Package 'COBRA'

February 19, 2015

Type Package

Title Nonlinear Aggregation of Predictors

Version 0.99.4

Date 2013-07-30

Author Benjamin Guedj <benjamin.guedj@upmc.fr>

Maintainer Benjamin Guedj <benjamin.guedj@upmc.fr>

Description This package performs prediction for regression-oriented problems, aggregating in a nonlinear scheme any basic regression machines suggested by the context and provided by the user. If the user has no valuable knowledge on the data, four defaults machines wrappers are implemented so as to cover a minimal spectrum of prediction methods. If necessary, the computations may be parallelized. The method is described in Biau, Fischer, Guedj and Malley (2013), ``COBRA: A Nonlinear Aggregation Strategy''.

License GPL (>= 2)

URL http://www.lsta.upmc.fr/doct/guedj/index.html

Suggests snowfall, lars, ridge, tree, randomForest

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-07-30 15:14:34

R topics documented:

COBRA-package *Nonlinear Aggregation of Predictors*

Description

The function COBRA delivers prediction outcomes for a testing sample on the basis of a training sample and a bunch of basic regression machines. By default, those machines are wrappers to the R packages `lars`, `ridge`, `tree` and `randomForest`, covering a minimal spectrum in contemporary prediction methods for regression. However the most interesting way to use COBRA is to use any regression method suggested by the context (see argument `machines`). COBRA may natively parallelize the computations (use option `parallel`).

Details

Package:	COBRA
Type:	Package
Version:	0.99.4
Date:	2013-07-30
License:	GPL (>= 2)

Author(s)

Benjamin Guedj

Maintainer: Benjamin Guedj <benjamin.guedj@upmc.fr>

References

http://www.lsta.upmc.fr/doct/guedj/index.html

G. Biau, A. Fischer, B. Guedj and J. D. Malley (2013), COBRA: A Nonlinear Aggregation Strategy. http://arxiv.org/abs/1303.2236 and http://hal.archives-ouvertes.fr/hal-00798579

See Also

`COBRA`

Examples

```
n <- 500
d <- 30
ntrain <- 400
X <- replicate(d,2*runif(n = n)-1)
Y <- X[,1]^2 + X[,3]^3 + exp(X[,10]) + rnorm(n = n, sd = .1)
train.design <- as.matrix(X[1:ntrain,])
```

```
train.responses <- Y[1:ntrain]
test <- as.matrix(X[-(1:ntrain),])
test.responses <- Y[-(1:ntrain)]

## using the default machines
if(require(lars) && require(tree) && require(ridge) &&
require(randomForest))
{
res <- COBRA(train.design = train.design,
             train.responses = train.responses,
             test = test)

print(cbind(res$predict,test.responses))
plot(test.responses,res$predict,xlab="Responses",ylab="Predictions",pch=3,col=2)
abline(0,1,lty=2)
}

## using own machines
machines.names <- c("Soothsayer","Dummy")
machines <- matrix(nr = n, nc = 2, data = 0)
machines[,1] <- Y+rnorm(n = n, sd=.1)          ## soothsayer
machines[,2] <- mean(train.responses)        ## dummy prediction, averaging train.responses

res2 <- COBRA(train.design = train.design,
              train.responses = train.responses,
              test = test,
              machines = machines,
              machines.names = machines.names)

print(cbind(res2$predict,test.responses))
plot(test.responses,res2$predict,xlab="Responses",ylab="Predictions",pch=3,col=2)
abline(0,1,lty=2)
```

COBRA *COBRA*

Description

The function COBRA delivers prediction outcomes for a testing sample on the basis of a training sample and a bunch of basic regression machines. By default, those machines are wrappers to

the R packages `lars`, `ridge`, `tree` and `randomForest`, covering a somewhat wide spectrum in contemporary prediction methods for regression. However the most interesting way to use COBRA is to use any regression method suggested by the context (see argument `machines`). COBRA may natively parallelize the computations (use option `parallel`).

