Package: spcov (via r-universe)

September 17, 2024

Type Package

Title Sparse Estimation of a Covariance Matrix

Version 1.3

Date 2022-09-21

Author Jacob Bien and Rob Tibshirani

Maintainer Jacob Bien <jbien@usc.edu>

Description Provides a covariance estimator for multivariate normal data that is sparse and positive definite. Implements the majorize-minimize algorithm described in Bien, J., and Tibshirani, R. (2011), ``Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807--820.

License GPL-2

LazyLoad yes

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Date/Publication 2022-09-23 20:20:02 UTC

Repository https://jacobbien.r-universe.dev

RemoteUrl https://github.com/cran/spcov

RemoteRef HEAD

RemoteSha a2431960bf48496fe0f1e8bebb503204c96583cd

Contents

spcov-package *Sparse Estimation of a Covariance Matrix*

Description

Performs the majorize-minimize algorithm described in

Details

Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807–820.

Generalized gradient descent is used to minimize each majorizer.

Package:	spcov
Type:	Package
Version:	1.01
Date:	2012-03-04
License:	GPL-2
LazyLoad:	yes

See the function spcov.

Author(s)

Jacob Bien and Rob Tibshirani

Maintainer: Jacob Bien <jbien@stanford.edu>

References

Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807–820.

See Also

spcov

GenerateCliquesCovariance
Generate a block diagonal covariance matrix

Description

This function is included in the package so that it can be used in the example code provided in spcov.

Usage

```
GenerateCliquesCovariance(ncliques, cliquesize, theta)
```

Arguments

ncliques	number of blocks
cliquesize	size of each block
theta	magnitude of non-zeros

Details

This function generates a block diagonal positive definite matrix with randomly-signed, non-zero elements. A shift is added to the diagonal of the matrix so that its condition number equals p, the number of variables.

Value

Sigma	the covariance matrix
A	symmetric square root of Sigma
shift	how much the eigenvalues were shifted. See details.

Author(s)

Jacob Bien and Rob Tibshirani

References

Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," accepted for publication in Biometrika.

`ProxADMM` *Solving penalized Frobenius problem.*

Description

This function solves the optimization problem

Usage

```
ProxADMM(A, del, lam, P, rho = 0.1, tol = 1e-06, maxiters = 100, verb = FALSE)
```

Arguments

`A`	A symmetric matrix.
`del`	A non-negative scalar. Lower bound on eigenvalues.
`lam`	A non-negative scalar. L1 penalty parameter.
`P`	Matrix with non-negative elements and dimension of A. Allows for differing L1 penalty parameters.
`rho`	ADMM parameter. Can affect rate of convergence a lot.
`tol`	Convergence threshold.
`maxiters`	Maximum number of iterations.
`verb`	Controls whether to be verbose.

Details

Minimize_X (1/2)||X - A||_F^2 + lam||P*X||_1 s.t. X >= del * I.

This is the prox function for the generalized gradient descent of Bien & Tibshirani 2011 (see full reference below).

This is the R implementation of the algorithm in Appendix 3 of Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807–820. It uses an ADMM approach to solve the problem

Minimize_X (1/2)||X - A||_F^2 + lam||P*X||_1 s.t. X >= del * I.

Here, the multiplication between P and X is elementwise. The inequality in the constraint is a lower bound on the minimum eigenvalue of the matrix X.

Note that there are two variables X and Z that are outputted. Both are estimates of the optimal X. However, Z has exact zeros whereas X has eigenvalues at least del. Running the ADMM algorithm long enough, these two are guaranteed to converge.

Value

`X`	Estimate of optimal X.
`Z`	Estimate of optimal X.
`obj`	Objective values.

Author(s)

Jacob Bien and Rob Tibshirani

References

Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807–820.

See Also

spcov

Examples

```
set.seed(1)
n <- 100
p <- 200
# generate a covariance matrix:
model <- GenerateCliquesCovariance(ncliques=4, cliquesize=p / 4, 1)

# generate data matrix with x[i, ] ~ N(0, model$Sigma):
x <- matrix(rnorm(n * p), ncol=p) %*% model$A
S <- var(x)

# compute sparse, positive covariance estimator:
P <- matrix(1, p, p)
diag(P) <- 0
lam <- 0.1
aa <- ProxADMM(S, 0.01, lam, P)
```

---

spcov *Sparse Covariance Estimation*

Description

Provides a sparse and positive definite estimate of a covariance matrix. This function performs the majorize-minimize algorithm described in Bien & Tibshirani 2011 (see full reference below).

