rnorm(100*100), nrow = 100, ncol = 100)
y <- x[,1] * 2 + x[,2] * 2.5 + rnorm(100)
sel <- bst.sel(x, y, q=10)
library("hdi")
fit.multi <- hdi(x, y, method = "multi.split",
model.selector =bst.sel,
args.model.selector=list(type="firstq", q=10))
fit.multi
fit.multi$pval[1:10] ## the first 10 p-values
fit.multi <- hdi(x, y, method = "multi.split",
model.selector =bst.sel,
args.model.selector=list(type="cv"))
fit.multi
fit.multi$pval[1:10] ## the first 10 p-values

## End(Not run)
```

---

bst_control *Control Parameters for Boosting*

Description

Specification of the number of boosting iterations, step size and other parameters for boosting algorithms.

Usage

```
bst_control(mstop = 50, nu = 0.1, twinboost = FALSE,
f.init = NULL, xselect.init = NULL, center = FALSE, trace = FALSE,
numsample = 50, df=4)
```

Arguments

mstop	an integer giving the number of boosting iterations.
nu	a small number (between 0 and 1) defining the step size or shrinkage parameter.
twinboost	a logical value: TRUE for twin boosting.
f.init	the estimate from the first round of twin boosting. Only useful when twinboost=TRUE.

xselect.init	the variable selected from the first round of twin boosting. Only useful when twinboost=TRUE.
center	a logical value: TRUE to center covariates with mean.
trace	a logical value for printout of more details of information during the fitting process.
numsample	number of random sample variable selected in the first round of twin boosting. This is potentially useful in the future implementation.
df	degree of freedom used in smoothing splines.

Details

Objects to specify parameters of the boosting algorithms implemented in bst, via the ctrl argument.

Value

An object of class bst_control, a list.

See Also

bst

---

cv.bst *Cross-Validation for Binary HingeBoost*

Description

Cross-validated estimation of the empirical risk for boosting parameter selection.

Usage

```
cv.bst(x, y, K = 10, cost = 0.5, family = c("hinge", "gaussian"),
learner = c("ls", "sm", "tree"), ctrl = bst_control(),
type = c("risk", "misc"), plot.it = TRUE, se = TRUE, ...)
```

Arguments

x	a data frame containing the variables in the model.
y	vector of responses. y must be in {1, -1} for family = "hinge".
K	K-fold cross-validation
cost	price to pay for false positive, 0 < cost < 1; price of false negative is 1-cost.
family	family = "hinge" for hinge loss and family="gaussian" for squared error loss. Implementing the negative gradient corresponding to the loss function to be minimized. By default, hinge loss for +1/-1 binary responses is used.

learner	a character specifying the component-wise base learner to be used: `ls` linear models, `sm` smoothing splines, `tree` regression trees.
ctrl	an object of class `bst_control`.
type	cross-validation criteria. For `family="hinge"`, `type="risk"` is hinge risk and `type="misc"` is misclassification error. For `family="gaussian"`, only empirical risks.
plot.it	a logical value, to plot the estimated risks if `TRUE`.
se	a logical value, to plot with standard errors.
...	additional arguments.

Value

object with

residmat	empirical risks in each cross-validation at boosting iterations
mstop	boosting iteration steps at which CV curve should be computed.
cv	The CV curve at each value of mstop
cv.error	The standard error of the CV curve
family	family = "hinge" for hinge loss and family="gaussian" for squared error loss.

...

See Also

`bst`

Examples

```
## Not run:
x <- matrix(rnorm(100*5),ncol=5)
c <- 2*x[,1]
p <- exp(c)/(exp(c)+exp(-c))
y <- rbinom(100,1,p)
y[y != 1] <- -1
x <- as.data.frame(x)
cv.bst(x, y, ctrl = bst_control(mstop=50), family = "hinge", learner = "ls")
cv.bst(x, y, ctrl = bst_control(mstop=50), family = "hinge", learner = "ls", type="misc")

