lrpsadmm

March 10, 2019

Type Package

Title Low-rank plus sparse estimation via Alternating Direction Method of Multipliers (ADMM)

Version 0.1.0

Author Benjamin Frot

Maintainer Benjamin Frot <benjamin.frot@gmail.com>

Description Fit the Low-Rank plus Sparse (LRpS) estimator using an accelerated version of the Alternating Direction Method of Multipliers (ADMM) (DOI:10.1137/120896219). This model learns an inverse covariance matrix which is the sum of a sparse matrix and a low-rank matrix as suggested by Chandrasekaran et al (2012) (DOI:10.1214/11-AOS949). The package supports robust estimation via an estimator of the correlation matrix based on Kendall rank correlations. It includes functions to compute a whole regularisation path and to select tuning parameters with cross-validation.

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

Imports MASS,
RSpectra,
mvtnorm,
matrixcalc,
cvTools,
Matrix,
pcaPP,
RColorBrewer

License MIT

R topics documented:

generate.latent.ggm.data 2
Kendall.correlation.estimator 3
lrpsadmm 4
lrpsadmm.cv 7
lrpsadmm.path 10
plot.lrpsadmm 13
plot.lrpsadmmcv 13
plot.lrpsadmmpath 14

Index **16**

`generate.latent.ggm.data` *Simulate data from a Gaussian graphical model with hidden variables*

Description

A partitioned inverse covariance (precision) matrix K is constructed as $K := [K_X, K_{XL}; K_{XL}^T, K_L]$, where K_X is a p x p sparse matrix, K_{LX} is a p x h matrix connecting the h hidden variables to observed ones and K_L is a h x h diagonal matrix.

n samples are then drawn from a multivariate normal distribution: $N(0, K^{-1})$ and only the first p variables are observed.

This function is here to demonstrate the features of the package.

Usage

```
generate.latent.ggm.data(n, p, h, sparsity = 0.02,
  sparsity.latent = 0.7, outlier.fraction = 0)
```

Arguments

`n`	Number of samples.
`p`	Number of observed variables.
`h`	Number of hidden variables.
`sparsity`	Real between 0 and 1. The density of the sparse graph (encoded by K_X).
`sparsity.latent`	Real between 0 and 1. Probability of connection between any observed variable and any hidden variable (K_{LX}).
`outlier.fraction`	Fraction of the samples that should be drawn from a Cauchy distribution. The remaining ones are drawn from a multivariate normal with the same scale matrix.

Value

A list with keys:

obs.data n x p data matrix of observed variables.

full.data (n+h) x p data matrix which also contains the hidden variables.

precision.matrix (n+h)x(n+h) matrix from which the data was sampled. Rows/Columns indexed by 1:p correspond to observed variables. The remaining h rows/cols are the hidden ones.

sparsity Realised sparsity of the precision matrix restricted to observed variables.

Examples

```
n <- 2000 # Number of samples
p <- 100 # Number of variables
h <- 5 # Number of hidden variables
sim.data <- generate.latent.ggm.data(n=n, p=p, h=h, outlier.fraction = 0.0,
                              sparsity = 0.02, sparsity.latent = 0.7)
true.S <- sim.data$precision.matrix[-((p+1):(p+h)),-((p+1):(p+h))] # The sparse matrix
```

```
observed.data <- sim.data$obs.data

# Generate data with 10 of samples drawn from a Cauchy
sim.data <- generate.latent.ggm.data(n=n, p=p, h=h, outlier.fraction = 0.1,
                                sparsity = 0.02, sparsity.latent = 0.7)
true.S <- sim.data$precision.matrix[-((p+1):(p+h)),-((p+1):(p+h))] # The sparse matrix
observed.data <- sim.data$obs.data
```

`Kendall.correlation.estimator` *Estimate the scale matrix of a semiparametric elliptical copula (a.k.a. transelliptical distribution).*

Description

Estimate the correlation matrix of a semiparametric elliptical copula using a modified Kendall rank correlation matrix

Usage

```
Kendall.correlation.estimator(X)
```

Arguments

`X`	An n x p data matrix.

