Package ‘CVglasso’

June 4, 2018

Type Package
Title Lasso Penalized Precision Matrix Estimation
Version 1.0
Date 2018-05-31
Description Estimates a lasso penalized precision matrix via the blockwise coordinate descent (BCD). This package is a simple wrapper around the popular 'glasso' package that extends and enhances its capabilities. These enhancements include built-in cross validation and visualizations.

URL https://github.com/MGallow/CVglasso

BugReports https://github.com/MGallow/CVglasso/issues
License GPL (>= 2)
ByteCompile TRUE
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Imports stats, parallel, foreach, ggplot2, dplyr, glasso
Depends doParallel
Suggests testthat
NeedsCompilation no
Author Matt Galloway [aut, cre]
Maintainer Matt Galloway <gall0441@umn.edu>
Repository CRAN
Date/Publication 2018-06-04 08:42:55 UTC

R topics documented:

CVglasso ............................................................... 2
plot.CVglasso .................................................... 5
Penalized precision matrix estimation using the graphical lasso (glasso) algorithm. Consider the case where \(X_1, \ldots, X_n\) are iid \(\mathcal{N}_p(\mu, \Sigma)\) and we are tasked with estimating the precision matrix, denoted \(\Omega \equiv \Sigma^{-1}\). This function solves the following optimization problem:

**Objective:**

\[
\hat{\Omega}_\lambda = \arg \min_{\Omega \in S_+^p} \left\{ Tr(S\Omega) - \log \det(\Omega) + \lambda \|\Omega\|_1 \right\}
\]

where \(\lambda > 0\) and we define \(\|A\|_1 = \sum_{i,j} |A_{ij}|\).

**Usage**

\[
\text{Cvglasso}(X = \text{NULL, } S = \text{NULL, } \text{nlam} = 10, \text{lamb.min.ratio} = 0.01, \text{lam} = \text{NULL, } \text{diagonal} = \text{FALSE, } \text{path} = \text{FALSE, } \text{tol} = 1e^{-04}, \text{maxit} = 10000, \text{adjmaxit} = \text{NULL, } K = 5, \text{crit.cv} = \text{c("loglik", "AIC", "BIC"), start} = \text{c("warm", "cold"), cores} = 1, \text{trace} = \text{c("progress", "print", "none")})
\]

**Arguments**

- **X**: option to provide a nxp data matrix. Each row corresponds to a single observation and each column contains n observations of a single feature/variable.
- **S**: option to provide a pxp sample covariance matrix (denominator n). If argument is NULL and X is provided instead then S will be computed automatically.
- **nlam**: number of lam tuning parameters for penalty term generated from lamb.min.ratio and lam.max (automatically generated). Defaults to 10.
- **lamb.min.ratio**: smallest lam value provided as a fraction of lam.max. The function will automatically generate nlam tuning parameters from lamb.min.ratio*lam.max to lam.max in log10 scale. lam.max is calculated to be the smallest lam such that all off-diagonal entries in Omega are equal to zero (alpha = 1). Defaults to 1e-2.
- **lam**: option to provide positive tuning parameters for penalty term. This will cause nlam and lamb.min.ratio to be disregarded. If a vector of parameters is provided, they should be in increasing order. Defaults to NULL.
- **diagonal**: option to penalize the diagonal elements of the estimated precision matrix (\(\Omega\)). Defaults to FALSE.
- **path**: option to return the regularization path. This option should be used with extreme care if the dimension is large. If set to TRUE, cores must be set to 1 and errors and optimal tuning parameters will based on the full sample. Defaults to FALSE.
- **tol**: convergence tolerance. Iterations will stop when the average absolute difference in parameter estimates in less than tol times multiple. Defaults to 1e-4.
- **maxit**: maximum number of iterations. Defaults to 1e4.
adjmaxit adjusted maximum number of iterations. During cross validation this option allows the user to adjust the maximum number of iterations after the first lam tuning parameter has converged. This option is intended to be paired with warm starts and allows for ’one-step’ estimators. Defaults to NULL.

K specify the number of folds for cross validation.

crit.cv cross validation criterion (loglik, AIC, or BIC). Defaults to loglik.

start specify warm or cold start for cross validation. Default is warm.

cores option to run CV in parallel. Defaults to cores = 1.

trace option to display progress of CV. Choose one of progress to print a progress bar, print to print completed tuning parameters, or none.

... additional arguments to pass to glasso.

Details

For details on the implementation of the ‘glasso’ function, see Tibshirani’s website. http://statweb.stanford.edu/~tibs/glasso/.

Value

returns class object CVglasso which includes:

Call function call.

Iterations number of iterations

Tuning optimal tuning parameters (lam and alpha).

Lambdas grid of lambda values for CV.

maxit maximum number of iterations for outer (blockwise) loop.

Omega estimated penalized precision matrix.

Sigma estimated covariance matrix from the penalized precision matrix (inverse of Omega).

Path array containing the solution path. Solutions will be ordered by ascending lambda values.

MIN.error minimum average cross validation error (cv.crit) for optimal parameters.

AVG.error average cross validation error (cv.crit) across all folds.

CV.error cross validation errors (cv.crit).

Author(s)

Matt Galloway <gall0441@umn.edu>
References

- Rothman, Adam. 2017. 'STAT 8931 notes on an algorithm to compute the Lasso-penalized Gaussian likelihood precision matrix estimator.'

See Also

plot.CVglasso

Examples

```R
# generate data from a sparse matrix
# first compute covariance matrix
S = matrix(0.7, nrow = 5, ncol = 5)
for (i in 1:5){
  for (j in 1:5){
    S[i, j] = S[i, j]*abs(i - j)
  }
}

# generate 100 x 5 matrix with rows drawn from iid N_p(0, S)
Z = matrix(rnorm(100*5), nrow = 100, ncol = 5)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$ vectors %*% diag(out$ values^0.5)
S.sqrt = S.sqrt %*% t(out$vectors)
X = Z %*% S.sqrt

# lasso penalty CV
CVglasso(X)
```
Description

Produces a plot for the cross validation errors, if available.

Usage

## S3 method for class 'CVglasso'
plot(x, type = c("line", "heatmap"), footnote = TRUE,
...)

Arguments

x class object CVglasso
type produce either 'heatmap' or 'line' graph
footnote option to print footnote of optimal values. Defaults to TRUE.
... additional arguments.

Examples

# generate data from a sparse matrix
# first compute covariance matrix
S = matrix(0.7, nrow = 5, ncol = 5)
for (i in 1:5){
  for (j in 1:5){
    S[i, j] = S[i, j]*abs(i - j)
  }
}

# generate 100 x 5 matrix with rows drawn from iid N_p(0, S)
Z = matrix(rnorm(100*5), nrow = 100, ncol = 5)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5)
S.sqrt = S.sqrt %*% t(out$vectors)
X = Z %*% S.sqrt

# produce line graph for CVglasso
plot(CVglasso(X))

# produce CV heat map for CVglasso
plot(CVglasso(X), type = 'heatmap')
Index

CVglasso, 2
plot.CVglasso, 4, 5