Package ‘cvfreeze’

December 6, 2014

Type Package

Title Cross-validation freezing for lasso-type problems

Version 1.0

Date 2013-06-28

Author Linn Cecilie Bergersen

Maintainer Linn Cecilie Bergersen <linncb@math.uio.no>

Description Procedures to compute and discover cross-validation freezing patterns to guide preselection in ultra high-dimensional lasso regression problems.

License GPL (>=2)

Depends glmnet, R (>= 2.10)

R topics documented:

cvfreeze-package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
cvfreeze . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
cvfreeze.pattern . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
init.order . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
plot.cvfreeze . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
simulated_data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
summary.cvfreeze . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

---

cvfreeze-package Cross-validation freezing for preselection in lasso-type problems

Description

The package provides procedures for computing, discovering and investigating cross-validation freezing patterns to guide preselection in ultra high-dimensional lasso regression problems. This includes algorithms to find the lasso solution of high-dimensional regression problems based on a small fraction of the covariates only. The algorithms make use of sequential computation of cross-validation curves for increasing subsets of the covariates and automatically determine when all relevant variables are most likely included in the subsets based on suitable stopping rules as described in Bergersen et al. (2014).
Details

Package: cvfreeze
Type: Package
Version: 1.0
Date: 2013-06-28
License: GPL (>=2)

Functions take X, y data for linear regression models as input.

Main function `cvfreeze` sequentially computes cross-validation curves and determines cross-validation freezing with two options for stopping.

Function `cvfreeze.pattern` computes the full cross-validation pattern for the sequence supplied for the increasing subsets of covariates.

Output of both functions is a `cvfreeze` object. `plot.cvfreeze` and `summary.cvfreeze` are suitable functions for summarizing `cvfreeze` objects.

Author(s)

Linn Cecilie Bergersen Maintainer: Linn Cecilie Bergersen <linncb@math.uio.no>

References


See Also

`<glmnet>` package. Functions in the `cvfreeze` package calls procedures in the `glmnet` package for computation of the lasso solutions and cross-validation curves. Users should be familiar with options and output of functions in the `glmnet` package.

Examples

```r
library(cvfreeze)
data(simulated_data)
attach(simulated_data)
#curves frozen before maximum number of covariates (given by steps) are included
foldid <- rep(1:10, length.out = nrow(X.ordered))
steps.vector1 <- c(500,1000,1500,2000,2500,3000,5000,10000)
cvfreezel <- cvfreeze(y=y, X=X.ordered, steps = steps.vector1, fold = foldid)
summary(cvfreezel)
plot(cvfreezel)
plot(cvfreezel, ylim = c(100,130), add.legend = FALSE) #zoom, no legend
#curves not frozen before maximum number given by steps are included
steps.vector2 <- c(500, 1000, 1500, 2000, 2500)
cvfreezel2 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector2, fold = foldid)
summary(cvfreezel2)
plot(cvfreezel2)
```
cvfreeze  A function to discover freezing patterns and compute the solution for (ultra) high-dimensional lasso problems.

Description

The function uses Algorithm Part 1 or Part 2 of Bergersen et al. (2014) to sequentially compute freezing patterns in (ultra) high-dimensional lasso problems with continuous response. If a frozen minimum is revealed, the lasso solution in this minimum is returned.

Usage

cvfreeze(y, X, steps, ordered = TRUE, lambda.vec = NULL, K = 10, fold, reorder = FALSE, reorder.step = NULL)

Arguments

y Continuous response variable.
X (Ordered) covariate matrix of dimension n x P where each row is an observation vector. Columns of X should be ordered by desired criterion prior to calling cvfreeze. Otherwise ordered should be FALSE.
steps Number equal to a fixed increment or vector of steps to be used in the algorithm to sequentially increase the subsets.
ordered Logical value. If TRUE, the covariate matrix X is already ordered. If FALSE, the covariate matrix will be ordered by the marginal correlation with the response y. Default is TRUE.
lambda.vec Optional user supplied grid for the penalization parameter lambda. Default is NULL, for which the grid is chosen by the procedures in the glmnet package.
K Number of folds, default is K = 10.
fold Optional vector of values between 1 and K identifying what fold each observation is in. If supplied K can be missing. If not supplied observations are randomly allocated to each fold.
reorder Logical value indicating whether Part 1 or Part 2 of the algorithm should be used. If TRUE, Part 2 with an additional reordering step is used. If FALSE, only Part 1 of the algorithm is used. Default is reorder = FALSE.
reorder.step If reorder = TRUE, the number of covariates to add in the additional reordering step of Part 2 of the algorithm must be provided. If steps is provided as a fixed increment and reorder.step = NULL, steps is used also for Part 2 of the algorithm.

