Package 'lass0'

December 18, 2019

Type Package

Title Lasso-Zero for (High-Dimensional) Linear Regression

Version 1.1.0

Date 2019-12-18

Author Pascaline Descloux,
Sylvain Sardy

Maintainer Pascaline Descloux <pascaline.descloux@unige.ch>

Description Model selection for the (possibly high-dimensional) linear regression model with Lasso-Zero, an L1-based methodology relying on the repeated use of noise dictionaries, as described by Descloux, P. and Sardy, S. (2018) <arXiv:1805.05133>.

Depends stats, graphics

Imports lpSolve (>= 5.6.13), ismev (>= 1.42), foreach (>= 1.4.4),
doRNG (>= 1.7.1)

License GPL-2

URL `http://arxiv.org/abs/1805.05133`

Encoding UTF-8

RoxygenNote 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-18 16:00:05 UTC

R topics documented:

lass0 *Variable selection for linear regression with Lasso-Zero*

Description

Fits a (possibly high-dimensional) linear model with Lasso-Zero. Lasso-Zero aggregates several estimates obtained by solving the basis pursuit problem after concatenating random noise dictionaries to the input matrix. The procedure is described in more details in the paper linked to in the References section below.

Usage

```
lass0(X, y, tau, alpha, q = nrow(X), M = 30, sigma = NULL,
  intercept = TRUE, standardizeX = TRUE, standardizeG = NULL,
  qut.MC.output = NULL, GEVapprox = TRUE, parallel = FALSE,
  soft.thresholding = FALSE, ols = TRUE, ...)
```

Arguments

`X`	input matrix of dimension `n x p`; each row is an observation vector.
`y`	response vector of size `n`.
`tau`	a positive threshold value. If missing, then `alpha` must be supplied.
`alpha`	level of the quantile universal threshold (number between 0 and 1). If missing, then `tau` must be supplied.
`q`	size of noise dictionaries. A noise dictionary consists in a Gaussian matrix G of size `n x q` concatenated horizontally to the input matrix X. Default is `q = nrow(X)`.
`M`	number of noise dictionaries used.
`sigma`	standard deviation of the noise. If `sigma = NULL` (default) and `tau = NULL`, the quantile universal threshold is computed based on a pivotal statistic.
`intercept`	whether an intercept should be fitted. If `TRUE` (default), `y` and the columns of `X` are mean-centered before the analysis, and the intercept is estimated by `mean(y) -colMeans(X) %*% coefficients`.
`standardizeX`	whether the columns of `X` should be standardized to have unit standard deviation. Default is `TRUE`.
`standardizeG`	either a positive numerical value indicating the desired Euclidean norm of all columns of the noise dictionaries, or a logical value indicating whether the columns of the noise dictionaries should be standardized to have unit standard deviation. If `NULL` (default), then it is set to `standardizeG = standardizeX`.
`qut.MC.output`	an object of type `"qut.MC"` (output of `qut.MC` function), providing the result of Monte Carlo simulations necessary for the approximation of the Quantile Universal Threshold. By default, `qut.MC.output = NULL` and the `qut.MC` function is called unless `tau` is supplied.

`GEVapprox`	whether to approximate the distribution of the null thresholding statistic by a GEV distribution (ignored if `tau` is supplied). Default is `TRUE`.
`parallel`	if `TRUE`, use parallel `foreach` to make computations with different noise dictionaries and to perform Monte Carlo simulations for estimating the quantile universal threshold. Must register parallel beforehand, e.g. with `doParallel`. Default is `FALSE`.
`soft.thresholding`	if `TRUE`, the coefficients are soft thresholded (rather than hard thresholded) at level `tau`. Default is `FALSE`.
`ols`	whether to refit the nonzero coefficients with an ordinary least squares procedure. Default is `TRUE`.
`...`	further arguments that can be passed to `qut.MC`.

Value

An object of class `"lass0"`. It is a list containing the following components:

`coefficients`	estimated regression coefficients.
`intercept`	intercept value.
`fitted.values`	fitted values.
`residuals`	residuals.
`selected`	set of selected features.
`tau`	threshold value.
`Betas`	matrix of size `p` x `M` containing the values of the `M` estimates for the regression coefficients (on the standardized scale if `standardizeX` = `TRUE`).
`Gammas`	matrix of size `q` x `M` containing the values of the `M` obtained noise coefficient vectors (on the standardized scale unless `standardizeG` = `FALSE`).
`madGammas`	statistics based on the noise coefficients, corresponding to the MAD of all nonzero entries in `Gammas`
`sdsX`	standard deviations of all columns of `X`. Can be used to transform `Betas` to the original scale doing `Betas` / `sdsX`.
`qut.MC.output`	either the list returned by `qut.MC`, or a character string explaining why `qut.MC` was not called.
`quant.type`	if tau is NULL, indicates the type of quantile used: "GEV" or "empirical" (even when GEVapprox = TRUE, the empirical quantile is used when gev.fit returns an error)
`call`	matched call.

