Package 'mglasso'

February 21, 2022

Type Package

Title Multiscale Graphical Lasso

Version 0.1.1

Description Inference of Multiscale graphical models with neighborhood selection approach. The method is based on solving a convex optimization problem combining a Lasso and fused-group Lasso penalties. This allows to infer simultaneously a conditional independence graph and a clustering partition. The optimization is based on the Continuation with Nesterov smoothing in a Shrinkage-Thresholding Algorithm solver (Hadj-Selem et al. 2018) <doi:10.1109/TMI.2018.2829802> implemented in python.

License MIT + file LICENSE

Imports corpcor, gridExtra, Matrix, methods, R.utils, reticulate, stats

Suggests knitr, mvtnorm, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

ByteCompile true

Config/reticulate list(packages = list(list(package = c(``"))))

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

NeedsCompilation no

Author Edmond Sanou [aut, cre],
Tung Le [ctb],
Christophe Ambroise [ths],
Geneviève Robin [ths]

Maintainer Edmond Sanou <doedmond.sanou@univ-evry.fr>

Repository CRAN

Date/Publication 2022-02-21 08:20:02 UTC

R topics documented:

`beta_ols` *Initialize regression matrix*

Description

Initialize regression matrix

Usage

```
beta_ols(X)
```

Arguments

`X`	data

Value

A zero-diagonal matrix of regression vectors.

beta_to_vector *Transform a matrix of regression coefficients to vector removing the diagonal*

Description

Transform a matrix of regression coefficients to vector removing the diagonal

Usage

```
beta_to_vector(beta_mat)
```

Arguments

beta_mat	matrix of regressions vectors

Value

A numeric vector of all regression coefficients.

conesta *CONESTA solver.*

Description

Solve the MGLasso optimization problem using CONESTA algorithm. Interface to the pylearn.parsimony python library.

Usage

```
conesta(
  X,
  lam1,
  lam2,
  beta_warm = c(0),
  type_ = "initial",
  W_ = NULL,
  mean_ = FALSE,
  max_iter_ = 10000,
  prec_ = 0.01
)
```

Arguments

X	Data matrix nxp.
lam1	Sparsity penalty.
lam2	Total variation penalty.
beta_warm	Warm initialization vector.
type_	Character scalar. By default set to initial version which don't use weights
W_	Weights matrix.
mean_	Logical scalar. If TRUE weights the optimization function by the inverse of sample size.
max_iter_	Numeric scalar. Maximum number of iterations.
prec_	Numeric scalar. Precision.

Details

COntinuation with NEsterov smoothing in a Shrinkage-Thresholding Algorithm (CONESTA, Hadj-Selem et al. 2018) <doi:10.1109/TMI.2018.2829802> is an algorithm design for solving optimization problems including group-wise penalties. This function is an interface with the python solver. The MGLasso problem is first reformulated in a problem of the form

$$argmin 1/2||Y - \tilde{X}\tilde{\beta}||_2^2 + \lambda_1||\tilde{\beta}||_1 + \lambda_2 \sum_{i<j} ||\boldsymbol{A}_{ij}\tilde{\beta}||_2$$

where vector Y is the vectorized form of matrix X.

Value

Numeric matrix of size pxp. Line k of the matrix represents the coefficients obtained from the L1-L2 penalized regression of variable k on the others.

See Also

mglasso for the MGLasso model estimation.

Examples

```
install_conesta()
n = 30
K = 2
p = 4
rho = 0.85
blocs <- list()
for (j in 1:K) {
  bloc <- matrix(rho, nrow = p/K, ncol = p/K)
  for(i in 1:(p/K)) { bloc[i,i] <- 1 }
  blocs[[j]] <- bloc
}
```

```
mat.covariance <- Matrix::bdiag(blocs)
mat.covariance
set.seed(11)
X <- mvtnorm::rmvnorm(n, mean = rep(0,p), sigma = as.matrix(mat.covariance))
X <- scale(X)
res <- conesta(X, 0.1, 0.1)
```

`cost` *'Mglasso' cost function*

Description

'cost' computes the cost function of 'Mglasso' method.

Usage

```
cost(beta, x, lambda1 = 0, lambda2 = 0)
```

Arguments

`beta`	p by p numeric matrix. In rows, regression vectors coefficients after node-wise regression. `diag(beta) = 0`.
`x`	n by p numeric matrix. Data with variables in columns.
`lambda1`	numeric scalar. Lasso penalization parameter.
`lambda2`	numeric scalar. Fused-group Lasso penalization parameter.

Value

numeric scalar. The cost.

`dist_beta` *Compute distance matrix between regression vectors*

Description

Compute distance matrix between regression vectors

Usage

```
dist_beta(beta, distance = "euclidean")
```

Arguments

`beta`	matrix of regression vectors
`distance`	euclidean or relative distance

Value

A numeric matrix of distances.

`fun_lines` *weighted sum/difference of two regression vectors*

Description

'fun_lines' applies function 'fun' to regression vectors while reordering the coefficients, such that the 'j'-th coefficient in 'beta[j,]' is permuted with the 'i'-th coefficient.

Usage

```
fun_lines(i, j, beta, fun = `-`, ni = 1, nj = 1)
```

Arguments

`i`	integer scalar. Index of the first vector.
`j`	integer scalar. Index of the second vector.
`beta`	p by p numeric matrix. In rows, regression vectors coefficients after node-wise regression. 'diag(beta) = 0'.
`fun`	function. Applied on lines.
`ni`	integer scalar. Weight for vector 'i'.
`nj`	integer scalar. Weight for vector 'j'.

