Package ‘CASCORE’

August 17, 2022

Type Package

Title Covariate Assisted Spectral Clustering on Ratios of Eigenvectors

Version 0.1.0

Author Yaofang Hu [aut, cre],
Wanjie Wang [aut]

Maintainer Yaofang Hu <yaofangh@smu.edu>

Description Functions for the novel algorithm CASCORE, proposed to detect the latent community structure in graphs with node covariates. The models we can handle include covariate assisted degree corrected stochastic block model (CADCSBM). CASCORE allows for the disagreement between the community structure revealed in the adjacency information and the community structure revealed in the covariate information. More details are in the reference paper: Yaofang Hu and Wanjie Wang (2022) <arXiv:2208.00257>.
This package also includes other classical community detection algorithms that are compared to CASCORE in our paper, such as Spectral Clustering On Ratios-of Eigenvectors (SCORE), normalized PCA, ordinary PCA and covariate-assisted spectral clustering (CASC).

Imports stats, pracma, igraph

License GPL-2

Encoding UTF-8

URL https://arxiv.org/abs/2006.03284

RoxygenNote 7.2.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-17 06:40:09 UTC

R topics documented:

CASC — *Covariate Assisted Spectral Clustering.*

Description

CASC clusters graph nodes by applying spectral clustering with the assistance from node covariates.

Usage

```
CASC(Adj, Covariate, K, alphan = 5, itermax = 100, startn = 10)
```

Arguments

`Adj`	A 0/1 adjacency matrix.
`Covariate`	A covariate matrix. The rows correspond to nodes and the columns correspond to covariates.
`K`	A positive integer, indicating the number of underlying communities in graph `Adj`.
`alphan`	A tuning parameter to balance between the contributions of the graph and the covariates.
`itermax`	`k-means` parameter, indicating the maximum number of iterations allowed. The default value is 100.
`startn`	`k-means` parameter. If centers is a number, how many random sets should be chosen? The default value is 10.

Details

CASC is a community detection algorithm for networks with node covariates, proposed in *Covariate-assisted spectral clustering* of Binkiewicz, et al. (2017). *CASC* applies `k-means` on the first `K` leading eigenvectors of the balanced matrix between the Laplacian matrix and the covariate matrix.

Value

A label vector.

Examples

```
# Simulate the Network
n = 10; K = 2;
theta = 0.4 + (0.45-0.05)*(seq(1:n)/n)^2; Theta = diag(theta);
P  = matrix(c(0.8, 0.2, 0.2, 0.8), byrow = TRUE, nrow = K)
set.seed(2022)
l = sample(1:K, n, replace=TRUE); # node labels
Pi = matrix(0, n, K) # label matrix
for (k in 1:K){
  Pi[l == k, k] = 1
}
Omega = Theta %*% Pi %*% P %*% t(Pi) %*% Theta;
Adj = matrix(runif(n*n, 0, 1), nrow = n);
Adj = Omega - Adj;
Adj = 1*(Adj >= 0)
diag(Adj) = 0
Adj[lower.tri(Adj)] = t(Adj)[lower.tri(Adj)]
caseno = 4; Nrange = 10; Nmin = 10; prob1 = 0.9; p = n*4;
Q = matrix(runif(p*K, 0, 1), nrow = p, ncol = K)
Q = sweep(Q,2,colSums(Q),`/`)
W = matrix(0, nrow = n, ncol = K);
for(jj in 1:n) {
  if(runif(1) <= prob1) {W[jj, 1:K] = Pi[jj, ];}
  else W[jj, sample(K, 1)] = 1;
}
W = t(W)
D0 = Q %*% W
X = matrix(0, n, p)
N = switch(caseno, rep(100, n), rep(100, n), round(runif(n)*Nrange+ Nmin),
  round(runif(n)* Nrange+Nmin))
for (i in 1: ncol(D0)){
  X[i, ] = rmultinom(1, N[i], D0[, i])
}
CASC(Adj, X, 2)
```

CASCORE *Covariate Assisted Spectral Clustering on Ratios of Eigenvectors.*

Description

Using ratios-of-eigenvectors to detect underlying communities in networks with node covariates.