Details

The function sequentially computes the cross-validation curves for the lasso to discover freezing patterns and to obtain the full lasso solution of ultra high-dimensional regression problems based on a small subset of the covariates. It is recommended that the columns of the covariate matrix X is ordered prior to calling cvfreeze. The algorithm stops based on suitable stopping criteria given by Part 1 and Part 2 of the algorithm described in Bergersen et al. (2014). Note that if fold is not supplied, observations are randomly allocated to each fold and might cause slightly different solutions from run to run. The function calls cv.glmnet in the glmnet package for the computation of the cross-validation curves.
Value

An object of class "cvfreeze" is returned, which is a list with

- `cv`: Matrix where each row represents the cross-validation curve (error) for each value in `steps` until the algorithm stops.
- `lambda`: The values of lambda used in the fits (same for each curve in `cv`).
- `steps`: Vector of how many covariates were included at each sequential step of the algorithm before it stops.
- `frozen`: Logical value. If TRUE, a solution is obtained from the algorithm before all covariates (or maximum number provided for `steps`) are included in the analysis.
- `lambda.freeze`: The value of lambda for which a frozen minimum is found. If no frozen minimum is found before the maximum number of covariates supplied in `steps` is included, `lambda.freeze` is the value of lambda for minimizing the last curve.
- `m.freeze`: The number of curves needed before the full solution can be revealed (depends on `steps`)
- `sol.freeze`: Vector of regression coefficients for the solution obtained using `lambda.freeze`.

Author(s)

Linn Cecilie Bergersen <linncb@math.uio.no>

References


See Also

`plot` and `summary` methods for `cvfreeze` objects.

Examples

```r
library(cvfreeze)
data(simulated_data)attach(simulated_data)
# fixed increments and unordered covariate matrix:
cvfreeze1 <- cvfreeze(y=y, X=X, steps = 1000, ordered = FALSE)
# fixed increments and already ordered covariates:
cvfreeze2 <- cvfreeze(y=y, X=X.ordered, steps = 1000)
# predefined steps sequence and folds:
steps.vector <- c(500,1000,1500,2000,2500,3000,5000,10000)
foldid <- rep(1:10, length.out = nrow(X.ordered))
cvfreeze3 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector, fold = foldid)
# with reordering step in Part 2 of algorithm
cvfreeze4 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector, fold = foldid, reorder = TRUE, reorder.step = 1000)
```
cvfreeze.pattern  A function to compute full freezing pattern for lasso regression problems

Description
Computes the full cross-validation pattern for the sequence supplied for the increasing subsets of covariates.

Usage
.cvfreeze.pattern(y, X, steps, ordered = TRUE, lambda.vec = NULL, K = 10, fold)

Arguments

y Continuous response variable.
X (Ordered) covariate matrix of dimension n x P where each row is an observation vector. Columns of X should be ordered by desired criterion prior to calling cvfreeze. Otherwise ordered should be FALSE.
steps Number equal to a fixed increment or vector of steps to be used in the algorithm to sequentially increase the subsets.
ordered Logical value. If TRUE, the covariate matrix X is already ordered. If FALSE, the covariate matrix will be ordered by the marginal correlation with the response y. Default is TRUE.
lambda.vec Optional user supplied grid for the penalization parameter lambda. Default is NULL, for which the grid is chosen by the procedures in the glmnet package.
K Number of folds, default is K = 10.
fold Optional vector of values between 1 and K identifying what fold each observation is in. If supplied K can be missing. If not supplied observations are randomly allocated to each fold.

Details
The function sequentially computes the cross-validation curves for the lasso for all values supplied through steps.

Value
an object of class "cvfreeze" is returned, which is a list with
cv Matrix where each row represents the cross-validation curve (error) for each value in steps.
lambda the values of lambda used in the fits (same for each curve in cv).
steps Vector of how many covariates were included at each sequential step.
frozen NULL. No procedure for determining when the solution is frozen. See cvfreeze function instead.
lambda.freeze The optimal value of lambda for the full data set using all covariates.
m.freeze NULL
sol.freeze Vector of regression coefficients for the solution obtained using lambda.freeze and the full data set.
init.order

A function to order the covariates according to their marginal correlation with the response.

Description

Function to order the columns of an input covariate matrix $X$ according to the marginal correlation with the response $y$. Returns a matrix of the same dimension as the input covariate matrix where the covariates are arranged according to the marginal correlation with the response such that the first column is the covariate with the largest marginal correlation with the response and the last column is the covariate with the least marginal correlation with the response.

Usage

```r
init.order(X, y, crit="univ")
```

Arguments

- **X**
  - Unordered covariate matrix of dimension $n \times P$ and each row is an observation vector.
- **y**
  - Continuous response variable
- **crit**
  - Criterion for the covariates to be ordered, only "univ" corresponding to marginal correlation with the response is possible.