References

Descloux, P., & Sardy, S. (2018). Model selection with lasso-zero: adding straw to the haystack to better find needles. arXiv preprint arXiv:1805.05133. https://arxiv.org/abs/1805.05133

See Also

`qut.MC`

Examples

```
#### EXAMPLE 1: fast example with 5x10 input matrix and a small number
#### (MCrep = 50) of Monte Carlo replications for computing QUT.

set.seed(201)
## design matrix
n <- 5
p <- 10
X <- matrix(rnorm(n*p), n, p)
## sparse vector
S0 <- 1:2 # support
beta0 <- rep(0, p)
beta0[S0] <- 2
## response:
y <- X[, S0] %*% beta0[S0] + rnorm(n)
## lasso-zero:
lass0.obj <- lass0(X, y, alpha = 0.05, MCrep = 50)
betahat <- lass0.obj$coefficients
plot(lass0.obj)


#### EXAMPLE 2: with 50x100 input matrix


set.seed(202)
## design matrix
n <- 50
p <- 100
X <- matrix(rnorm(n*p), n, p)
## sparse vector
S0 <- 1:3 # support
beta0 <- rep(0, p)
beta0[S0] <- 2
## response:
y <- X[, S0] %*% beta0[S0] + rnorm(n)

## 1) lasso-zero tuned by QUT with unknown noise level
lass0.obj1 <- lass0(X, y, alpha = 0.05)
betahat1 <- lass0.obj1$coefficients
plot(lass0.obj1)

## 2) lasso-zero tuned by QUT with known noise level
lass0.obj2 <- lass0(X, y, alpha = 0.05, sigma = 1)
betahat2 <- lass0.obj2$coefficients

## 3) lasso-zero with fixed threshold tau = 1
lass0.obj3 <- lass0(X, y, tau = 1)
betahat3 <- lass0.obj3$coefficients
```

plot.lass0 *Visualizing Lasso-Zero's estimates*

Description

Plots the regression coefficients obtained by Lasso-Zero.

Usage

```
## S3 method for class 'lass0'
plot(x, ...)
```

Arguments

x	a "lass0" object
...	further arguments that can be passed to plot

Details

For a "lass0" object, produces boxplots of the M obtained estimates for each regression coefficient and indicates the threshold level tau. Coefficients whose median is larger than tau is absolute value are the ones selected by Lasso-Zero. Note that if lass0 was called with standardizeX = TRUE, the coefficients and threshold are represented on the standardized scale.

References

Descloux, P., & Sardy, S. (2018). Model selection with lasso-zero: adding straw to the haystack to better find needles. arXiv preprint arXiv:1805.05133. https://arxiv.org/abs/1805.05133

See Also

lass0 and qut.MC

predict.lass0 *Predict method for a Lasso-Zero fit*

Description

Predicted values for the response given a new input matrix Xnew, based on a lass0 fit.

Usage

```
## S3 method for class 'lass0'
predict(object, Xnew, ...)
```

Arguments

`object`	a `"lass0"` object
`Xnew`	a new input matrix whose number of columns equals the number of coefficients returned in `obj`.
`...`	further arguments passed to or from other methods.

Value

vector of predictions

References

Descloux, P., & Sardy, S. (2018). Model selection with lasso-zero: adding straw to the haystack to better find needles. arXiv preprint arXiv:1805.05133. `https://arxiv.org/abs/1805.05133`

See Also

`lass0`

`print.lass0` *Print a lass0 object*

Description

Print a summary of the Lasso-Zero estimate

Usage

```
## S3 method for class 'lass0'
print(x, ...)
```

Arguments

`x`	a `"lass0"` object.
`...`	additional print arguments.

Details

The call that produced the object `x` is printed, followed by the estimated regression coefficients and intercept.

References

Descloux, P., & Sardy, S. (2018). Model selection with lasso-zero: adding straw to the haystack to better find needles. arXiv preprint arXiv:1805.05133. `https://arxiv.org/abs/1805.05133`

See Also

`lass0` and `plot.lass0`

qut.MC — *Monte Carlo simulation for estimating the Quantile Universal Threshold*

Description

Performs a Monte Carlo simulation to estimate the distribution of the null thresholding statistic required for computation of the quantile universal threshold, and computes its upper alpha-quantile if alpha is provided.