Value

numeric vector

Examples

```
beta <- matrix(round(rnorm(9),2), ncol = 3)
diag(beta) <- 0
beta
fun_lines(1, 2, beta)
fun_lines(2, 1, beta)
```

image_sparse *Plot the image of a matrix*

Description

Plot the image of a matrix

Usage

```
image_sparse(matrix, main_ = "", sub_ = "", col_names = FALSE)
```

Arguments

matrix	matrix of regression coefficients
main_	title
sub_	subtitle
col_names	columns names

Value

No return value.

install_conesta *Install CONESTA solver*

Description

Install CONESTA solver

Usage

```
install_conesta(
  extra_pack = c("scipy == 1.7.1", "scikit-learn", "numpy", "six", "matplotlib"),
  py_version = "3.8.0"
)
```

Arguments

extra_pack	Character vector. Extra-packages to be installed.
py_version	Character. Python version.

Value

No return value.

merge_clusters *compute clusters partition from pairs of variables to merge*

Description

compute clusters partition from pairs of variables to merge

Usage

```
merge_clusters(pairs_to_merge, clusters)
```

Arguments

pairs_to_merge	table of the indices of variables to be merge
clusters	numeric vector. By default 1:p where p is the number of variables

Value

A numeric vector.

mglasso *Inference of Multiscale Gaussian Graphical Model.*

Description

Cluster variables using L2 fusion penalty and simultaneously estimates a gaussian graphical model structure with the addition of L1 sparsity penalty.

Usage

```
mglasso(
  x,
  lambda1 = 0,
  fuse_thresh = 0.001,
  maxit = NULL,
  distance = c("euclidean", "relative"),
  lambda2_start = 1e-04,
  lambda2_factor = 1.5,
  precision = 0.01,
  weights_ = NULL,
  type = c("initial"),
  compact = TRUE
)
```

Arguments

x	Numeric matrix (nxp). Multivariate normal sample with n independent observations.
lambda1	Positive numeric scalar. Lasso penalty.
fuse_thresh	Positive numeric scalar. Threshold for clusters fusion.
maxit	Integer scalar. Maximum number of iterations.
distance	Character. Distance between regression vectors with permutation on symmetric coefficients.
lambda2_start	Numeric scalar. Starting value for fused-group Lasso penalty (clustering penalty).
lambda2_factor	Numeric scalar. Step used to update fused-group Lasso penalty in a multiplicative way..
precision	Precision of estimation algorithm.
weights_	Matrix of weights.
type	If "initial" use classical version of **MGLasso** without weights.
compact	Logical scalar. If TRUE, only save results when previous clusters are different from current.

Details

Estimates a gaussian graphical model structure while hierarchically grouping variables by optimizing a pseudo-likelihood function combining Lasso and fuse-group Lasso penalties. The problem is solved via the *COntinuation with NEsterov smoothing in a Shrinkage-Thresholding Algorithm* (Hadj-Selem et al. 2018). Varying the fusion penalty λ_2 in a multiplicative fashion allow to uncover a seemingly hierarchical clustering structure. For $\lambda_2 = 0$, the approach is theoretically equivalent to the Meinshausen-Bühlmann (2006) *neighborhood selection* as resuming to the optimization of *pseudo-likelihood* function with ℓ_1 penalty (Rocha et al., 2008). The algorithm stops when all the variables have merged into one cluster. The criterion used to merge clusters is the ℓ_2-norm distance between regression vectors.

For each iteration of the algorithm, the following function is optimized :

$$1/2 \sum_{i=1}^{p} ||X^i - X^{\ i}\beta^i||_2^2 + \lambda_1 \sum_{i=1}^{p} ||\beta^i||_1 + \lambda_2 \sum_{i<j} ||\beta^i - \tau_{ij}(\beta^j)||_2.$$

where β^i is the vector of coefficients obtained after regression X^i on the others and τ_{ij} is a permutation exchanging β^i_j with β^j_i.

Value

A list-like object of class mglasso is returned.

out	List of lists. Each element of the list corresponds to a clustering level. An element named levelk contains the regression matrix beta and clusters vector clusters for a clustering in k clusters. When compact = TRUE out has as many elements as the number of unique partitions. When set to FALSE, the function returns as many items as the the range of values taken by lambda2.
l1	the sparsity penalty lambda1 used in the problem solving.

See Also

conesta for the problem solver, plot_mglasso for plotting the output of mglasso.

Examples

```
install_conesta
n = 50
K = 3
p = 9
rho = 0.85
blocs <- list()
for (j in 1:K) {
  bloc <- matrix(rho, nrow = p/K, ncol = p/K)
  for(i in 1:(p/K)) { bloc[i,i] <- 1 }
  blocs[[j]] <- bloc
}

mat.covariance <- Matrix::bdiag(blocs)
mat.covariance

set.seed(11)
X <- mvtnorm::rmvnorm(n, mean = rep(0,p), sigma = as.matrix(mat.covariance))
X <- scale(X)

res <- mglasso(X, 0.1, lambda2_start = 0.1)
res$out[[1]]$clusters
res$out[[1]]$beta
```

---

plot_mglasso *Plot* mglasso *function output.*

Description

Plot the object returned by the mglasso function.

Usage

```
plot_mglasso(mglasso_, levels_ = NULL)
```

Arguments

mglasso_	Object of class mglasso.
levels_	Character vector. Selected levels for which estimated matrices will be plot. If NULL plot all levels.

Value

No return value.

precision_to_regression *Compute precision matrix from regression vectors*

Description

Compute precision matrix from regression vectors

Usage

```
precision_to_regression(K)
```

Arguments

K	precision matrix

Value

A numeric matrix.

symmetrize *Apply symmetrization on estimated graph*

Description

Apply symmetrization on estimated graph

Usage

```
symmetrize(mat, rule = "and")
```

Arguments

mat	graph or precision matrix
rule	"and" or "or" rule

Value

A numeric matrix.

Index