Package: BOSSreg (via r-universe)

October 31, 2024

Type Package

Title Best Orthogonalized Subset Selection (BOSS)

Version 0.2.0

Date 2021-3-6

Maintainer Sen Tian <stian@stern.nyu.edu>

Description Best Orthogonalized Subset Selection (BOSS) is a least-squares (LS) based subset selection method, that performs best subset selection upon an orthogonalized basis of ordered predictors, with the computational effort of a single ordinary LS fit. This package provides a highly optimized implementation of BOSS and estimates a heuristic degrees of freedom for BOSS, which can be plugged into an information criterion (IC) such as AICc in order to select the subset from candidates. It provides various choices of IC, including AIC, BIC, AICc, Cp and GCV. It also implements the forward stepwise selection (FS) with no additional computational cost, where the subset of FS is selected via cross-validation (CV). CV is also an option for BOSS. For details see: Tian, Hurvich and Simonoff (2021), ``On the Use of Information Criteria for Subset Selection in Least Squares Regression", <arXiv:1911.10191>.

Depends R (>= 3.5.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports glmnet, Matrix, Rcpp, stats

RoxygenNote 7.1.1

Suggests devtools, ISLR, kableExtra, knitr, MASS, rmarkdown, sparsenet

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/sentian/BOSSreg

BugReports https://github.com/sentian/BOSSreg/issues

NeedsCompilation yes

Author Sen Tian [aut, cre], Clifford Hurvich [aut], Jeffrey Simonoff [aut]

Repository CRAN

Date/Publication 2021-03-06 19:20:02 UTC

Contents

`boss` *Best Orthogonalized Subset Selection (BOSS).*

Description

- Compute the solution path of BOSS and forward stepwise selection (FS).
- Compute various information criteria based on a heuristic degrees of freedom (hdf) that can serve as the selection rule to choose the subset given by BOSS.

Usage

```
boss(
  x,
  y,
  maxstep = min(nrow(x) - intercept - 1, ncol(x)),
  intercept = TRUE,
  hdf.ic.boss = TRUE,
  mu = NULL,
  sigma = NULL,
  ...
)
```

Arguments

`x`	A matrix of predictors, with `nrow(x)=length(y)=n` observations and `ncol(x)=p` predictors. Intercept shall NOT be included.
`y`	A vector of response variable, with `length(y)=n`.

maxstep	Maximum number of steps performed. Default is min(n-1,p) if intercept=FALSE, and it is min(n-2, p) otherwise.
intercept	Logical, whether to include an intercept term. Default is TRUE.
hdf.ic.boss	Logical, whether to calculate the heuristic degrees of freedom (hdf) and information criteria (IC) for BOSS. IC includes AIC, BIC, AICc, BICc, GCV, Cp. Default is TRUE.
mu	True mean vector, used in the calculation of hdf. Default is NULL, and is estimated via least-squares (LS) regression of y upon x for n>p, and 10-fold CV cross-validated lasso estimate for n<=p.
sigma	True standard deviation of the error, used in the calculation of hdf. Default is NULL, and is estimated via least-squares (LS) regression of y upon x for n>p, and 10-fold cross-validated lasso for n<=p.
...	Extra parameters to allow flexibility. Currently none allows or requires, just for the convinience of call from other parent functions like cv.boss.

Details

This function computes the full solution path given by BOSS and FS on a given dataset (x,y) with n observations and p predictors. It also calculates the heuristic degrees of freedom for BOSS, and various information criteria, which can further be used to select the subset from the candidates. Please refer to the Vignette for implementation details and Tian et al. (2021) for methodology details (links are given below).