Details

Given X, an n x p data matrix, let K be its Kendall correlation matrix. This function first computes `Chat <- sin(0.5*pi*K)` and projects it onto the space of correlation matrices (in Frobenius norm) using the `nearPD` function of the `Matrix` package

Value

A p x p correlation matrix.

Examples

```
set.seed(0)
# Data from a mixture semiparametric elliptical copula (Cauchy) / multivariate normal
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.05,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data;
Sigma.Kendall <- Kendall.correlation.estimator(X)
```

`lrpsadmm` *Fit the the low-rank plus sparse estimator using an accelerated alternating method direction of multipliers*

Description

Given a n x p data matrix X and its empirical correlation matrix Σ, an alternating direction method of multipliers (ADMM) algorithm is used to obtain the solutions S, L to

$$(S, L) = argmin_{A,B} - logdet(A - B) + Tr((A - B)\Sigma) + \lambda_1 ||A||_1 + \lambda_2 Tr(B),$$

subject to $A - B > 0$ and $B \geq 0$.

Usage

```
lrpsadmm(Sigma, Lambda1, Lambda2, n = NA, init = NULL,
  maxiter = 2000, mu = 0.1, tol = 1e-05, eta = 0.999,
  print_progress = T, print_every = 20, zeros = NULL)
```

Arguments

`Sigma`	An estimate of the correlation matrix.
`Lambda1`	Penalty on the l1 norm of S
`Lambda2`	Penalty on the sum of the eigenvalues of L
`n`	Number of samples. Giving its value is useful when n < p.
`init`	The output of a previous run of the algorithm. For warm starts.
`maxiter`	Maximal number of iterations
`mu`	Stepsize of the ADMM algorithm.
`tol`	Relative tolerance required to stop the algorithm. Algorithm is stopped when either the relative change in log-likelihood is below this threshold, or the relative change in the Frobenius norm of the parameters is below this threshold.
`eta`	Restart parameter in the accelerated ADMM.
`print_progress`	Whether the algorithm should report on its progress.
`print_every`	How often should the algorithm report on its progress (in terms of #iterations).
`zeros`	A p x p matrix with entries set to 0 or 1. Whereever its entries are 0, the entries of the estimated S will be forced to 0.

Details

Given a n x p data matrix X and its empirical correlation matrix Σ, an alternating direction method of multipliers (ADMM) algorithm is used to obtain the solutions S, L to

$$(S, L) = argmin_{A,B} - logdet(A - B) + Tr((A - B)\Sigma) + \lambda_1 ||A||_1 + \lambda_2 Tr(B),$$

subject to $A - B > 0$ and $B \geq 0$. This is the estimator suggested in Chandrasekaran et al.

The optimisation problem is decomposed as a three-block ADMM optimisation problem, as described in Ye et al. Because it is a so-called consensus problem, the ADMM is guaranteed to converge. Given this decomposition of the problem, we use the "Fast ADMM with Restart" (Alg. 8) of Goldstein et al. in order to fit the problem (see references).

The tuning parameters λ_1 and λ_2 are typically reparametrised as $\lambda_1 = \lambda\gamma$ and $\lambda_2 = \lambda(1-\gamma)$, for $\gamma \in (0,1)$. Here, for a fixed γ, λ controls the overall shrinkage along the path defined by γ. γ controls the tradeoff on the penalties between sparse and low-rank components.

For numerical stability, a smaller value of γ is preferable when n is close to p. See examples.

Value

An S3 object of class lrpsadmm. It is essentially a list with keys:

S A p x p matrix. The sparse estimate S

L A p x p matrix. The low-rank estimate L

termcode An integer. Its value determines whether the algorithm terminated normally or with an error. 0: Convergence reached. -1: Maxiter reached. -2: Shrinkage too strong. -3: Too many restarts in a row.

termmsg A character vector. The message corresponding to the value `termcode`.

iter An integer. Number of iterations until convergence.

diffs A vector. Relative difference in change of parameters (in Frobenius norm) at each iteration.

lls A vector. Values taken by the objective function at each iteration.

A A p x p matrix. A variable used by the algorithm. Its value should be close to S - L. It is stored and returned by the algorithm in order to allow warm starts.

U A p x p matrix. Augmented Lagrangian multiplier. It is stored in order to allow warm starts.

