Package ‘HDPenReg’

April 10, 2019

Version 0.94.6
Date 2019-03-29
Encoding UTF-8
Title High-Dimensional Penalized Regression
Copyright inria
Depends R (>= 3.0.2), rtkore (>= 1.5.5)
Imports methods, Matrix
Description Algorithms for lasso and fused-lasso problems: implementation of
the lars algorithm for lasso and fusion penalization and EM-based
algorithms for (logistic) lasso and fused-lasso penalization.
License GPL (>= 2)
LinkingTo rtkore, Rcpp
SystemRequirements GNU make
RoxygenNote 6.1.1
NeedsCompilation yes
Author Quentin Grimonprez [aut, cre],
Serge Iovleff [aut]
Maintainer Quentin Grimonprez <quentin.grimonprez@inria.fr>
Repository CRAN
Date/Publication 2019-04-10 08:35:18 UTC

R topics documented:

HDPenReg-package .................................................. 2
coeLarsPath .......................................................... 3
coeff ................................................................. 4
computeCoefficients .............................................. 4
EMcvFusedlasso ...................................................... 5
EMcvlasso .......................................................... 6
EMfusedlasso ......................................................... 8
Description

This package contains algorithms for lasso and fused-lasso problems. It contains an implementation of the lars algorithm [1], for the lasso and fusion penalization and EM-based algorithms for (logistic) lasso and fused-lasso.

Details

- **Package:** HDPenReg
- **Type:** Package
- **Version:** 0.94.5
- **Date:** 2019-03-29
- **License:** GPL (>=2)

The main function is `HDLars`.

Author(s)

Maintainer: Quentin Grimonprez <quentin.grimonprez@inria.fr>

See Also

- `HDLars`
- `HDcvlars`

Examples

```r
## Not run:
# see vignette
vignette("HDPenReg")
```
### Description

Compute coefficients at a given level of penalty

### Usage

```r
## S3 method for class 'LarsPath'
coef(object, index = NULL, mode = c("lambda", "step", "fraction", "norm"), ...)
```

### Arguments

- **object**: a LarsPath object
- **index**: If mode = "norm", index represents the l1-norm of the coefficients with which we want to predict. If mode = "fraction", index represents the ratio (l1-norm of the coefficients with which we want to predict) / (l1-norm maximal of the LarsPath object). If mode = "lambda", index represents the value of the penalty parameter. If mode = "step", index represents the number of the step at which we want coefficients.
- **mode**: "fraction" or "norm" or "lambda" or "step".
- **...**: other arguments. Not used

### Value

A vector containing the estimated coefficient for index

### Author(s)

Quentin Grimonprez

### See Also

- `hdlars`
- `LarsPath`

### Examples

```r
dataset <- simul(50, 10000, 0.4, 10, 50, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- hdlars(dataset$data[1:40,], dataset$response[1:40])
coeff <- coef(result, 0.3, "fraction")
```
computeCoefficients

get coefficients at a given step.

Description

Get the vector of coefficients at a given step

Usage

coeff(x, step)

Arguments

x          A LarsPath object.
step       The step at which you want to get the coefficients.

Value

a vector of size p containing the value of coefficients at the desired step.

See Also

HDLars HDFusion LarsPath

Examples

dataset <- simul(50, 1000, 0.4, 10, 50, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- HDFusion(dataset$data, dataset$response)
coefficient <- coeff(result, result@nbStep) # get the coefficients

computeCoefficients

Compute coefficients

Description

Compute coefficients at a given level of penalty

Usage

computeCoefficients(x, lambda, mode = "fraction")
Arguments

- **x**: a LarsPath object
- **lambda**: If mode = "norm", lambda represents the l1-norm of the coefficients with which we want to predict. If mode = "fraction", lambda represents the ratio (l1-norm of the coefficients with which we want to predict)/(l1-norm maximal of the LarsPath object).
- **mode**: "fraction" or "norm" or "lambda".

Value

A list containing

- **variable**: Index of non-zeros coefficients.
- **coefficient**: Non-zeros coefficients.

Author(s)

Quentin Grimonprez

Examples

```r
dataset <- simul(50, 10000, 0.4, 10, 50, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- HDlars(dataset$data[1:40,], dataset$response[1:40])
coeff <- computeCoefficient(result, 0.3, "fraction")
```

Description

cross validation function for EM fusedlasso.