Value

- beta_fs: A matrix of regression coefficients for all the subsets given by FS, from a null model until stop, with nrow=p and ncol=min(n,p)+1, where min(n,p) is the maximum number of steps performed.
- beta_boss: A matrix of regression coefficients for all the subsets given by BOSS, with nrow=p and ncol=min(n,p)+1. Note that unlike beta_fs and due to the nature of BOSS, the number of non-zero components in columns of beta_boss may not be unique, i.e. there maybe multiple columns corresponding to the same size of subset.
- steps_x: A vector of numbers representing which predictor joins at each step, with length(steps)=min(n,p). The ordering is determined by the partial correlation between a predictor x_j and the response y.
- steps_q: A vector of numbers representing which predictor joins at each step in the orthogonal basis, with length(steps)=min(n,p). BOSS takes the ordered predictors (ordering given in steps_x) and performs best subset regression upon their orthogonal basis, which is essentially ordering the orthogonalized predictors by their marginal correlations with the response y. For example, steps_q=c(2,1) indicates that the orthogonal basis of x_2 joins first.
- hdf_boss: A vector of heuristic degrees of freedom (hdf) for BOSS, with length(hdf_boss)=p+1. Note that hdf_boss=NULL if n<=p or hdf.ic.boss=FALSE.
- IC_boss: A list of information criteria (IC) for BOSS, where each element in the list is a vector representing values of a given IC for each candidate subset of BOSS (or each column in beta_boss). The output IC includes AIC, BIC, AICc, BICc, GCV and Mallows' Cp. Note that each IC is calculated by plugging in hdf_boss.

- sigma: estimated error standard deviation. It is only returned when hdf is calculated, i.e. `hdf.ic.boss=TRUE`.

Author(s)

Sen Tian

References

- Tian, S., Hurvich, C. and Simonoff, J. (2021), On the Use of Information Criteria for Subset Selection in Least Squares Regression. https://arxiv.org/abs/1911.10191
- Reid, S., Tibshirani, R. and Friedman, J. (2016), A Study of Error Variance Estimation in Lasso Regression. Statistica Sinica, P35-67, JSTOR.
- BOSSreg Vignette https://github.com/sentian/BOSSreg/blob/master/r-package/vignettes/BOSSreg.pdf

See Also

`predict` and `coef` methods for "boss" object, and the `cv.boss` function

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(n, sd=0.01), center = TRUE, scale = FALSE)

## Fit the model
boss_result = boss(x, y)

## Get the coefficient vector selected by AICc-hdf (S3 method for boss)
beta_boss_aicc = coef(boss_result)
# the above is equivalent to the following
beta_boss_aicc = boss_result$beta_boss[, which.min(boss_result$IC_boss$aicc), drop=FALSE]
## Get the fitted values of BOSS-AICc-hdf (S3 method for boss)
mu_boss_aicc = predict(boss_result, newx=x)
# the above is equivalent to the following
mu_boss_aicc = cbind(1,x) %*% beta_boss_aicc

## Repeat the above process, but using Cp-hdf instead of AICc-hdf
## coefficient vector
beta_boss_cp = coef(boss_result, method.boss='cp')
beta_boss_cp = boss_result$beta_boss[, which.min(boss_result$IC_boss$cp), drop=FALSE]
## fitted values of BOSS-Cp-hdf
mu_boss_cp = predict(boss_result, newx=x, method.boss='cp')
mu_boss_cp = cbind(1,x) %*% beta_boss_cp
```

`calc.ic` *Calculate an information criterion.*

Description

Calculate a specified information criterion (IC) for an estimate or a group of estimates. The choices of IC include AIC, BIC, AICc, BICc, GCV and Mallows' Cp.

Usage