Usage

```
qut.MC(X, q = nrow(X), M = 30, alpha = NULL, sigma = NULL,
  intercept = TRUE, standardizeX = TRUE, standardizeG = NULL,
  MCrep = 100, GEVapprox = TRUE, parallel = FALSE,
  var.subset = 1:ncol(X))
```

Arguments

`X`	input matrix of dimension `n x p`, previously mean-centered and standardized if necessary!
`q`	size of noise dictionaries. A noise dictionary consists in a Gaussian matrix G of size `n x q` concatenated horizontally to the input matrix X. Default is `q = nrow(X)`.
`M`	number of noise dictionaries used.
`alpha`	level of the quantile universal threshold. By default `alpha = NULL` and no quantile is returned.
`sigma`	standard deviation of the noise. If `sigma = NULL` (default), the statistic of interest if pivotized.
`intercept`	if `TRUE` (default), the columns of `X` are mean-centered before the analysis.
`standardizeX`	whether the columns of `X` should be standardized to have unit standard deviation. Default is `TRUE`.
`standardizeG`	either a positive numerical value indicating the desired Euclidean norm of all columns of the noise dictionaries, or a logical value indicating whether the columns of the noise dictionaries should be standardized to have unit standard deviation. If `NULL` (default), then it is set to `standardizeG = standardizeX`.
`MCrep`	number of Monte Carlo replications. Default is `MCrep = 100`.
`GEVapprox`	whether to approximate the distribution of the null thresholding statistic by a GEV distribution. If `TRUE`, the maximum likelihood estimates of the GEV parameters are computed on the Monte Carlo sample. Default if `TRUE`.
`parallel`	if `TRUE`, use parallel `foreach` to perform the Monte Carlo simulation. Must register parallel beforehand, e.g. with `doParallel`. Default is `FALSE`.
`var.subset`	subset of variables for which QUT is computed (i.e. we will compute the parameter value for which P(betahat[var.subset]) = 0) = 1-alpha when beta = 0.

Details

If the noise level `sigma` is known, the statistic of interest is simply the sup-norm of the Lasso-Zero coefficients obtained under the null hypothesis (i.e. when all coefficients all zero) when the threshold `tau` is set to 0, and its upper alpha-quantile is the quantile universal threshold. If `sigma = NULL` (sigma unknown) a pivotized statistic is used, which is obtained by dividing the statistic described above by the MAD of all nonzero noise coefficients obtained by Lasso-Zero.

Value

An object of class `"qut.MC"`, which is a list with the following components:

`allMC`	all `MCrep` realizations of the null thresholding statistic of interest (pivotized if `sigma = NULL`).
`GEVpar`	MLE estimates of the GEV distribution parameters (`NULL` if `GEVapprox` was set to `FALSE`).
`GEVfit`	set to NULL is GEVapprox is FALSE. If GEVapprox is TRUE, GEVfit is either the result of the gev.fit function, or the character string "error" if gev.fit produced an error.
`upperQuant`	upper alpha-quantile of the null thresholding statistis (either the empirical quantile, or the quantile of the fitted GEV distribution).
`call`	matched call.
`lass0settings`	a list containing the chosen settings for the computation of lasso-zero: `q`, `M`, `sigma`, `intercept`, `standardizeX` and `standardizeG`. When an object of type `"qut.MC"` is supplied to the `lass0` function, a warning message appears if the corresponding arguments passed to `lass0` are different.

References

Descloux, P., & Sardy, S. (2018). Model selection with lasso-zero: adding straw to the haystack to better find needles. arXiv preprint arXiv:1805.05133. https://arxiv.org/abs/1805.05133

Giacobino, C., Sardy, S., Diaz-Rodriguez, J., & Hengartner, N. (2017). Quantile universal threshold. Electronic Journal of Statistics, 11(2), 4701-4722.

See Also

`lass0`

Examples

```
### Fast toy example with 5x10 input matrix and a small number (MCrep = 50)
### of Monte Carlo replications.
### Illustrates how to tune Lasso-Zero with QUT for the same input matrix but
### different responses and/or different alpha values, without calling
### qut.MC several times:

### (for faster computation when X and MCrep are larger: register a parallel
### backend and choose parallel = TRUE when calling lass0 and qut.MC functions.)
```

```
set.seed(3)

## input matrix:
n <- 5
p <- 10
X <- matrix(rnorm(n*p), n, p)

## two sparse vectors and corresponding responses:
S1 <- 1:2 # first support
beta1 <- numeric(p)
beta1[S1] <- 5
y1 <- X[, S1] %*% beta1[S1] + rnorm(n)
S2 <- 3:4 # second support
beta2 <- numeric(p)
beta2[S2] <- 5
y2 <- X[, S2] %*% beta2[S2] + rnorm(n)


## Monte Carlo simulation giving empirical distribution for the statistic P (see paper below):
qut.MC.output <-  qut.MC(X, parallel = FALSE, MCrep = 50)

## lasso-zero estimates:

## for y1 with alpha = 0.1:
lass01 <- lass0(X, y1, alpha = 0.1, qut.MC.output = qut.MC.output, parallel = FALSE)
plot(lass01)

## for y2 with alpha = 0.05:
lass02 <- lass0(X, y2, alpha = 0.05, qut.MC.output = qut.MC.output, parallel = FALSE)
plot(lass02)
```

Index