Usage

```r
EMcvfusedlasso(X, y, lambda1, lambda2, nbFolds = 10, maxSteps = 1000,
                burn = 50, intercept = TRUE, model = c("linear", "logistic"),
                eps = 1e-05, eps0 = 1e-08, epsCG = 1e-08)
```

Arguments

- **X**: the matrix (of size n*p) of the covariates.
- **y**: a vector of length n with the response.
- **lambda1**: Values of lambda1 at which prediction error should be computed. Can be a single value.
lambda2 Values of lambda2 at which prediction error should be computed. Can be a single value.
nbFolds the number of folds for the cross-validation.
maxSteps Maximal number of steps for EM algorithm.
burn Number of steps for the burn period.
intercept If TRUE, there is an intercept in the model.
model "linear" or "logistic".
eps Tolerance of the algorithm.
eps0 Zero tolerance. Coefficients under this value are set to zero.
epsCG Epsilon for the convergence of the conjugate gradient.

Value
A list containing

- cv Mean prediction error for each value of index.
- cvError Standard error of cv.
- minCv Minimal cv criterion.
- lambda1 Values of lambda1 at which prediction error should be computed.
- lambda2 Values of lambda2 at which prediction error should be computed.
- lambda_optimal Value of (lambda1,lambda2) for which the cv criterion is minimal.

Author(s)
Quentin Grimonprez, Serge Iovleff

Examples

```r
dataset <- simul(50, 100, 0.4, 1, 10, matrix(c(0.1,0.8,0.02,0.02), nrow = 2))
result <- EMcvfusedlasso(X = dataset$data, y = dataset$response, lambda1 = 3:1,
                         lambda2 = 3:1, nbFolds = 5, intercept = FALSE)
```

**EMcvlasso** cross validation for **EMlasso**

Description
cross validation function for **EMlasso**.

Usage

```r
EMcvlasso(X, y, lambda = NULL, nbFolds = 10, maxSteps = 1000,
          intercept = TRUE, model = c("linear", "logistic"), burn = 30,
          threshold = 1e-08, eps = 1e-05, epsCG = 1e-08)
```
Arguments

- \( x \): the matrix (of size \( n \times p \)) of the covariates.
- \( y \): a vector of length \( n \) with the response.
- \( \text{lambda} \): Values at which prediction error should be computed.
- \( \text{nbFolds} \): the number of folds for the cross-validation.
- \( \text{maxSteps} \): Maximal number of steps for EM algorithm.
- \( \text{intercept} \): If TRUE, there is an intercept in the model.
- \( \text{model} \): "linear" or "logistic".
- \( \text{burn} \): Number of steps for the burn period.
- \( \text{threshold} \): Zero tolerance. Coefficients under this value are set to zero.
- \( \text{eps} \): Tolerance of the EM algorithm.
- \( \text{epsCG} \): Epsilon for the convergence of the conjugate gradient.

Value

A list containing

- \( \text{cv} \): Mean prediction error for each value of index.
- \( \text{cvError} \): Standard error of \( \text{lambda} \).
- \( \text{minCv} \): Minimal \( \text{lambda} \) criterion.
- \( \text{lambda} \): Values of \( \text{lambda} \) at which prediction error should be computed.
- \( \text{lambda.optimal} \): Value of \( \text{lambda} \) for which the \( \text{cv} \) criterion is minimal.

Author(s)

Quentin Grimonprez, Serge Iovleff

Examples

dataset <- simul(50, 100, 0.4, 1, 10, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- EMcvlasso(X = dataset$data, y = dataset$response,
lambda = 5:1, nbFolds = 5, intercept = FALSE)
EMfusedlasso

EM algorithm for fused-lasso penalty

Description

EM algorithm for fused-lasso penalty

Usage

EMfusedlasso(x, y, lambda1, lambda2, maxSteps = 1000, burn = 50,
              intercept = TRUE, model = c("linear", "logistic"), eps = 1e-05,
              eps0 = 1e-08, epsCG = 1e-08)

Arguments

- x: the matrix (of size n*p) of the covariates.
- y: a vector of length n with the response.
- lambda1: a positive real. Parameter associated with the lasso penalty.
- lambda2: a positive real. Parameter associated with the fusion penalty.
- maxSteps: Maximal number of steps for EM algorithm.
- burn: Number of steps before regrouping some variables in segment.
- intercept: If TRUE, there is an intercept in the model.
- model: "linear" or "logistic"
- eps: tolerance for convergence of the EM algorithm.
- eps0: Zero tolerance. Coefficients under this value are set to zero.
- epsCG: tolerance for convergence of the conjugate gradient.