References

Chandrasekaran, Venkat; Parrilo, Pablo A.; Willsky, Alan S. Latent variable graphical model selection via convex optimization. Ann. Statist. 40 (2012), no. 4, 1935–1967. doi:10.1214/11-AOS949. https://projecteuclid.org/euclid.aos/1351602527

Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk; Fast Alternating Direction Optimization Methods. SIAM Journal on Imaging Sciences, 2014, Vol. 7, No. 3 : pp. 1588-1623

Gui-Bo Ye, Yuanfeng Wang, Yifei Chen, Xiaohui Xie; Efficient Latent Variable Graphical Model Selection via Split Bregman Method. https://arxiv.org/abs/1110.3076

See Also

lrpsadmm.cv lrpsadmm.path

Examples

```
set.seed(1)
# Generate data for a well-powered dataset
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.0,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data; Sigma <- cor(X) # Sample correlation matrix

### Fit the estimator for some value of the tuning parameters
lambda <- 0.7; gamma <- 0.1 # The tuning parameters.
l1 <- lambda * gamma; l2 <- lambda * (1 - gamma)
fit <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2, n = dim(X)[1])
plot(fit) # Use the S3 method plot
estS <- fit$S # Sparse estimate
```

```
image(estS!=0) # Visualise its non-zero pattern
estL <- fit$L # Low-rank estimate
plot(eigen(estL)$values) # Visualise the spectrum of the low-rank estimate
plot(fit$lls[15:50], type='l') # The log-likelihood from iteration 15 onwards

### Fit for another value of the tuning parameters and compare cold/warm starts
lambda <- 0.4; gamma <- 0.1 # Change the tuning parameters
l1 <- lambda * gamma; l2 <- lambda * (1 - gamma)
# Reuse the previous fit as warm start:
warm.fit <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2, n = dim(X)[1],
init=fit, tol = 1e-09)
warm.fit$iter # Number of itereations of the algorithm
# Fit without warm start
cold.fit <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2, n = dim(X)[1], tol=1e-09)
cold.fit$iter # Number of iterations
plot(cold.fit$lls[10:50], type='l', col='red', ylab = "Log-Likelihood", xlab = "#Iteration")
lines(warm.fit$lls[10:50], col='blue')
legend(20, 103, c("Cold Start", "Warm Start"), col=c("red", "blue"), lty=c(1,1))

### Force the sparsity pattern of the sparse component
zeros = 1 * (sim.data$precision.matrix != 0) # A mtrix of 0 and 1.
zeros = zeros[1:100,1:100] # Keep only the observed part
# Whereever zeros[i,j] = 0, the estimated S will be 0.
fit.zeros <- lrpsadmm(Sigma, l1, l2, n=dim(X)[1], tol=1e-09, zeros = zeros)
fit.no.zeros <- lrpsadmm(Sigma, l1, l2, n=dim(X)[1], tol=1e-09)
image(fit.zeros$S!=0) # Comparing the sparsity patterns
image(fit.no.zeros$S!=0)

### Fit the estimator when the problem is not so well-posed (n close to p)
set.seed(0)
# n = 80, p = 100 with 5 latent variables.
sim.data <- generate.latent.ggm.data(n=80, p=100, h=5, outlier.fraction = 0.0,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data; Sigma <- cor(X) # Sample correlation matrix

lambda <- 2; gamma <- 0.1 # Here gamma is fairly small
l1 <- lambda * gamma; l2 <- lambda * (1 - gamma)
fit.small.gamma <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2, n = dim(X)[1])
plot(eigen(fit.small.gamma$L)$value) # Spectrum of L
lambda <- 0.28 ; gamma <- 0.7 # A large gamma, favourising a non low-rank component.
# This is too high for this ratio n/p
l1 <- lambda * gamma; l2 <- lambda * (1 - gamma)
fit.large.gamma <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2, n = dim(X)[1])
plot(eigen(fit.large.gamma$L)$value) # Spectrum of L
# Numerical stability and convergence are not guaranteed for such an ill-posed proble.
# Gamma is too high.
plot(fit.large.gamma$lls[350:500], type = 'l', xlab= "#Iterations", ylab = "Log-Likelihood")

### Fit the estimator with a robust estimator of the correlation matrix
# Generate data with 5% of outliers
set.seed(0)
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.05,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data;
Sigma <- cor(X) # Sample correlation matrix
Sigma.Kendall <- Kendall.correlation.estimator(X) # The robust estimator
```

```
lambda <- 0.7; gamma <- 0.1 # The tuning parameters.
l1 <- lambda * gamma; l2 <- lambda * (1 - gamma)
fit <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2,
n = dim(X)[1], tol=1e-08, print_every = 200) # Outliers make the problem very ill-posed
Kendall.fit <- lrpsadmm(Sigma = Sigma.Kendall, Lambda1 = l1,
Lambda2 = l2, n = dim(X)[1], tol=1e-08) # Use the Kendall based estimator
image(fit$S!=0)
image(Kendall.fit$S!=0)
plot(fit$lls[800:1200], xlab="#Iterations", ylab="Log-Likelihood")
plot(Kendall.fit$lls, xlab="#Iterations", ylab="Log-Likelihood")
```