```
calc.ic(
  y_hat,
  y,
  ic = c("aicc", "bicc", "aic", "bic", "gcv", "cp"),
  df,
  sigma = NULL
)
```

Arguments

`y_hat`	A vector of fitted values with `length(y_hat)=length(y)=n`, or a matrix, with `nrow(coef)=length(y)=n` and `ncol(y_hat)=m`, containing m different fits.
`y`	A vector of response variable, with `length(y)=n`.
`ic`	A specified IC to calculate. Default is AICc ('aicc'). Other choices include AIC ('aic'), BIC ('bic'), BICc ('bicc'), GCV ('gcv') and Mallows' Cp ('cp').
`df`	A number if y_hat is a vector, or a vector with `length(df)=ncol(y_hat)=m` if y_hat is a matrix. df represents the degrees of freedom for each fit.
`sigma`	Standard deviation of the error term. It only needs to be specified if the argument `ic='cp'`.

Details

This function enables the computation of various common IC for model fits, which can further be used to choose the optimal fit. This allows user comparing the effect of different IC. In order to calculate an IC, degrees of freedoms (df) needs to be specified. To be more specific, here are the formulas used to calculate each IC:

$$AIC = \log(\frac{RSS}{n}) + 2\frac{df}{n}$$

$$BIC = \log(\frac{RSS}{n}) + \log(n)\frac{df}{n}$$

$$AICc = \log(\frac{RSS}{n}) + 2\frac{df+1}{n-df-2}$$

$$BICc = \log(\frac{RSS}{n}) + \log(n)\frac{df+1}{n-df-2}$$

$$GCV = \frac{RSS}{(n - df)^2}$$

$$Mallows'Cp = RSS + 2 \times \sigma^2 \times df$$

Value

The value(s) of the specified IC for each fit.

Author(s)

Sen Tian

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(20, sd=0.01), center = TRUE, scale = FALSE)

## Fit the model
boss_result = boss(x, y)
## Print the values of AICc-hdf for all subsets given by BOSS
print(boss_result$IC_boss$aicc)
## calculate them manually using the calc.ic function
y_hat = cbind(rep(1,n),x)%*%boss_result$beta_boss
print(calc.ic(y_hat, y, df=boss_result$hdf_boss))
```

`coef.boss` *Select coefficient vector(s) for BOSS.*

Description

This function returns the optimal coefficient vector of BOSS selected by AICc (by default) or other types of information criterion.

Usage

```
## S3 method for class 'boss'
coef(
  object,
  ic = c("aicc", "bicc", "aic", "bic", "gcv", "cp"),
  select.boss = NULL,
  ...
)
```

Arguments

object	The boss object, returned from calling the boss function.
ic	Which information criterion is used to select the optimal coefficient vector for BOSS. The default is AICc-hdf.
select.boss	The index (or indicies) of columns in the coefficient matrix for which one wants to select. By default (NULL) it's selected by the information criterion specified in 'ic'.
...	Extra arguments (unused for now)

Details

If select.boss is specified, the function returns corresponding column(s) in the coefficient matrix.

If select.boss is unspecified, the function returns the optimal coefficient vector selected by AICc-hdf (other choice of IC can be specified in the argument ic).

Value

The chosen coefficient vector(s) for BOSS.

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(n, sd=0.01), center = TRUE, scale = FALSE)

## Fit the model
boss_result = boss(x, y)

## Get the coefficient vector selected by AICc-hdf (S3 method for boss)
beta_boss_aicc = coef(boss_result)
# the above is equivalent to the following
beta_boss_aicc = boss_result$beta_boss[, which.min(boss_result$IC_boss$aicc), drop=FALSE]
## Get the fitted values of BOSS-AICc-hdf (S3 method for boss)
mu_boss_aicc = predict(boss_result, newx=x)
# the above is equivalent to the following
mu_boss_aicc = cbind(1,x) %*% beta_boss_aicc

## Repeat the above process, but using Cp-hdf instead of AICc-hdf
## coefficient vector
beta_boss_cp = coef(boss_result, method.boss='cp')
beta_boss_cp = boss_result$beta_boss[, which.min(boss_result$IC_boss$cp), drop=FALSE]
## fitted values of BOSS-Cp-hdf
mu_boss_cp = predict(boss_result, newx=x, method.boss='cp')
mu_boss_cp = cbind(1,x) %*% beta_boss_cp
```

`coef.cv.boss` *Select coefficient vector based on cross-validation for BOSS or FS.*

Description

This function returns coefficient vector that minimizes out-of-sample (OOS) cross validation score.

Usage