Usage

```
COBRA(train.design,
      train.responses,
      split,
      test,
      machines,
      machines.names,
      logGrid = FALSE,
      grid = 200,
      alpha.machines,
      parallel = FALSE,
      nb.cpus = 2,
      plots = FALSE,
      savePlots = FALSE,
      logs = FALSE,
      progress = TRUE,
      path = "")
```

Arguments

`train.design`	Mandatory. The design matrix for the training sample.
`train.responses`	Mandatory. The responses vector for the training sample.
`split`	Optional. How should COBRA cut the training sample?
`test`	Mandatory. The design matrix of the testing sample.
`machines`	Optional. Regression basic machines provided by the user. This should be a matrix, whose number of rows is the length of the training sample (ntrain) plus the length of the testing sample (ntest), and with as many columns as machines. Element (i,j) of this matrix is assumed to be r_j(X_i), the (scalar) prediction of machine j for query point X_i, where i is from 1 to ntrain+ntest.
`machines.names`	Optional. If `machines` is provided, a list including the names of the machines.
`logGrid`	Optional. If `TRUE`, parameter epsilon is generated according to a logarithmic scale. This should be `TRUE` if the user has a clue about the small magnitude of predictions.
`grid`	Optional. How many points should be used in the discretization scheme for calibrating the parameter epsilon.
`alpha.machines`	Optional. Coerce COBRA to use exactly `alpha.machines`. Obviously this should be a integer between 1 and the total number of machines.
`parallel`	Optional. If `TRUE`, computations will be dispatched over available cpus.

nb.cpus	Optional. If parallel, how many cpus should be used. Obviously this should not exceed the number of available cpus!
plots	Optional. If TRUE, explanatory plots about calibrating epsilon and alpha (see publication) are generated according to the path variable.
savePlots	Optional. If TRUE, plots are saved as .pdf files according to path, otherwise they pop up in the R IDE.
logs	Optional. If TRUE, quadratic risks over the training sample for all machines and COBRA are written in the file "risks.txt" according to the path variable.
progress	Optional. If TRUE, a progress bar and final quadratic errors are printed.
path	Optional. If savePlots and either plots or logs are TRUE, where should the corresponding files be created?

Details

For most users, options grid and split should be set to their default values.

Value

Returns a list including only

predict	The vector of predicted values.

Note

Caution: If your data is ordered, you should shuffle the observations before calling COBRA since the algorithm assumes all data points are independent and identically distributed.

Author(s)

Benjamin Guedj <benjamin.guedj@upmc.fr>

References

http://www.lsta.upmc.fr/doct/guedj/index.html

G. Biau, A. Fischer, B. Guedj and J. D. Malley (2013), COBRA: A Nonlinear Aggregation Strategy. http://arxiv.org/abs/1303.2236 and http://hal.archives-ouvertes.fr/hal-00798579

See Also

COBRA-package

Examples

```
n <- 500
d <- 30
ntrain <- 400
X <- replicate(d,2*runif(n = n)-1)
Y <- X[,1]^2 + X[,3]^3 + exp(X[,10]) + rnorm(n = n, sd = .1)
train.design <- as.matrix(X[1:ntrain,])
```

```
train.responses <- Y[1:ntrain]
test <- as.matrix(X[-(1:ntrain),])
test.responses <- Y[-(1:ntrain)]

## using the default machines
if(require(lars) && require(tree) && require(ridge) &&
require(randomForest))
{
res <- COBRA(train.design = train.design,
             train.responses = train.responses,
             test = test)

print(cbind(res$predict,test.responses))
plot(test.responses,res$predict,xlab="Responses",ylab="Predictions",pch=3,col=2)
abline(0,1,lty=2)
}

## using own machines
machines.names <- c("Soothsayer","Dummy")
machines <- matrix(nr = n, nc = 2, data = 0)
machines[,1] <- Y+rnorm(n = n, sd=.1)          ## soothsayer
machines[,2] <- mean(train.responses)        ## dummy prediction, averaging train.responses

res2 <- COBRA(train.design = train.design,
              train.responses = train.responses,
              test = test,
              machines = machines,
              machines.names = machines.names)

print(cbind(res2$predict,test.responses))
plot(test.responses,res2$predict,xlab="Responses",ylab="Predictions",pch=3,col=2)
abline(0,1,lty=2)
```

Index