`lrpsadmm.cv` *Perform K-fold cross-validation for the Low-Rank plus Sparse estimator*

Description

Performs K-fold cross-validation in order to select the tuning parameters lambda and gamma.

Recall that the penalty for the LRpS estimator is written as $\lambda_1||S||_1 + \lambda_2 Trace(L)$ in the objective function of `lrpsadmm`. This can be equivalently rewritten in terms of the regularisation parameters λ and γ as follows

$$\lambda\gamma||S||_1 + \lambda(1-\gamma)Trace(L),$$

for $\gamma \in (0, 1)$.

For a given value of γ, one can perform cross-validation along the regularisation path in order to choose λ. This function computes the regularisation paths for each value of γ supplied as arguments and performs cross-validation. The pair (λ, γ) that produces the smallest cross-validated log-likelihood is returned.

Usage

```
lrpsadmm.cv(X, gammas = c(0.05, 0.1, 0.15), covariance.estimator = cor,
  n.folds = 5, lambdas = NULL, lambda.max = NULL,
  lambda.ratio = 1e-04, n.lambdas = 20, max.sparsity = 0.5,
  max.rank = NA, tol = 1e-05, max.iter = 2000, mu = 0.1,
  verbose = FALSE, seed = NA, zeros = NULL)
```

Arguments

`X`	n x p data matrix
`gammas`	A real or a vector of reals between 0 and 1 (non inclusive). For each value of gamma the regularisation path is computed and cross-validation is performed. See examples for guidance on how to choose reasonable values for gamma. Too high a value might result in the problem being under-identified and therefore numerical instabilities.
`covariance.estimator`	A function that takes a data matrix and outputs an estimate of its correlation matrix. Default: the sample correlation matrix output by the `cor` function.
`n.folds`	Number of folds for cross-validation. Default 5.
`lambdas`	A decreasing sequence of values of lambda. See Details for the default values.

`lambda.max`	A positive real. Maximum value of lambda. See Details.
`lambda.ratio`	A real between 0 and 1. The smallest value of lambda is given by lambda.max * lambda.ratio. See Details.
`n.lambdas`	A positive integer. The number of values of lambda to generate according a geometric sequence between lambda.max and lambda.max * lambda.ratio. See Details.
`max.sparsity`	A real between 0 and 1. Abort the computation of the path if S becomes denser than this value.
`max.rank`	A real between 0 and 1. Abort the computuation of the path if the rank of L becomes higher than this value.
`tol`	`tol` parameter of the `lrpsadmm` function.
`max.iter`	`max.iter` parameter of the `lrpsadmm` function.
`mu`	`mu` parameter of the `lrpsadmm` function.
`verbose`	A boolean. Whether to print the value of lambda, gamma, sparsity of S, etc... after each fit
`seed`	Set the seed of the random number generator used for the K folds.
`zeros`	A p x p matrix with entries set to 0 or 1. Whereever its entries are 0, the entries of the estimated S will be forced to 0.