```
## S3 method for class 'cv.boss'
coef(object, method = c("boss", "fs"), ...)
```

Arguments

`object`	The cv.boss object, returned from calling `cv.boss` function.
`method`	It can either be 'fs' or 'boss'. The default is 'boss'.
`...`	Extra arguments (unused for now).

Value

The chosen coefficient vector for BOSS or FS.

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(20, sd=0.01), center = TRUE, scale = FALSE)

## Perform 10-fold CV without replication
boss_cv_result = cv.boss(x, y)
## Get the coefficient vector of BOSS that gives minimum CV OSS score (S3 method for cv.boss)
beta_boss_cv = coef(boss_cv_result)
# the above is equivalent to
boss_result = boss_cv_result$boss
beta_boss_cv = boss_result$beta_boss[, boss_cv_result$i.min.boss, drop=FALSE]
## Get the fitted values of BOSS-CV (S3 method for cv.boss)
mu_boss_cv = predict(boss_cv_result, newx=x)
# the above is equivalent to
mu_boss_cv = cbind(1,x) %*% beta_boss_cv

## Get the coefficient vector of FS that gives minimum CV OSS score (S3 method for cv.boss)
beta_fs_cv = coef(boss_cv_result, method='fs')
## Get the fitted values of FS-CV (S3 method for cv.boss)
mu_fs_cv = predict(boss_cv_result, newx=x, method='fs')
```

`cv.boss` *Cross-validation for Best Orthogonalized Subset Selection (BOSS) and Forward Stepwise Selection (FS).*

Description

Cross-validation for Best Orthogonalized Subset Selection (BOSS) and Forward Stepwise Selection (FS).

Usage

```
cv.boss(
  x,
  y,
  maxstep = min(nrow(x) - intercept - 1, ncol(x)),
  intercept = TRUE,
  n.folds = 10,
  n.rep = 1,
  show.warning = TRUE,
  ...
)
```

Arguments

`x`	A matrix of predictors, see `boss`.
`y`	A vector of response variable, see `boss`.
`maxstep`	Maximum number of steps performed. Default is `min(n-1,p)` if `intercept=FALSE`, and it is `min(n-2, p)` otherwise.
`intercept`	Logical, whether to fit an intercept term. Default is TRUE.
`n.folds`	The number of cross validation folds. Default is 10.
`n.rep`	The number of replications of cross validation. Default is 1.
`show.warning`	Whether to display a warning if CV is only performed for a subset of candidates. e.g. when n<p and 10-fold. Default is TRUE.
`...`	Arguments to `boss`, such as `hdf.ic.boss`.

Details

This function fits BOSS and FS (`boss`) on the full dataset, and performs `n.folds` cross-validation. The cross-validation process can be repeated `n.rep` times to evaluate the out-of-sample (OOS) performance for the candidate subsets given by both methods.

Value

- boss: An object boss that fits on the full dataset.
- n.folds: The number of cross validation folds.
- cvm.fs: Mean OOS deviance for each candidate given by FS.
- cvm.boss: Mean OSS deviance for each candidate given by BOSS.
- i.min.fs: The index of minimum cvm.fs.
- i.min.boss: The index of minimum cvm.boss.

Author(s)

Sen Tian

References

- Tian, S., Hurvich, C. and Simonoff, J. (2021), On the Use of Information Criteria for Subset Selection in Least Squares Regression. https://arxiv.org/abs/1911.10191
- BOSSreg Vignette https://github.com/sentian/BOSSreg/blob/master/r-package/vignettes/BOSSreg.pdf

See Also

predict and coef methods for cv.boss object, and the boss function

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(20, sd=0.01), center = TRUE, scale = FALSE)

## Perform 10-fold CV without replication
boss_cv_result = cv.boss(x, y)
## Get the coefficient vector of BOSS that gives minimum CV OSS score (S3 method for cv.boss)
beta_boss_cv = coef(boss_cv_result)
# the above is equivalent to
boss_result = boss_cv_result$boss
beta_boss_cv = boss_result$beta_boss[, boss_cv_result$i.min.boss, drop=FALSE]
## Get the fitted values of BOSS-CV (S3 method for cv.boss)
mu_boss_cv = predict(boss_cv_result, newx=x)
# the above is equivalent to
mu_boss_cv = cbind(1,x) %*% beta_boss_cv

## Get the coefficient vector of FS that gives minimum CV OSS score (S3 method for cv.boss)
beta_fs_cv = coef(boss_cv_result, method='fs')
## Get the fitted values of FS-CV (S3 method for cv.boss)
mu_fs_cv = predict(boss_cv_result, newx=x, method='fs')
```

predict.boss *Prediction given new data entries.*

Description

This function returns the prediction(s) given new observation(s), for BOSS, where the optimal coefficient vector is chosen via certain selection rule.

Usage