Usage

```
CASCORE(Adj, Covariate, K, alpha = NULL, itermax = 100, startn = 10, L = NULL)
```

Arguments

Adj	A 0/1 adjacency matrix.
Covariate	A covariate matrix. The rows correspond to nodes and the columns correspond to covariates.
K	A positive integer, indicating the number of underlying communities in graph Adj.
alpha	A numeric vector, each element of which is a tuning parameter to weigh the covariate matrix.
itermax	k-means parameter, indicating the maximum number of iterations allowed. The default value is 100.

`startn`	`k-means` parameter. If centers is a number, how many random sets should be chosen? The default value is 10.
`L`	Either 1 or 2. 1 indicates L_1 norm and 2 indicates L_2 norm. The default values is 2.

Details

CASCORE is fully established in *Covariate-Assisted Community Detection on Sparse Networks* of Hu & Wang (2022). *CASCORE* detects the latent community structure under the covariate assisted degree corrected stochastic block model (CADCSBM), and it allows the disagreement between the community structures indicated in the graph and the covariates, respectively. `K-means` is applied on the entry-wise ratios between first leading eigenvector and each of the other K leading eigenvectors of the combined matrix of the adjacency matrix and the covariate matrix, to reveal the underlying memberships.

Value

`estall`	A lavel vector
`optimal_alpha`	The optimal α that gives the smallest within-group variances.
`within_var`	The values of within-group variances for each value of α.

References

Hu, Y., & Wang, W. (2022) *Covariate-Assisted Community Detection on Sparse Networks*, https://arxiv.org/abs/2208.00257

Examples

```
# Simulate the Network
n = 10; K = 2;
theta = 0.4 + (0.45-0.05)*(seq(1:n)/n)^2; Theta = diag(theta);
P  = matrix(c(0.8, 0.2, 0.2, 0.8), byrow = TRUE, nrow = K)
set.seed(2022)
l = sample(1:K, n, replace=TRUE); # node labels
Pi = matrix(0, n, K) # label matrix
for (k in 1:K){
  Pi[l == k, k] = 1
}
Omega = Theta %*% Pi %*% P %*% t(Pi) %*% Theta;
Adj = matrix(runif(n*n, 0, 1), nrow = n);
Adj = Omega - Adj;
Adj = 1*(Adj >= 0)
diag(Adj) = 0
Adj[lower.tri(Adj)] = t(Adj)[lower.tri(Adj)]
caseno = 4; Nrange = 10; Nmin = 10; prob1 = 0.9; p = n*4;
Q = matrix(runif(p*K, 0, 1), nrow = p, ncol = K)
Q = sweep(Q,2,colSums(Q),`/`)
W = matrix(0, nrow = n, ncol = K);
```

```
for(jj in 1:n) {
  if(runif(1) <= prob1) {W[jj, 1:K] = Pi[jj, ];}
  else W[jj, sample(K, 1)] = 1;
}
W = t(W)
D0 = Q %*% W
X = matrix(0, n, p)
N = switch(caseno, rep(100, n), rep(100, n), round(runif(n)*Nrange+ Nmin),
  round(runif(n)* Nrange+Nmin))
for (i in 1: ncol(D0)){
  X[i, ] = rmultinom(1, N[i], D0[, i])
}
CASCORE(Adj, X, 2)
```

---

nPCA *Normalized Principle Component Analysis.*

Description

Normalized Principle Component Analysis (nPCA), also known as spectral clustering on the graph Laplacian, is a classical spectral clustering method that applies `k-means` on the first K leading (unit-norm) eigenvectors of the degree-corrected normalized graph laplacian.

Usage

```
nPCA(Adj, K, tau = NULL, itermax = 100, startn = 10)
```

Arguments

`Adj`	A 0/1 adjacency matrix.
`K`	A positive integer, indicating the number of underlying communities in graph `Adj`.
`tau`	An optional regularization parameter for suitable degree normalization. The default value is the average degree of graph `Adj`.
`itermax`	`k-means` parameter, indicating the maximum number of iterations allowed. The default value is 100.
`startn`	`k-means` parameter. If centers is a number, how many random sets should be chosen? The default value is 10.

Value

A label vector.