Details

Recall that the penalty for the LRpS estimator is written as $\lambda_1||S||_1 + \lambda_2 Trace(L)$ in the objective function of `lrpsadmm`. This can be equivalently rewritten in terms of the regularisation parameters λ and γ as follows

$$\lambda\gamma||S||_1 + \lambda(1-\gamma)Trace(L),$$

for $\gamma \in (0, 1)$.

For a given value of γ, one can perform cross-validation along the regularisation path in order to choose λ. This function computes the regularisation paths for each value of γ supplied as arguments and performs cross-validation. One can then select the pair (λ, γ) that produces the smallest cross-validated log-likelihood.

The function `lrpsadmm` is fitted for successive values of λ using warm starts. The sequence of values of λ can be provided directly by the user. It is automatically sorted in decreasing order. By default, a decreasing sequence of 20 values within a reasonable range is selected as follows. We set $\lambda_{max} = \max_{ij, i \neq j} |\Sigma_{ij}|/\gamma$ and $\lambda_{min} = \lambda_{max}$ * `lambda.ratio`; then 20 values between λ_{max} and λ_{min} are taken following a geometric progression.

Because it does not make much sense to fit this estimator when the sparse estimate S becomes too dense or if the rank of the low-rank estimate L becomes too high, the computation of the path is aborted when the sparsity of S reaches `max.sparsity` or when the rank of L reaches `max.rank`.

Recall that cross-validation overselects. It might be good for prediction purposes but methods such statibility selection are probably better if the support of S is what is of interest.

Value

An object of class lrpsadmmcv. It contains the values of the mean cross-validated log-likelihood, its standard deviation for each pair (lambda, gamma) and an object of class lrpsadmm.path for each value of gamma. See the examples for how to access the selected tuning paramters, best fit etc...

See Also

lrpsadmm lrpadmm.path

Examples

```
set.seed(0)
## 1 - A simple example: Well-powered dataset
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.0,
                                     sparsity = 0.02, sparsity.latent = 0.7)
ground.truth <- sim.data$precision.matrix[1:100, 1:100]
ground.truth <- 1 * (( ground.truth - diag(diag(ground.truth)) ) !=0)
X <- sim.data$obs.data;
gammas <- c(0.1, 0.15, 0.2)
cvpath <- lrpsadmm.cv(X, gammas = gammas,
                          lambda.ratio = 1e-02, n.lambdas = 30, verbose = TRUE)
best.gamma <- cvpath$best.gamma
best.lambda <- cvpath$best.lambda
best.fit <- cvpath$best.fit$fit # Object of class lrpsadmm
plot(best.fit)
# The value Gamma = 0.15 is selected by X-validation
plot(cvpath) # Plot the outcome. Object of class lrpsadmmcv
# We can look at the path corresponding to this value
best.path <- cvpath$cross.validated.paths[[which(gammas == best.gamma)]]$cross.validated.path
# We know the ground truth, so let's use it to see how well we did:
plot(best.path, ground.truth = ground.truth)

## 2 - Data with outliers: use a robust estimator
set.seed(0)
sim.data <- generate.latent.ggm.data(n=2000, p=50, h=5, outlier.fraction = 0.05,
                                     sparsity = 0.02, sparsity.latent = 0.7)
ground.truth <- sim.data$precision.matrix[1:50, 1:50]
# Remove the elements along the diagonal. Keep a matrix of 0s and 1s
ground.truth <- 1 * (( ground.truth - diag(diag(ground.truth)) ) !=0)
X <- sim.data$obs.data;
# We can afford high values of gamma because n >> p
gammas <- c(0.2, 0.3, 0.4)
# Use the Kendall based estimator:
cvpath <- lrpsadmm.cv(X, gammas = gammas, covariance.estimator = Kendall.correlation.estimator,
                          lambda.ratio = 1e-03, n.lambdas = 30, verbose = TRUE)
plot(cvpath)
best.gamma <- cvpath$best.gamma
best.lambda <- cvpath$best.lambda
best.path <- cvpath$cross.validated.paths[[which(gammas == best.gamma)]]$cross.validated.path
plot(best.path, ground.truth = ground.truth)

## 3 - A tougher problem n is close to p
set.seed(0)
sim.data <- generate.latent.ggm.data(n=150, p=100, h=5, outlier.fraction = 0.0,
                                      sparsity = 0.02, sparsity.latent = 0.7)
ground.truth <- sim.data$precision.matrix[1:100, 1:100]
# Remove the elements along the diagonal. Keep a matrix of 0s and 1s
ground.truth <- 1 * (( ground.truth - diag(diag(ground.truth)) ) !=0)
X <- sim.data$obs.data;
# Since n < p, do not try too high values of gamma. Stay close to 0.
gammas <- c(0.07, 0.1, 0.12)
cvpath <- lrpsadmm.cv(X, gammas = gammas,
```

```
                          lambda.ratio = 0.1, n.lambdas = 20, verbose = TRUE)
plot(cvpath) # Plot the outcome

# Clearly the range selected by the function is not good enough
# We need better values for lambda. In that case it is better
# to see which value of lambda yields a very sparse graph
# for a given gamma:
gammas <- c(0.1)
# We set the seed so we can compre the x-validated log-likelihoods between
# two runs of the function
cvpath <- lrpsadmm.cv(X, gammas = gammas, lambda.max = 2.2,
                       lambda.ratio = 0.1, n.lambdas = 20, verbose = TRUE, seed = 0)
plot(cvpath)
gammas <- c(0.12)
cvpath <- lrpsadmm.cv(X, gammas = gammas, lambda.max = 1.7,
                        lambda.ratio = 0.1, n.lambdas = 20, verbose = TRUE, seed = 0)
plot(cvpath)
```