```
## S3 method for class 'boss'
predict(object, newx, ...)
```

Arguments

object	The boss object, returned from calling 'boss' function.
newx	A new data entry or several entries. It can be a vector, or a matrix with nrow(newx) being the number of new entries and ncol(newx)=p being the number of predictors. The function takes care of the intercept, NO need to add 1 to newx.
...	Extra arguments to be plugged into coef, such as select.boss, see the description of coef.boss for more details.

Details

The function basically calculates $x * coef$, where coef is a coefficient vector chosen by a selection rule. See more details about the default and available choices of the selection rule in the description of coef.boss.

Value

The prediction(s) for BOSS.

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(n, sd=0.01), center = TRUE, scale = FALSE)

## Fit the model
boss_result = boss(x, y)

## Get the coefficient vector selected by AICc-hdf (S3 method for boss)
```

```
beta_boss_aicc = coef(boss_result)
# the above is equivalent to the following
beta_boss_aicc = boss_result$beta_boss[, which.min(boss_result$IC_boss$aicc), drop=FALSE]
## Get the fitted values of BOSS-AICc-hdf (S3 method for boss)
mu_boss_aicc = predict(boss_result, newx=x)
# the above is equivalent to the following
mu_boss_aicc = cbind(1,x) %*% beta_boss_aicc

## Repeat the above process, but using Cp-hdf instead of AICc-hdf
## coefficient vector
beta_boss_cp = coef(boss_result, method.boss='cp')
beta_boss_cp = boss_result$beta_boss[, which.min(boss_result$IC_boss$cp), drop=FALSE]
## fitted values of BOSS-Cp-hdf
mu_boss_cp = predict(boss_result, newx=x, method.boss='cp')
mu_boss_cp = cbind(1,x) %*% beta_boss_cp
```

`predict.cv.boss` *Prediction given new data entries.*

Description

This function returns the prediction(s) given new observation(s) for BOSS or FS, where the optimal coefficient vector is chosen via cross-validation.

Usage

```
## S3 method for class 'cv.boss'
predict(object, newx, ...)
```

Arguments

`object`	The cv.boss object, returned from calling `cv.boss` function.
`newx`	A new data entry or several entries. It can be a vector, or a matrix with `nrow(newx)` being the number of new entries and `ncol(newx)=p` being the number of predictors. The function takes care of the intercept, NO need to add `1` to `newx`.
`...`	Extra arguments to be plugged into `coef`, such as `method`, see the description of `coef.cv.boss` for more details.

Value

The prediction for BOSS or FS.

Examples

```
## Generate a trivial dataset, X has mean 0 and norm 1, y has mean 0
set.seed(11)
n = 20
p = 5
x = matrix(rnorm(n*p), nrow=n, ncol=p)
```

```
x = scale(x, center = colMeans(x))
x = scale(x, scale = sqrt(colSums(x^2)))
beta = c(1, 1, 0, 0, 0)
y = x%*%beta + scale(rnorm(20, sd=0.01), center = TRUE, scale = FALSE)

## Perform 10-fold CV without replication
boss_cv_result = cv.boss(x, y)
## Get the coefficient vector of BOSS that gives minimum CV OSS score (S3 method for cv.boss)
beta_boss_cv = coef(boss_cv_result)
# the above is equivalent to
boss_result = boss_cv_result$boss
beta_boss_cv = boss_result$beta_boss[, boss_cv_result$i.min.boss, drop=FALSE]
## Get the fitted values of BOSS-CV (S3 method for cv.boss)
mu_boss_cv = predict(boss_cv_result, newx=x)
# the above is equivalent to
mu_boss_cv = cbind(1,x) %*% beta_boss_cv

## Get the coefficient vector of FS that gives minimum CV OSS score (S3 method for cv.boss)
beta_fs_cv = coef(boss_cv_result, method='fs')
## Get the fitted values of FS-CV (S3 method for cv.boss)
mu_fs_cv = predict(boss_cv_result, newx=x, method='fs')
```

Index