## End(Not run)
```

cv.mada *Cross-Validation for one-vs-all AdaBoost with multi-class problem*

Description

Cross-validated estimation of the empirical misclassification error for boosting parameter selection.

Usage

```
cv.mada(x, y, balance=FALSE, K=10, nu=0.1, mstop=200, interaction.depth=1,
trace=FALSE, plot.it = TRUE, se = TRUE, ...)
```

Arguments

x	a data matrix containing the variables in the model.
y	vector of multi class responses. y must be an interger vector from 1 to C for C class problem.
balance	logical value. If TRUE, The K parts were roughly balanced, ensuring that the classes were distributed proportionally among each of the K parts.
K	K-fold cross-validation
nu	a small number (between 0 and 1) defining the step size or shrinkage parameter.
mstop	number of boosting iteration.
interaction.depth	used in gbm to specify the depth of trees.
trace	if TRUE, iteration results printed out.
plot.it	a logical value, to plot the cross-validation error if TRUE.
se	a logical value, to plot with 1 standard deviation curves.
...	additional arguments.

Value

object with

residmat	empirical risks in each cross-validation at boosting iterations
fraction	abscissa values at which CV curve should be computed.
cv	The CV curve at each value of fraction
cv.error	The standard error of the CV curve

...

See Also

mada

cv.mhingebst *Cross-Validation for Multi-class Hinge Boosting*

Description

Cross-validated estimation of the empirical multi-class hinge loss for boosting parameter selection.

Usage

```
cv.mhingebst(x, y, balance=FALSE, K = 10, cost = NULL, family = "hinge",
learner = c("tree", "ls", "sm"), ctrl = bst_control(),
type = c("risk","misc"), plot.it = TRUE, se = TRUE, ...)
```

Arguments

x	a data frame containing the variables in the model.
y	vector of responses. y must be integers from 1 to C for C class problem.
balance	logical value. If TRUE, The K parts were roughly balanced, ensuring that the classes were distributed proportionally among each of the K parts.
K	K-fold cross-validation
cost	price to pay for false positive, 0 < cost < 1; price of false negative is 1-cost.
family	family = "hinge" for hinge loss. Implementing the negative gradient corresponding to the loss function to be minimized.
learner	a character specifying the component-wise base learner to be used: ls linear models, sm smoothing splines, tree regression trees.
ctrl	an object of class bst_control.
type	for family="hinge", type="risk" is hinge risk.
plot.it	a logical value, to plot the estimated risks if TRUE.
se	a logical value, to plot with standard errors.
...	additional arguments.

Value

object with

residmat	empirical risks in each cross-validation at boosting iterations
fraction	abscissa values at which CV curve should be computed.
cv	The CV curve at each value of fraction
cv.error	The standard error of the CV curve

...

See Also

mhingebst

`cv.mhingeova` *Cross-Validation for one-vs-all HingeBoost with multi-class problem*

Description

Cross-validated estimation of the empirical misclassification error for boosting parameter selection.

Usage

```
cv.mhingeova(x, y, balance=FALSE, K=10, cost = NULL, nu=0.1,
learner=c("tree", "ls", "sm"), maxdepth=1, m1=200, twinboost = FALSE,
m2=200, trace=FALSE, plot.it = TRUE, se = TRUE, ...)
```

Arguments

`x`	a data frame containing the variables in the model.
`y`	vector of multi class responses. `y` must be an interger vector from 1 to C for C class problem.
`balance`	logical value. If TRUE, The K parts were roughly balanced, ensuring that the classes were distributed proportionally among each of the K parts.
`K`	K-fold cross-validation
`cost`	price to pay for false positive, $0 <$ `cost` < 1; price of false negative is 1-`cost`.
`nu`	a small number (between 0 and 1) defining the step size or shrinkage parameter.
`learner`	a character specifying the component-wise base learner to be used: `ls` linear models, `sm` smoothing splines, `tree` regression trees.
`maxdepth`	tree depth used in `learner=tree`
`m1`	number of boosting iteration
`twinboost`	logical: twin boosting?
`m2`	number of twin boosting iteration
`trace`	if TRUE, iteration results printed out
`plot.it`	a logical value, to plot the estimated risks if `TRUE`.
`se`	a logical value, to plot with standard errors.
`...`	additional arguments.

Value

object with

`residmat`	empirical risks in each cross-validation at boosting iterations
`fraction`	abscissa values at which CV curve should be computed.
`cv`	The CV curve at each value of fraction
`cv.error`	The standard error of the CV curve

...

Note

The functions for balanced cross validation were from R package pmar.

See Also

mhingeova

---

ex1data *Generating Three-class Data*

Description

Randomly generate data for a three-class model.

Modified code from http://www.stat.osu.edu/~yklee/software.html.

Usage