`lrpsadmm.path` *Compute the LRpS estimator along a path (for a fixed value of $gamma$)*

Description

The penalty for the LRpS estimator is written as $\lambda_1||S||_1 + \lambda_2 Trace(L)$ in the objective function of `lrpsadmm`. This can be equivalently rewritten in terms of the regularisation parameters λ and γ as follows

$$\lambda\gamma||S||_1 + \lambda(1-\gamma)Trace(L),$$

for $\gamma \in (0, 1)$. This function estimates the path of the estimator for a fixed value of γ (which controls the trade-off between the two penalties) by varying the value of λ. See the documentation of `lrpsadmm` and references therein for more details.

Usage

```
lrpsadmm.path(Sigma, gamma, n = NA, lambdas = NULL,
  lambda.max = NULL, lambda.ratio = 1e-04, n.lambdas = 20,
  max.sparsity = 0.5, max.rank = NA, tol = 1e-05, max.iter = 2000,
  mu = 0.1, zeros = NULL, verbose = FALSE)
```

Arguments

`Sigma`	A p x p matrix. An estimate of the correlation matrix
`gamma`	A real between 0 and 1. The value of the tuning parameter gamma in the parametrisation of the penalty described avove. This is the trade-off between the sparse and trace penalties.
`n`	An integer. Number of samples from which Sigma has been computed. Useful when n < p.
`lambdas`	A decreasing sequence of values of lambda. See Details for the default value.
`lambda.max`	A positive real. Maximum value of lambda. See Details.

`lambda.ratio`	A real between 0 and 1. The smallest value of lambda is given by lambda.max * lambda.ratio. See Details.
`n.lambdas`	A positive integer. The number of values of lambda to generate according a geometric sequence between lambda.max and lambda.max * lambda.ratio. See Details.
`max.sparsity`	A real between 0 and 1. Abort the computation of the path if S becomes denser than this value.
`max.rank`	A real between 0 and 1. Abort the computuation of the path if the rank of L becomes higher than this value.
`tol`	`tol` parameter of the `lrpsadmm` function.
`max.iter`	`max.iter` parameter of the `lrpsadmm` function.
`mu`	`mu` parameter of the `lrpsadmm` function.
`zeros`	A p x p matrix with entries set to 0 or 1. Whereever its entries are 0, the entries of the estimated S will be forced to 0.
`verbose`	A boolean. Whether to print the value of lambda, gamma, sparsity of S and rank of L after each fit.

Details

The function `lrpsadmm` is fitted for successive values of λ using warm starts. The sequence of values of λ can be provided directly by the user. It is automatically sorted in decreasing order. By default, a decreasing sequence of 20 values within a reasonable range is selected as follows. We set $\lambda_{max} = \max_{ij, i \neq j} |\Sigma_{ij}|/\gamma$ and $\lambda_{min} = \lambda_{max}$ * `lambda.ratio`; then 20 values between λ_{max} and λ_{min} are taken following a geometric progression.

