Package ‘hierband’

June 13, 2015

Title Convex Banding of the Covariance Matrix

Description Implementation of the convex banding procedure (using a hierarchical group lasso penalty) for covariance estimation that is introduced in Bien, Bunea, Xiao (2015) Convex Banding of the Covariance Matrix. Accepted for publication in JASA.

Version 1.0

Author Jacob Bien <jbien@cornell.edu>, Florentina Bunea, and Luo Xiao

Depends R (>= 3.2.0)

Suggests knitr

License GPL-3

LazyData true

Maintainer Jacob Bien <jbien@cornell.edu>

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-06-13 00:52:38

R topics documented:

`hierband-package` *Convex banding of the covariance matrix using*

Description

`hierband` is the R package implementing the convex banding approach to covariance estimation of Bien, Bunea, & Xiao (see full reference below).

Details

The package is designed for situations in which there is a large number of variables that have a known ordering and in which it is believed that variables far apart in this ordering have little to no dependence.

It is called `hierband` (pronounced "hair band") because it makes use of a hierarchical group lasso penalty and provides a banded estimate of the covariance matrix.

Its main functions are `hierband`, `hierband.path`, `hierband.cv`.

Author(s)

Jacob Bien <`jbien@cornell.edu`>, Florentina Bunea, Luo Xiao

References

Bien, J., Bunea, F. Xiao, L. (2014) "Convex banding of the covariance matrix." Accepted for publication in JASA.

`banded` *Generates a banded covariance matrix and matrix squareroot sig: value of kth band (starting with diagonal) size of band is length(sig)*

Description

Generates a banded covariance matrix and matrix squareroot sig: value of kth band (starting with diagonal) size of band is length(sig)

Usage

```
banded(p, sig)
```

Arguments

`p`	dimension of covariance matrix
`sig`	vector of values of Toeplitz matrix

formw — *Form the "general weights" matrix*

Description

Form the "general weights" matrix

Usage

```
formw(p)
```

Arguments

`p`	dimension of covariance matrix

gpband — *Groupwise soft-thresholds subdiagonals by lam * w*

Description

Groupwise soft-thresholds subdiagonals by lam * w

Usage

```
gpband(R, lam, w = NULL)
```

Arguments

`R`	p-by-p symmetric matrix
`lam`	Non-negative penalty parameter. Controls sparsity level.
`w`	`(p-1)` vector of weights. Default, w[l]=sqrt(2 * l)

hierband *Solves main optimization problem for fixed lambda value*

Description

Solves the main optimization problem appearing in Bien, Bunea, & Xiao (2015):

$$min_P ||Sighat - P||_F^2 + lam * sum_l ||(W_l * P)_{g_l}||_2$$

where g_l are the outermost l(l+1) elements of a square matrix. and || (W_l * P)_g_l ||^2 = sum_m<=l w_lm^2 ||P_s_m||^2. If a non-NULL `delta` is provided, then a constraint of the form $P >= delta I_p$ is included. Problem is solved by performing blockwise coordinate descent on the dual problem. See paper for more explanation.

Usage

```
hierband(Sighat, lam, w = NULL, delta = NULL, maxiter = 100,
  tol = 1e-07)
```

Arguments

`Sighat`	The sample covariance matrix
`lam`	Non-negative penalty parameter. Controls sparsity level.
`w`	`(p-1)`-by-`(p-1)` lower-triangular matrix (above diagonal ignored). `w[l,]` gives the `l` weights for g_l. Defaults to `w[l,m]=sqrt(2 * l)/(l - m + 1)` for `m <= l`
`delta`	Lower bound on eigenvalues. If this is NULL (which is default), then no eigenvalue constraint is included.
`maxiter`	Number of iterations of blockwise coordinate descent to perform.
`tol`	Only used when `delta` is non-NULL. When no eigenvalue changes by more than `tol` in BCD, convergence is assumed.

Value

Returns the convex banded estimate of covariance.

See Also

`hierband.path` `hierband.cv`

Examples

```
set.seed(123)
p <- 100
n <- 50
K <- 10
true <- ma(p, K)
```

```
x <- matrix(rnorm(n*p), n, p) %*% true$A
Sighat <- cov(x)
fit <- hierband(Sighat, lam=0.4)
min(eigen(fit)$values)
fit2 <- hierband(Sighat, lam=0.4, delta=0.2)
min(eigen(fit2)$values)
# Use cross validation to select lambda:
path <- hierband.path(Sighat)
cv <- hierband.cv(path, x)
fit <- hierband(Sighat, lam=cv$lam.best)
```

`hierband.cv` *Performs nfolds-cross validation*

Description

This function can be used to select a value of lam that performs well according to a user-specified measure of error.