Value

A list containing:

- step: Vector containing the number of steps of the algorithm for every lambda.
- variable: List of vector of size "step+1". The i+1-th item contains the index of non-zero coefficients at the i-th step.
- coefficient: List of vector of size "step+1". The i+1-th item contains the non-zero coefficients at the i-th step.
- lambda: Vector of length "step+1", containing the lambda at each step.
- mu: Intercept.

Author(s)

Quentin Grimonprez, Serge Iovleff
**EMlasso**

**Description**

EM algorithm for lasso penalty

**Usage**

```r
esday = dataset[,1:2]
result = EMlasso(dataset$x, dataset$y, lambda = 0.1, maxsteps = 1000, intercept = TRUE, model = c("linear", "logistic"), burn = 50, threshold = 1e-08, eps = 1e-05, epsCG = 1e-08)
```

**Arguments**

- `x` : the matrix (of size n*p) of the covariates.
- `y` : a vector of length n with the response.
- `lambda` : a sequence of l1 penalty regularization term. If no sequence is provided, the function computes his own sequence.
- `maxSteps` : Maximal number of steps for EM algorithm.
- `intercept` : If TRUE, there is an intercept in the model.
- `model` : "linear" or "logistic"
- `burn` : Number of steps before thresholding some variables to zero.
- `threshold` : Zero tolerance. Coefficients under this value are set to zero.
- `eps` : Epsilon for the convergence of the EM algorithm.
- `epsCG` : Epsilon for the convergence of the conjugate gradient.

**Value**

A list containing:

- `step` : Vector containing the number of steps of the algorithm for every `lambda`.
- `variable` : List of vector of the same length as `lambda`. The i-th item contains the index of non-zero coefficients for the i-th `lambda` value.
- `coefficient` : List of vector of the same length as `lambda`. The i-th item contains the non-zero coefficients for the i-th `lambda` value.
- `lambda` : Vector containing the `lambda` values.
- `mu` : Intercept.
Author(s)
Quentin Grimonprez, Serge Iovleff

See Also
EMcvlasso

Examples

dataset <- simul(50, 100, 0.4, 1, 10, matrix(c(0.1,0.9,0.02,0.02), nrow = 2))
result <- EMlasso(dataset$data, dataset$response)
# Obtain estimated coefficient in matrix format
coefficient <- listToMatrix(result)

HDCvlars  

cross validation

Description

cross validation function for lars algorithm

Usage

HDCvlars(X, y, nbFolds = 10, index = seq(0, 1, by = 0.01),
   mode = c("fraction", "lambda"), maxSteps = 3 * min(dim(X)),
   partition = NULL, intercept = TRUE, eps = .Machine$double.eps^0.5)

Arguments

X  the matrix (of size n*p) of the covariates.
y  a vector of length n with the response.
nbFolds  the number of folds for the cross-validation.
index  Values at which prediction error should be computed. When mode = "fraction",
   this is the fraction of the saturated |beta|. The default value is seq(0,1,by=0.01).
   When mode="lambda", this is values of lambda.
mode  Either "fraction" or "lambda". Type of values containing in partition.
maxSteps  Maximal number of steps for lars algorithm.
partition  partition in nbFolds folds of y. Must be a vector of same size than y containing
   the index of folds.
intercept  If TRUE, there is an intercept in the model.
eps  Tolerance of the algorithm.
**Value**

A list containing

- `cv` Mean prediction error for each value of index.
- `cvError` Standard error of cv.
- `minCv` Minimal cv criterion.
- `minIndex` Value of index for which the cv criterion is minimal.
- `index` Values at which prediction error should be computed. This is the fraction of the saturated lbetal. The default value is seq(0,1,by=0.01).
- `maxSteps` Maximum number of steps of the lars algorithm.

**Author(s)**

Quentin Grimonprez

**Examples**

```r
dataset <- simul(50, 10000, 0.4, 10, 50, matrix(c(0.1,0.8,0.02,0.02), nrow =2 ))
result <- HDcvlar(dataset$data, dataset$response, 5)
```

---

**Description**

It performs the lars algorithm for solving a special case of lasso problem. It is a linear regression problem with a l1-penalty on the difference of two successive coefficients.