Usage

```
spcov(
  Sigma,
  S,
  lambda,
  step.size,
  nesterov = TRUE,
  n.outer.steps = 10000,
```

```
  n.inner.steps = 10000,
  tol.outer = 1e-04,
  thr.inner = 0.01,
  backtracking = 0.2,
  trace = 0
)
```

Arguments

`Sigma`	an initial guess for Sigma (suggestions: `S` or `diag(diag(S))`).
`S`	the empirical covariance matrix of the data. Must be positive definite (if it is not, add a small constant to the diagonal).
`lambda`	penalty parameter. Either a scalar or a matrix of the same dimension as `Sigma`. This latter choice should be used to penalize only off-diagonal elements. All elements of `lambda` must be non-negative.
`step.size`	the step size to use in generalized gradient descent. Affects speed of algorithm.
`nesterov`	indicates whether to use Nesterov's modification of generalized gradient descent. Default: `TRUE`.
`n.outer.steps`	maximum number of majorize-minimize steps to take (recall that MM is the outer loop).
`n.inner.steps`	maximum number of generalized gradient steps to take (recall that generalized gradient descent is the inner loop).
`tol.outer`	convergence threshold for outer (MM) loop. Stops when drop in objective between steps is less than `tol.outer`.
`thr.inner`	convergence threshold for inner (i.e. generalized gradient) loop. Stops when mean absolute change in `Sigma` is less than `thr.inner * mean(abs(S))`.
`backtracking`	if `FALSE`, then fixed step size used. If numeric and in (0,1), this is the parameter of backtracking that multiplies `step.size` on each step. Usually, in range of (0.1, 0.8). Default: `0.2`.
`trace`	controls how verbose output should be.

Details

This is the R implementation of Algorithm 1 in Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807–820. The goal is to approximately minimize (over Sigma) the following non-convex optimization problem:

minimize logdet(Sigma) + trace(S Sigma^-1) + || lambda*Sigma ||_1 subject to Sigma positive definite.

Here, the L1 norm and matrix multiplication between lambda and Sigma are elementwise. The empirical covariance matrix must be positive definite for the optimization problem to have bounded objective (see Section 3.3 of paper). We suggest adding a small constant to the diagonal of S if it is not. Since the above problem is not convex, the returned matrix is not guaranteed to be a global minimum of the problem.

In Section 3.2 of the paper, we mention a simple modification of gradient descent due to Nesterov. The argument `nesterov` controls whether to use this modification (we suggest that it be used). We

also strongly recommend using backtracking. This allows the algorithm to begin by taking large steps (the initial size is determined by the argument `step.size`) and then to gradually reduce the size of steps.

At the start of the algorithm, a lower bound (`delta` in the paper) on the eigenvalues of the solution is calculated. As shown in Equation (3) of the paper, the prox function for our generalized gradient descent amounts to minimizing (over a matrix X) a problem of the form

minimize (1/2)|| X-A ||_F^2 + || lambda*X ||_1 subject to X >= delta I

This is implemented using an alternating direction method of multipliers approach given in Appendix 3.

Value

`Sigma`	the sparse covariance estimate
`n.iter`	a vector giving the number of generalized gradient steps taken on each step of the MM algorithm
`obj`	a vector giving the objective values after each step of the MM algorithm

Author(s)

Jacob Bien and Rob Tibshirani

References

Bien, J., and Tibshirani, R. (2011), "Sparse Estimation of a Covariance Matrix," Biometrika. 98(4). 807–820.

See Also

ProxADMM

Examples

```
set.seed(1)
n <- 100
p <- 20
# generate a covariance matrix:
model <- GenerateCliquesCovariance(ncliques=4, cliquesize=p / 4, 1)

# generate data matrix with x[i, ] ~ N(0, model$Sigma):
x <- matrix(rnorm(n * p), ncol=p) %*% model$A
S <- var(x)

# compute sparse, positive covariance estimator:
step.size <- 100
tol <- 1e-3
P <- matrix(1, p, p)
diag(P) <- 0
lam <- 0.06
mm <- spcov(Sigma=S, S=S, lambda=lam * P,
            step.size=step.size, n.inner.steps=200,
```

```
            thr.inner=0, tol.outer=tol, trace=1)
sqrt(mean((mm$Sigma - model$Sigma)^2))
sqrt(mean((S - model$Sigma)^2))
## Not run: image(mm$Sigma!=0)
```

Index