Because it does not make much sense to fit this estimator when the sparse estimate S becomes too dense or if the rank of the low-rank estimate L becomes too high, the computation of the path is aborted when the sparsity of S reaches `max.sparsity` or when the rank of L reaches `max.rank`.

Value

An object of class lrpsaddmpath. This is essentially a list (see examples). Each element is itself a list with keys:

lambda Value of lambda used for that fit. Recall that the value of the tuning parameters λ_1, λ_2 is given by $\lambda_1 = lambda * gamma$ and $\lambda_2 = lambda * (1 - gamma)$

gamma Value of gamma used for that fit.

lambda1 Corresponds to the parameter `l1` given as argument to the `lrpsadmm` function. lambda1 = lambda * gamma

lambda2 Corresponds to the parameter `l2` given as argument to the `lrpsadmm` function. lambda2 = lambda * (1 - gamma)

number.of.edges Number of edges in the estimated sparse graphical model.

rank.L Rank of the estimated low-rank matrix L.

sparsity Sparsity of the estimated sparse matrix. This is fraction of entries that are non-zero.

fit An object of class lrpsadmm. This is the outcome of calling `lrpsadmm` with tuning parameters l1 = lambda * gamma and l2 = lambda (1 - gamma). See the documentation of the function `lrpsadmm` for more information.

Examples

```
set.seed(0)
# Generate data with a well-powered dataset
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.0,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data; Sigma <- cor(X) # Sample correlation matrix


gamma <- 0.1 # Some reasonble value for gamma
# We ask for 30 lambdas, but the sparse graph becomes too dense so the
# computation is stopped.
my.path <- lrpsadmm.path(Sigma = Sigma, gamma = gamma,
                         lambda.ratio = 1e-03, n.lambdas = 30, verbose = TRUE)

# This time let us ask for 30 values,
# but let us narrow down the range by using a
# a smaller ratio
my.path <- lrpsadmm.path(Sigma = Sigma, gamma = gamma,
                    lambda.max = 0.96, lambda.ratio = 0.1, n.lambdas = 30, verbose = TRUE)

# Plot some basic information about the path
plot(my.path)
# Look at the first graph in the path
plot(my.path[[1]]$fit)
# Because this is simulated data, we know the ground truth
# Let us use it to compute the precsion and recall metrics
# along the path
ground.truth <- sim.data$precision.matrix[1:100, 1:100]
# Remove the elements along the diagonal. Keep a matrix of 0s and 1s
ground.truth <- 1 * (( ground.truth - diag(diag(ground.truth)) ) !=0)
# There is a new plot with the precision / recall curve
plot(my.path, ground.truth = ground.truth)

### Let us use a robust estimator of the correlation matrix
# Generate data with 5% of outliers
set.seed(0)
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.05,
                                     sparsity = 0.02, sparsity.latent = 0.7)
ground.truth <- sim.data$precision.matrix[1:100, 1:100]
# Remove the elements along the diagonal. Keep a matrix of 0s and 1s
ground.truth <- 1 * (( ground.truth - diag(diag(ground.truth)) ) !=0)
X <- sim.data$obs.data;
Sigma <- cor(X) # Sample correlation matrix
Sigma.Kendall <- Kendall.correlation.estimator(X) # The robust estimator

# With that many strong outliers, using the sample corr. mat.
# is not well-suited to the problem
gamma <- 0.2
my.path <- lrpsadmm.path(Sigma = Sigma, gamma = gamma,
                         lambda.ratio = 1e-02, n.lambdas = 30, verbose = TRUE)
# Use another estimator for the correlation matrix:
my.robust.path <- lrpsadmm.path(Sigma = Sigma.Kendall, gamma = gamma,
                         lambda.ratio = 1e-01, n.lambdas = 30, verbose = TRUE)
# The output of the sample correlation path is poor (in terms of prec/recall)
# This is pretty much noise
plot(my.path, ground.truth)
```

```
# The Kendall estimator produces far better results.
# It is not affected by the 5% of outliers
plot(my.robust.path, ground.truth)
```

`plot.lrpsadmm` *Plotting function for 'lrpsadmm' Objects*

Description

Plots the sparsity pattern of S, the eigenvalues of L and the values taken by the objective function at each iteration.