**Usage**

```r
HDFusion(X, y, maxSteps = 3 * min(dim(X)), intercept = TRUE, 
eps = .Machine$double.eps^0.5)
```

**Arguments**

- **X** the matrix (of size n*p) of the covariates.
- **y** a vector of length n with the response.
- **maxSteps** Maximal number of steps for lars algorithm.
- **intercept** If TRUE, there is an intercept in the model.
- **eps** Tolerance of the algorithm.

**Value**

An object of type **LarsPath.LarsPath-class.**
Author(s)
 Quentin Grimonprez

References

See Also
 LarsPath HDlars

Examples

```r
set.seed(10)
dataset <- simu(50, 10000, 0.4, 10, 50, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- HDfusion(dataset$data, dataset$response)
```

### HDlars

#### Lars algorithm

**Description**

It performs the lars algorithm for solving lasso problem. It is a linear regression problem with a l1-penalty on the estimated coefficient.

**Usage**

```r
HDlars(X, y, maxSteps = 3 * min(dim(X)), intercept = TRUE,
eps = .Machine$double.eps^0.5)
```

**Arguments**

- `X` the matrix (of size n*p) of the covariates.
- `y` a vector of length n with the response.
- `maxSteps` Maximal number of steps for lars algorithm.
- `intercept` If TRUE, add an intercept to the model.
- `eps` Tolerance of the algorithm.

**Details**

The l1 penalty performs variable selection via shrinkage of the estimated coefficient. It depends on a penalty parameter called lambda controlling the amount of regularization. The objective function of lasso is:

\[ ||y - X\beta||_2 + \lambda||\beta||_1 \]
Value

An object of type LarsPath.

Author(s)

Quentin Grimonprez

References


See Also

LarsPath HDbvlars listToMatrix

Examples

dataset <- simul(50, 10000, 0.4, 10, 50, matrix(c(0.1,0.8,0.02,0.02), nrow = 2))
result <- Hdlars(dataset$data, dataset$response)
# Obtain estimated coefficient in matrix format
coefficient <- listToMatrix(result)

Description

This class stores the results of lars and fusion algorithms.

Details

nbStep  Number of steps of the algorithm.
variable  List of vector of size "step+1". The i+1-th item contains the index of non-zero coefficients at the i-th step.
coefficient  List of vector of size "step+1". The i+1-th item contains the non-zero coefficients at the i-th step.
l1norm  Vector of length "step+1", containing the L1-norm of the coefficients at each step.
lambda  Vector of length "step+1", containing the lambda at each step.
dropIndex  Vector of length "step" containing the index of the dropped variable at the i-th step, 0 means no variable has been dropped at this step.
addIndex  Vector of length "step" containing the index of the added variable at the i-th step, 0 means no variable has been added at this step.
mu  Intercept.
**meanX**  Mean of columns of X.

**ignored**  A vector containing index of ignored variables during the algorithm.

**p**  Total number of covariates.

**fusion**  If TRUE, results from HDfusion function.

**error**  Error message from lars.

**See Also**

HDLars

---

**listToMatrix**  *List to sparse matrix conversion*

**Description**

create a matrix with all estimated coefficients from the output of HDlars or Elasso functions.

**Usage**

```r
listToMatrix(x, row = c("covariates", "lambda"))
```

**Arguments**

- `x`  a LarsPath or Elasso object
- `row`  if covariates, covariates are in row

**Value**

A sparse matrix containing the values of estimated coefficients for all penalty parameter and all covariates

**See Also**

HDLars Elasso
plot-methods

plot methods for LarsPath object

Description

plot the path of the lars algorithm.

Usage

## S4 method for signature 'LarsPath'
plot(x, sep.line = FALSE, abscissa = c("l1norm", "lambda"), log.scale = FALSE, ...)

Arguments

- **x**: LarsPath object
- **sep.line**: If TRUE, print vertical dashed line when a variable is added or dropped in the path
- **abscissa**: either "l1norm" or "lambda". If "lambda", regularization parameter is used as abscissa, else l1 norm of the solution is used.
- **log.scale**: If TRUE, use logarithm scale on abscissa
- **...**: Other plot arguments

See Also

- HDlars
- LarsPath

plot.HDcvlars

plot cross validation mean square error

Description

plot cross validation mean square error

Usage

## S3 method for class 'HDcvlars'
plot(x, ...)

Arguments

- **x**: Output from HDcvlars function.
- **...**: graphical parameters
Author(s)

Quentin Grimonprez

Examples

dataset <- simu(50, 10000, 0.4, 10, 50, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- HDcvalars(dataset$data, dataset$response, 5)
plot(result)

plotCoefficient

Plot of coefficients

Description

Plot of the coefficients of a step

Usage

plotCoefficient(x, step, ylab = "coefficients", xlab = "variables",
                ...