```
ex1data(n.data)
```

Arguments

n.data	number of data samples.

Details

The data is generated based on Example 1 described in Wang (2011).

Value

A list with n.data by 50 predictor matrix x, three-class response y and conditional probabilities p.

Author(s)

Zhu Wang

References

Zhu Wang (2012), Multi-class HingeBoost: Method and Application to the Classification of Cancer Types Using Gene Expression Data. *Methods of Information in Medicine*, **51**(2), 162–7.

Examples

```
## Not run:
dat <- ex1data(200)
mhingebst(x=dat$x, y=dat$y)

## End(Not run)
```

loss *Internal Function*

Description

Internal Function

mada *Multi-class AdaBoost*

Description

One-vs-all multi-class AdaBoost

Usage

```
mada(xtr, ytr, xte=NULL, yte=NULL, mstop=50, nu=0.1, interaction.depth=1)
```

Arguments

`xtr`	training data matrix containing the predictor variables in the model.
`ytr`	training vector of responses. `ytr` must be integers from 1 to C, for C class problem.
`xte`	test data matrix containing the predictor variables in the model.
`yte`	test vector of responses. `yte` must be integers from 1 to C, for C class problem.
`mstop`	number of boosting iteration.
`nu`	a small number (between 0 and 1) defining the step size or shrinkage parameter.
`interaction.depth`	used in gbm to specify the depth of trees.

Details

For a C-class problem (C > 2), each class is separately compared against all other classes with AdaBoost, and C functions are estimated to represent confidence for each class. The classification rule is to assign the class with the largest estimate.

Value

A list contains variable selected `xselect` and training and testing error `err.tr, err.te`.

Author(s)

Zhu Wang

See Also

`cv.mada` for cross-validated stopping iteration.

mhingebst *Boosting for Multi-class Classification*

Description

Gradient boosting for optimizing multi-class hinge loss functions with componentwise linear least squares, smoothing splines and trees as base learners.

Usage

```
mhingebst(x, y, cost = NULL, family = c("hinge"), ctrl = bst_control(),
control.tree = list(fixed.depth=TRUE, n.term.node=6, maxdepth = 1),
learner = c("ls", "sm", "tree"))
## S3 method for class 'mhingebst'
print(x, ...)
## S3 method for class 'mhingebst'
predict(object, newdata=NULL, newy=NULL, mstop=NULL,
type=c("response", "class", "loss", "error"), ...)
## S3 method for class 'mhingebst'
fpartial(object, mstop=NULL, newdata=NULL)
```

Arguments

x	a data frame containing the variables in the model.
y	vector of responses. y must be in {1, -1} for family = "hinge".
cost	equal costs for now and unequal costs will be implemented in the future.
family	family = "hinge" for multi-class hinge loss.
ctrl	an object of class bst_control.
control.tree	control parameters of rpart.
learner	a character specifying the component-wise base learner to be used: ls linear models, sm smoothing splines, tree regression trees.
type	in predict a character indicating whether the response, classes, loss or classification errors should be predicted in case of hinge
object	class of mhingebst.
newdata	new data for prediction with the same number of columns as x.
newy	new response.
mstop	boosting iteration for prediction.
...	additional arguments.

Details

A linear or nonlinear classifier is fitted using a boosting algorithm based on component-wise base learners for multi-class responses.

Value

An object of class `mhingebst` with `print` and `predict` methods being available for fitted models.

Author(s)

Zhu Wang

References

Zhu Wang (2011), HingeBoost: ROC-Based Boost for Classification and Variable Selection. *The International Journal of Biostatistics*, **7**(1), Article 13.

Zhu Wang (2012), Multi-class HingeBoost: Method and Application to the Classification of Cancer Types Using Gene Expression Data. *Methods of Information in Medicine*, **51**(2), 162–7.

See Also

`cv.mhingebst` for cross-validated stopping iteration. Furthermore see `bst_control`

Examples