Usage

```
hierband.cv(pathObj, x, errfun = function(est, true) sum((est - true)^2),
  nfolds = 5)
```

Arguments

`pathObj`	output of hierband.path
`x`	`n` by `p` matrix
`errfun`	a user-specified function measuring the loss incurred by estimating `est` (first argument) when the true covariance matrix is `true` (second argument). Default: Squared Frobenius norm.
`nfolds`	number of folds (default: 5)

Value

errs: A `nlam`-by-`nfolds` matrix of errors. `errs[i,j]` is error incurred in using `lamlist[i]` on fold `j`

m: CV error error for each value of lambda.

se: Standard error (estimated over folds) for each value of lambda

lam.best: Value of `lamlist` minimizing CV error.

ibest: Index of `lamlist` minimizing CV error.

lam.1se.rule: Selected value of lambda using the one-standard-error rule, a common heuristic that favors a sparser model when there isn't strong evidence against it.

i.1se.rule: Index of `lamlist` of one-standard-error rule.

See Also

hierband hierband.path

Examples

```
set.seed(123)
p <- 100
n <- 50
K <- 10
true <- ma(p, K)
x <- matrix(rnorm(n*p), n, p) %*% true$A
Sighat <- cov(x)
path <- hierband.path(Sighat)
cv <- hierband.cv(path, x)
fit <- hierband(Sighat, lam=cv$lam.best)
## Not run:
plot(cv$m, main="CV Frob Error", type="b")
lines(cv$m+cv$se, main="CV Frob Error")
lines(cv$m-cv$se, main="CV Frob Error")
abline(v=c(cv$ibest,cv$i.1se.rule), lty=2)

## End(Not run)
```

---

hierband.path *Solves main optimization problem over a grid of lambda values*

Description

See hierband for the problem this is solving. If `lamlist` not provided, then grid will be constructed starting at lambda_max, the smallest value of lam for which the solution (with `delta=NULL`) is diagonal.

Usage

```
hierband.path(Sighat, nlam = 20, flmin = 0.01, lamlist = NULL, w = NULL,
  delta = NULL, maxiter = 100, tol = 1e-07)
```

Arguments

Sighat	The sample covariance matrix
nlam	Number of lambda values to include in grid.
flmin	Ratio between the smallest lambda and largest lambda in grid. (Default: 0.01) Decreasing this gives less sparse solutions.
lamlist	A grid of lambda values to use. If this is non-NULL, then `nlam` and `flmin` are ignored.
w	`(p-1)-by-(p-1)` lower-triangular matrix (above diagonal ignored). `w[l,]` gives the `l` weights for g_l. Defaults to `w[l,m]=sqrt(2 * l)/(l - m + 1)` for `m <= l`

delta	Lower bound on eigenvalues. If this is NULL (which is default), then no eigenvalue constraint is included.
maxiter	Number of iterations of blockwise coordinate descent to perform.
tol	Only used when delta is non-NULL. When no eigenvalue changes by more than tol in BCD, convergence is assumed.

Value

Returns a sequence of convex banded estimates of the covariance matrix.

P: A nrow(Sighat)-by-nrow(Sighat)-by-nlam array where P[, , i] gives the ith estimate of the covariance matrix.

lamlist: Grid of lambda values used.

w: Value of w used.

delta: Value of delta used.

See Also

hierband hierband.cv

Examples

```
set.seed(123)
p <- 100
n <- 50
K <- 10
true <- ma(p, K)
x <- matrix(rnorm(n*p), n, p) %*% true$A
Sighat <- cov(x)
path <- hierband.path(Sighat)
cv <- hierband.cv(path, x)
fit <- hierband(Sighat, lam=cv$lam.best)
```

---

lam.max.hierband *Computes the smallest lambda such that P=0.*

Description

Computes lambda_max, which is the smallest value of lam for which hierband (with delta=NULL) gives a diagonal covariance matrix.

Usage

```
lam.max.hierband(Sighat, ww)
```

Arguments

Sighat	empirical covariance matrix
ww	the diagonal of w

ma *Covariance of an equal-weighted moving-average process*

Description

Here, `Sig[j,k] = 0` if `|j-k|>K` and `Sig[j,k] = 1 - |j-k| / K` otherwise.

Usage

```
ma(p, K)
```

Arguments

`p`	dimension of covariance matrix
`K`	moving-average bandwidth

Value

Returns the covariance matrix, `Sig`, and the symmetric square root, `A`, of this matrix.

MakeFolds *Make folds for cross validation*

Description

Divides the indices `1:n` into `nfolds` random folds of about the same size.

Usage

```
MakeFolds(n, nfolds)
```

Arguments

`n`	sample size
`nfolds`	number of folds

subdiag.thresh — *Performs a single pass of BCD on a matrix R.*

Description

To solve the unconstrained problem, R is Sigmahat. To solve constrained problem, R is the current partial residual (excluding A terms).

Usage

```
subdiag.thresh(R, lam, w = NULL)
```

Arguments

`R`	p-by-p symmetric matrix
`lam`	Non-negative penalty parameter. Controls sparsity level.
`w`	`(p-1)`-by-`(p-1)` lower-triangular matrix (above diagonal ignored). `w[l,]` gives the `l` weights for g_l. Defaults to `w[l,m]=sqrt(2 * l)/(l - m + 1)` for `m <= l`

subdiagonal.l2norms — *Compute the L2 norm of each subdiagonal of a symmetric matrix R.*

Description

Compute the L2 norm of each subdiagonal of a symmetric matrix R.

Usage

```
subdiagonal.l2norms(R)
```

Arguments

`R`	a symmetric matrix

Index