Arguments

x          A LarsPath object.
step       The step at which you want to plot the coefficients.
ylab       Name of the y axis.
xlab       Name of the x axis.
...        Other plot arguments.

See Also

HDlars LarsPath

Examples

dataset <- simu(50, 10000, 0.4, 10, 50, matrix(c(0.1, 0.8, 0.02, 0.02), nrow = 2))
result <- HDfusion(dataset$data, dataset$response)
plotCoefficient(result, result@nbStep) #plot coefficients at the last step
predict.LarsPath

Prediction of response

Description

Predict response of a new sample Xnew at a given level of penalty

Usage

## S3 method for class 'LarsPath'
predict(object, Xnew, lambda, mode = c("fraction", 
  "lambda", "norm"), ...)

Arguments

- **object**: a LarsPath object
- **Xnew**: a matrix (of size n*object@p) of covariates.
- **lambda**: If mode = "norm", lambda represents the l1-norm of the coefficients with which we want to predict. If mode = "fraction", lambda represents the ratio (l1-norm of the coefficients with which we want to predict)/(l1-norm maximal of the LarsPath object).
- **mode**: "fraction", "lambda" or "norm".
- **...**: other arguments. Not used.

Value

The predicted response

Author(s)

Quentin Grimonprez

Examples

dataset <- simul(50, 10000, 0.4, 10, 50, matrix(c(0.1,0.8,0.02,0.02), nrow = 2))
result <- HDlars(dataset$data[1:40,], dataset$response[1:40])
y <- predict(result, dataset$data[41:50,], 0.3, "fraction")
Simulate copy number data for a case-control study.

Usage

```r
simul(n, nbSNP, probCas, nbSeg, meanSegmentSize, prob, alpha = 15)
```

Arguments

- `n`: Number of individuals.
- `nbSNP`: Size of the DNA sequence.
- `probCas`: Probability to be a case individual.
- `nbSeg`: Number of causal segments.
- `meanSegmentSize`: The mean size of anormal segment.
- `prob`: A 2*2 matrix containing probabilities:
  - `prob[1,1]`: probability to have an anomaly to a SNP given the person does not have the disease and the SNP is causal.
  - `prob[1,2]`: probability to have an anomaly to a SNP given the person does not have the disease and the SNP is not causal.
  - `prob[2,1]`: probability to have an anomaly to a SNP given the person has the disease and the SNP is causal.
  - `prob[2,2]`: probability to have an anomaly to a SNP given the person has the disease and the SNP is not causal.
- `alpha`: Parameter of the beta(alpha, alpha).

Value

- A list containing:
  - `data`: A matrix of size n*nbSeg, containing values of the copy-number signal.
  - `response`: A vector of size n containing the cas/control status.
  - `causalSNP`: A vector of size nbSeg containing the center of causal segments.

Author(s)

Quentin Grimonprez, Serge Iovleff

Examples

```r
data <- simul(50, 10000, 0.4, 10, 150, matrix(c(0.1, 0.8, 0.001, 0.001), nrow = 2))
```
Index

*Topic package

HDPenReg-package, 2

coeff.LarsPath, 3
coeff, 4
computeCoefficients, 4

EMcvfusedlasso, 5, 9
EMcvlasso, 6, 10
EMfusedlasso, 5, 8
EMlasso, 6, 9, 14

HDCvlars, 2, 10, 13
HDFusion, 4, 11
HDLars, 2–4, 12, 14–16
HDPenReg (HDPenReg-package), 2
HDPenReg-package, 2
HDPenReg-package, (HDPenReg-package), 2

LarsPath, 3, 4, 11, 13–16
LarsPath (LarsPath-class), 13
LarsPath-class, 13
listToMatrix, 13, 14

plot,LarsPath-method (plot-methods), 15
plot-methods, 15
plot.HDCvlars, 15
plotCoefficient, 16
predict.LarsPath, 17

simul, 18