```
## Not run:
dat <- ex1data(100)
res <- mhingebst(x=dat$x, y=dat$y)

## End(Not run)
```

`mhingeova` *Multi-class HingeBoost*

Description

Multi-class algorithm with one-vs-all binary HingeBoost which optimizes the hinge loss functions with componentwise linear, smoothing splines, tree models as base learners.

Usage

```
mhingeova(xtr, ytr, xte=NULL, yte=NULL, cost = NULL, nu=0.1,
learner=c("tree", "ls", "sm"), maxdepth=1, m1=200, twinboost = FALSE, m2=200)
## S3 method for class 'mhingeova'
print(x, ...)
```

Arguments

xtr	training data containing the predictor variables.
ytr	vector of training data responses. ytr must be in {1,2,...,k}.
xte	test data containing the predictor variables.
yte	vector of test data responses. yte must be in {1,2,...,k}.
cost	default is NULL for equal cost; otherwise a numeric vector indicating price to pay for false positive, 0 < cost < 1; price of false negative is 1-cost.
nu	a small number (between 0 and 1) defining the step size or shrinkage parameter.
learner	a character specifying the component-wise base learner to be used: ls linear models, sm smoothing splines, tree regression trees.
maxdepth	tree depth used in learner=tree
m1	number of boosting iteration
twinboost	logical: twin boosting?
m2	number of twin boosting iteration
x	class of mhingeova.
...	additional arguments.

Details

For a C-class problem (C > 2), each class is separately compared against all other classes with HingeBoost, and C functions are estimated to represent confidence for each class. The classification rule is to assign the class with the largest estimate. A linear or nonlinear multi-class HingeBoost classifier is fitted using a boosting algorithm based on one-against component-wise base learners for +1/-1 responses, with possible cost-sensitive hinge loss function.

Value

An object of class mhingeova with print method being available.

Author(s)

Zhu Wang

References

Zhu Wang (2011), HingeBoost: ROC-Based Boost for Classification and Variable Selection. *The International Journal of Biostatistics*, **7**(1), Article 13.

Zhu Wang (2012), Multi-class HingeBoost: Method and Application to the Classification of Cancer Types Using Gene Expression Data. *Methods of Information in Medicine*, **51**(2), 162–7.

See Also

bst for HingeBoost binary classification. Furthermore see cv.bst for stopping iteration selection by cross-validation, and bst_control for control parameters.

Examples

```
## Not run:
dat1 <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/
thyroid-disease/ann-train.data")
dat2 <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/
thyroid-disease/ann-test.data")
res <- mhingeova(xtr=dat1[,-22], ytr=dat1[,22], xte=dat2[,-22], yte=dat2[,22],
cost=c(2/3, 0.5, 0.5), nu=0.5, learner="ls", m1=100, K=5, cv1=FALSE,
twinboost=TRUE, m2= 200, cv2=FALSE)
res <- mhingeova(xtr=dat1[,-22], ytr=dat1[,22], xte=dat2[,-22], yte=dat2[,22],
cost=c(2/3, 0.5, 0.5), nu=0.5, learner="ls", m1=100, K=5, cv1=FALSE,
twinboost=TRUE, m2= 200, cv2=TRUE)

## End(Not run)
```

Index