Usage

```
## S3 method for class 'lrpsadmm'
plot(x)
```

Arguments

`x`	An object of class lrpsadmm output by the function `lrpsadmm`

Examples

```
set.seed(1)
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.0,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data; Sigma <- cor(X) # Sample correlation matrix
lambda <- 0.7; gamma <- 0.1 # The tuning parameters.
l1 <- lambda * gamma; l2 <- lambda * (1 - gamma)
fit <- lrpsadmm(Sigma = Sigma, Lambda1 = l1, Lambda2 = l2, n = dim(X)[1])
plot(fit)
```

`plot.lrpsadmmcv` *Plotting for Objects of class "lrpsadmmcv"*

Description

Plots an object of class lrpsadmmcv (output by the `lrpsadmm.cv` function). The function plots the support and spectrum of the fit that achieves the best cross-validated negative log-likelihood, along with the cross-validated log-likelihood (and its standard deviation) for each value of gamma

Usage

```
## S3 method for class 'lrpsadmmcv'
plot(x)
```

Arguments

`x`	An object of class lrpsadmmcv output by the `lrpsadmm.cv` function.

See Also

plot.lrpsadmm plot.lrpsadmmpath

plot.lrpsadmmpath *Plotting for 'lrpsadmmpath' Objects*

Description

Plots the sparsity and number of edges of S as a function of the tuning parameter lambda (see documentation of lrpsadmmpath). The rank of L as a function of lambda.

The user can also provide a matrix of 0s and 1s to be used as "ground truth". If the location of non-zero entries of S is known (for example because this is simulated data), then the precision/recall metrics are computed and plotted. See examples below.

Usage

```
## S3 method for class 'lrpsadmmpath'
plot(x, ground.truth = NULL)
```

Arguments

x	An object of class lrpsadmmpath output by the function `lrpsadmmpath`
ground.truth	A binary matrix representing the adjacency matrix of the "true" graph one seeks to recover. Useful mostly on simulated data where the true parameter is known.

See Also

lrpsadmm lrpsadmm.cv

Examples

```
set.seed(0)
# Generate data with a well-powered dataset
sim.data <- generate.latent.ggm.data(n=2000, p=100, h=5, outlier.fraction = 0.0,
                                     sparsity = 0.02, sparsity.latent = 0.7)
X <- sim.data$obs.data; Sigma <- cor(X) # Sample correlation matrix


gamma <- 0.1 # Some reasonble value for gamma
# We ask for 30 lambdas, but the sparse graph becomes too dense so the
# computation is stopped.
my.path <- lrpsadmm.path(Sigma = Sigma, gamma = gamma,
                         lambda.ratio = 1e-03, n.lambdas = 30, verbose = TRUE)

# This time let us ask for 30 values, but let us narrow down the range by using a
# a smaller ratio
my.path <- lrpsadmm.path(Sigma = Sigma, gamma = gamma,
                    lambda.max = 0.96, lambda.ratio = 0.1, n.lambdas = 30, verbose = TRUE)

# Plot some basic information about the path
plot(my.path)
# Look at the first graph in the path
plot(my.path[[1]]$fit)
# Because this is simulated data, we know the ground truth
# Let us use it to compute the precsion and recall metrics
# along the path
```

```
ground.truth <- sim.data$precision.matrix[1:100, 1:100]
# Remove the elements along the diagonal. Keep a matrix of 0s and 1s
ground.truth <- 1 * (( ground.truth - diag(diag(ground.truth)) ) !=0)
# There is a new plot with the precision / recall curve
plot(my.path, ground.truth = ground.truth)
```

Index

generate.latent.ggm.data, 2

Kendall.correlation.estimator, 3

lrpsadmm, 4
lrpsadmm.cv, 7
lrpsadmm.path, 10

plot.lrpsadmm, 13
plot.lrpsadmmcv, 13
plot.lrpsadmmpath, 14