Value

A covariate matrix where the columns are ordered based on the marginal correlation between each covariate and the response. The covariate with the largest marginal correlation with the response is stored in the first column with the other covariates following and the least correlated covariates in the last column.

Author(s)

Linn Cecilie Bergersen <linncb@math.uio.no>
Examples

```r
data(simulated_data)
attach(simulated_data)
# covariate matrix with columns ordered by the marginal correlation
# with the response:
ordered.covmatrix <- init.order(X=X, y=y)
```

Description

Plots the cross-validation curves of sequentially increasing subsets of the data as contained in a `cvfreeze` object.

Usage

```r
plot.cvfreeze(x, add.legend = TRUE, main = NULL, ylim = NULL, ...)
```

Arguments

- `x` A `cvfreeze` object
- `add.legend` Logical value. If TRUE, a legend is automatically added. FALSE otherwise. Default is TRUE.
- `main` Figure title. Default is NULL generating a standard title with information on how many covariates are used.
- `ylim` Limits for y axis of plot. If not specified, limits are chosen such that all curves are visible. Zooms by specifying `ylim` can be convenient.
- `...` Other graphical parameters to plot.

Details

A plot is produced, and nothing is returned. The appearance of the plot depends on whether freezing is discovered when the `cvfreeze` object is constructed or whether the `cvfreeze` object contains the full freezing pattern (including curves computed after freezing could have been discovered).

For `cvfreeze` objects obtained by the `cvfreeze` function: If the `frozen` component of the `cvfreeze` object is TRUE, only curves up until the freezing is discovered is plotted. Green color indicates the curves for which freezing is discovered, while red color indicates that freezing is not discovered. If the `frozen` component of the `cvfreeze` object is FALSE, curves for all the steps used as input to the `cvfreeze` function are plotted and the cross-validation curve for the largest subset of the data are indicated by a black line.

For `cvfreeze` objects obtained by the `cvfreeze.pattern` function: All curves are plotted. The black line corresponds to the cross-validation curve for the full data set of all covariates.

Author(s)

Linn Cecilie Bergersen <linncb@math.uio.no>
Examples

```r
library(cvfreeze)
data(simulated_data)
attach(simulated_data)
# curves frozen before maximum number of covariates (given by steps) are included
foldid <- rep(1:10, length.out = nrow(X.ordered))
steps.vector1 <- c(500,1000,1500,2000,2500,3000,5000,10000)
cvfreeze1 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector1, fold = foldid)
plot(cvfreeze1)
plot(cvfreeze1, ylim = c(100,130), add.legend = FALSE) # zoom, no legend
# curves not frozen before maximum number given by steps are included
steps.vector2 <- c(500, 1000, 1500, 2000, 2500)
cvfreeze2 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector2, fold = foldid)
plot(cvfreeze2)
cvfreeze3 <- cvfreeze.pattern(y=y, X=X.ordered, steps = 2000)
plot(cvfreeze3)
```

**simulated_data**

**Simulated data set**

**Description**

Data set generated as described in Bergersen et al. (2014), scenario B.

**Usage**

data(simulated_data)

**Format**

A list object containing two data matrices and one response vector.

- **X**: covariate matrix, n=200, P=10000
- **X.ordered**: ordered covariate matrix X, n = 200, P = 10000
- **y**: response vector (n=200)

**Details**

The data are randomly generated as blocks of Toeplitz correlated features and are intended only for illustration. The covariates were simulated according to a multivariate standard normal distribution with variables correlated in blocks of 100 covariates. Within each block the pairwise correlation between the j-th and the k-th covariate is given by 0.9^|j-k|. Between blocks the data are simulated independently.

**References**


**Examples**

data(simulated_data)
summary.cvfreeze

Prints a summary of a cvfreeze object

Description

Prints a summary of a cvfreeze objects including information on whether the cross-validation curves were found to be frozen or not, and information about the final lasso solution obtained.

Usage

summary.cvfreeze(object)

Arguments

object A cvfreeze object

Author(s)

Linn Cecilie Bergersen <linncb@math.uio.no>

Examples

library(cvfreeze)
data(simulated_data)
attach(simulated_data)
# curves frozen before maximum number of covariates (given by steps) are included
foldid <- rep(1:10, length.out = nrow(X.ordered))
steps.vector1 <- c(500, 1000, 1500, 2000, 2500, 3000, 5000, 10000)
cvfreeze1 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector1, fold = foldid)
summary(cvfreeze1)
# curves not frozen before maximum number given by steps are included
steps.vector2 <- c(500, 1000, 1500, 2500)
cvfreeze2 <- cvfreeze(y=y, X=X.ordered, steps = steps.vector2, fold = foldid)
summary(cvfreeze2)