Package: fingraph (via r-universe)

March 1, 2025

Title Learning Graphs for Financial Markets

Version 0.1.0

Date 2023-02-02

Description Learning graphs for financial markets with optimization algorithms. This package contains implementations of the algorithms described in the paper: Cardoso JVM, Ying J, and Palomar DP (2021) <https://papers.nips.cc/paper/2021/hash/a64a034c3cb8eac64eb46ea474902797-Abstract.html> ``Learning graphs in heavy-tailed markets'', Advances in Neural Informations Processing Systems (NeurIPS).

URL https://github.com/convexfi/fingraph/

BugReports https://github.com/convexfi/fingraph/issues

License GPL-3

Encoding UTF-8

Depends spectralGraphTopology

Imports MASS, stats, progress, mvtnorm

Suggests testthat

RoxygenNote 7.1.1

NeedsCompilation no

Author Ze Vinicius [cre, aut]

Maintainer Ze Vinicius <jvmirca@gmail.com>

Date/Publication 2023-02-14 09:20:02 UTC

Additional_repositories https://cranhaven.r-universe.dev

Config/pak/sysreqs libxml2-dev

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

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/fingraph

RemoteSha 70bb222f452ef145989b6a01479ee351972e4a9b

RemoteSubdir fingraph

Contents

learn_connected_graph 2
learn_kcomp_heavytail_graph 3
learn_regular_heavytail_graph 5

Index **7**

`learn_connected_graph` *Laplacian matrix of a connected graph with Gaussian data Computes the Laplacian matrix of a graph on the basis of an observed data matrix, where we assume the data to be Gaussian distributed.*

Description

Laplacian matrix of a connected graph with Gaussian data

Computes the Laplacian matrix of a graph on the basis of an observed data matrix, where we assume the data to be Gaussian distributed.

Usage

```
learn_connected_graph(
  S,
  w0 = "naive",
  d = 1,
  rho = 1,
  maxiter = 10000,
  reltol = 1e-05,
  verbose = TRUE
)
```

Arguments

`S`	a p x p covariance matrix, where p is the number of nodes in the graph
`w0`	initial vector of graph weights. Either a vector of length p(p-1)/2 or a string indicating the method to compute an initial value.
`d`	the nodes' degrees. Either a vector or a single value.
`rho`	constraint relaxation hyperparameter.
`maxiter`	maximum number of iterations.
`reltol`	relative tolerance as a convergence criteria.
`verbose`	whether or not to show a progress bar during the iterations.

Value

A list containing possibly the following elements:

`laplacian`	estimated Laplacian matrix
`adjacency`	estimated adjacency matrix
`theta`	estimated Laplacian matrix slack variable
`maxiter`	number of iterations taken to reach convergence
`convergence`	boolean flag to indicate whether or not the optimization converged

---

`learn_kcomp_heavytail_graph` *Laplacian matrix of a k-component graph with heavy-tailed data Computes the Laplacian matrix of a graph on the basis of an observed data matrix, where we assume the data to be Student-t distributed.*

Description

Laplacian matrix of a k-component graph with heavy-tailed data

Computes the Laplacian matrix of a graph on the basis of an observed data matrix, where we assume the data to be Student-t distributed.

Usage

```
learn_kcomp_heavytail_graph(
  X,
  k = 1,
  heavy_type = "gaussian",
  nu = NULL,
  w0 = "naive",
  d = 1,
  beta = 1e-08,
  update_beta = TRUE,
  early_stopping = FALSE,
  rho = 1,
  update_rho = FALSE,
  maxiter = 10000,
  reltol = 1e-05,
  verbose = TRUE,
  record_objective = FALSE
)
```

Arguments

`X` an n x p data matrix, where n is the number of observations and p is the number of nodes in the graph.

`k` the number of components of the graph.

`heavy_type` a string which selects the statistical distribution of the data . Valid values are "gaussian" or "student".

`nu` the degrees of freedom of the Student-t distribution. Must be a real number greater than 2.

`w0` initial vector of graph weights. Either a vector of length p(p-1)/2 or a string indicating the method to compute an initial value.

`d` the nodes' degrees. Either a vector or a single value.

`beta` hyperparameter that controls the regularization to obtain a k-component graph

`update_beta` whether to update beta during the optimization.

`early_stopping` whether to stop the iterations as soon as the rank constraint is satisfied.

`rho` constraint relaxation hyperparameter.

`update_rho` whether or not to update rho during the optimization.

`maxiter` maximum number of iterations.

`reltol` relative tolerance as a convergence criteria.

`verbose` whether to show a progress bar during the iterations.

`record_objective` whether to record the objective function per iteration.

Value

A list containing possibly the following elements:

`laplacian` estimated Laplacian matrix

`adjacency` estimated adjacency matrix

`theta` estimated Laplacian matrix slack variable

`maxiter` number of iterations taken to reach convergence

`convergence` boolean flag to indicate whether or not the optimization conv erged

`beta_seq` sequence of values taken by the hyperparameter beta until convergence

`primal_lap_residual` primal residual for the Laplacian matrix per iteratio n

`primal_deg_residual` primal residual for the degree vector per iteration

`dual_residual` dual residual per iteration

`lagrangian` Lagrangian value per iteration

`elapsed_time` Time taken to reach convergence

learn_regular_heavytail_graph

Laplacian matrix of a connected graph with heavy-tailed data Computes the Laplacian matrix of a graph on the basis of an observed data matrix, where we assume the data to be Student-t distributed.

Description

Laplacian matrix of a connected graph with heavy-tailed data

Computes the Laplacian matrix of a graph on the basis of an observed data matrix, where we assume the data to be Student-t distributed.

Usage

```
learn_regular_heavytail_graph(
  X,
  heavy_type = "gaussian",
  nu = NULL,
  w0 = "naive",
  d = 1,
  rho = 1,
  update_rho = TRUE,
  maxiter = 10000,
  reltol = 1e-05,
  verbose = TRUE
)
```

Arguments

`X`	an n x p data matrix, where n is the number of observations and p is the number of nodes in the graph
`heavy_type`	a string which selects the statistical distribution of the data. Valid values are "gaussian" or "student".
`nu`	the degrees of freedom of the Student-t distribution. Must be a real number greater than 2.
`w0`	initial vector of graph weights. Either a vector of length p(p-1)/2 or a string indicating the method to compute an initial value.
`d`	the nodes' degrees. Either a vector or a single value.
`rho`	constraint relaxation hyperparameter.
`update_rho`	whether or not to update rho during the optimization.
`maxiter`	maximum number of iterations.
`reltol`	relative tolerance as a convergence criteria.
`verbose`	whether or not to show a progress bar during the iterations.

Value

A list containing possibly the following elements:

`laplacian`	estimated Laplacian matrix
`adjacency`	estimated adjacency matrix
`theta`	estimated Laplacian matrix slack variable
`maxiter`	number of iterations taken to reach convergence
`convergence`	boolean flag to indicate whether or not the optimization conv erged
`primal_lap_residual`	primal residual for the Laplacian matrix per iteration
`primal_deg_residual`	primal residual for the degree vector per iteration
`dual_residual`	dual residual per iteration
`lagrangian`	Lagrangian value per iteration
`elapsed_time`	Time taken to reach convergence

Index

learn_connected_graph, 2
learn_kcomp_heavytail_graph, 3
learn_regular_heavytail_graph, 5