Package 'MGL'

January 20, 2025

Type Package

Title Module Graphical Lasso

Version 1.1

Date 2014-11-01

Author Safiye Celik

Maintainer Safiye Celik <safiye@cs.washington.edu>

Description An aggressive dimensionality reduction and network estimation technique for a high-dimensional Gaussian graphical model (GGM). Please refer to: Efficient Dimensionality Reduction for High-Dimensional Network Estimation, Safiye Celik, Benjamin A. Logsdon, Su-In Lee, Proceedings of The 31st International Conference on Machine Learning, 2014, p. 1953--1961.

License GPL (>= 2)

URL https://sites.google.com/a/cs.washington.edu/mgl/

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-11-05 13:33:01

Contents

MGL *Module network inference*

Description

Takes a high-dimensional data matrix, initial values of the module latent variables, and a penalty parameter, and returns the final assignment of the data points to the modules, the values of the module latent variables, and the conditional dependency network among the module latent variables.

Usage

```
MGL(data, L, lambda, printoutput = 0, maxiter = 100, threshold = 0.01)
```

Arguments

`data`	An nxp matrix which contains n samples from p variables, where typically p»n
`L`	An nxk matrix which contains the initial latent variable values, a column for each module
`lambda`	A penalty parameter controlling the sparsity of the conditional dependency network among the modules
`printoutput`	1 if the user wants the output from each iteration to be displayed, 0 for silent run
`maxiter`	Maximum number of iterations to be performed
`threshold`	Threshold for convergence

Value

`L`	An nxk matrix which contains the final latent variable values, a column for each module
`theta`	A kxk symmetric positive-semidefinite matrix respresenting the conditional dependency network among the modules
`Z`	A p-vector containing values between 1 to k, representing the assignment of the p variables to k modules

Examples

```
## Not run:
library(MGL)
n = 20 #sample size
p = 100 #variable size
k = 5 #module size
lambda = .01 #penalty parameter to induce sparsity
data = matrix(rnorm(n*p), ncol=p)
# to start with initial random module latent variables
L = matrix(rnorm(n*k), ncol=k)
MGL(data, L, lambda)
# to start with k-means cluster centroids as module latent variables
L = t(kmeans(t(data), k)$centers)
MGL(data, L, lambda)

## End(Not run)
```

Index