References

Chung, F. R., & Graham, F. C. (1997). *Spectral graph theory (Vol. 92). American Mathematical Soc..*

Examples

```
# Simulate the Network
n = 10; K = 2;
theta = 0.4 + (0.45-0.05)*(seq(1:n)/n)^2; Theta = diag(theta);
P  = matrix(c(0.8, 0.2, 0.2, 0.8), byrow = TRUE, nrow = K)
set.seed(2022)
l = sample(1:K, n, replace=TRUE); # node labels
Pi = matrix(0, n, K) # label matrix
for (k in 1:K){
  Pi[l == k, k] = 1
}
Omega = Theta %*% Pi %*% P %*% t(Pi) %*% Theta;
Adj = matrix(runif(n*n, 0, 1), nrow = n);
Adj = Omega - Adj;
Adj = 1*(Adj >= 0)
diag(Adj) = 0
Adj[lower.tri(Adj)] = t(Adj)[lower.tri(Adj)]
nPCA(Adj, 2)
```

---

oPCA *Ordinary Principle Component Analysis.*

Description

Ordinary Principle Component Analysis (oPCA), also known as spectral clustering on the adjacency matrix is a classical spectral clustering method that applies `k-means` on the first K leading (unit-norm) eigenvectors of the adjacency matrix of a graph.

Usage

```
oPCA(Adj, K, itermax = 100, startn = 10)
```

Arguments

`Adj`	A 0/1 adjacency matrix.
`K`	A positive integer, indicating the number of underlying communities in graph `Adj`.
`itermax`	`k-means` parameter, indicating the maximum number of iterations allowed. The default value is 100.
`startn`	`k-means` parameter. If centers is a number, how many random sets should be chosen? The default value is 10.

Value

A label vector.

References

Chung, F. R., & Graham, F. C. (1997). *Spectral graph theory (Vol. 92) American Mathematical Soc..*

Examples

```
# Simulate the Network
n = 10; K = 2;
theta = 0.4 + (0.45-0.05)*(seq(1:n)/n)^2; Theta = diag(theta);
P  = matrix(c(0.8, 0.2, 0.2, 0.8), byrow = TRUE, nrow = K)
set.seed(2022)
l = sample(1:K, n, replace=TRUE); # node labels
Pi = matrix(0, n, K) # label matrix
for (k in 1:K){
  Pi[l == k, k] = 1
}
Omega = Theta %*% Pi %*% P %*% t(Pi) %*% Theta;
Adj = matrix(runif(n*n, 0, 1), nrow = n);
Adj = Omega - Adj;
Adj = 1*(Adj >= 0)
diag(Adj) = 0
Adj[lower.tri(Adj)] = t(Adj)[lower.tri(Adj)]
oPCA(Adj, 2)
```

SCORE *Spectral Clustering On Ratios-of-Eigenvectors.*

Description

Using ratios-of-eigenvectors to detect underlying communities.

Usage

```
SCORE(G, K, itermax = NULL, startn = NULL)
```

Arguments

`G`	A 0/1 adjacency matrix of a connected graph.
`K`	A positive integer, indicating the number of underlying communities in graph `G`.
`itermax`	`k-means` parameter, indicating the maximum number of iterations allowed. The default value is 100.
`startn`	`k-means` parameter. If centers is a number, how many random sets should be chosen? The default value is 10.

Details

SCORE is fully established in *Fast community detection by SCORE* of Jin (2015). *SCORE* uses the entry-wise ratios between the first leading eigenvector and each of the other $K - 1$ leading eigenvectors for clustering. It is noteworthy that SCORE only works on connected graphs, in other words, it does not allow for isolated vertices.

Value

A label vector.

References

Jin, J. (2015) *Fast community detection by score*. *The Annals of Statistics 43 (1), 57–89*
https://projecteuclid.org/euclid.aos/1416322036

Examples

```
# Simulate the Network
n = 10; K = 2;
theta = 0.4 + (0.45-0.05)*(seq(1:n)/n)^2; Theta = diag(theta);
P  = matrix(c(0.8, 0.2, 0.2, 0.8), byrow = TRUE, nrow = K)
set.seed(2022)
l = sample(1:K, n, replace=TRUE); # node labels
Pi = matrix(0, n, K) # label matrix
for (k in 1:K){
  Pi[l == k, k] = 1
}
Omega = Theta %*% Pi %*% P %*% t(Pi) %*% Theta;
Adj = matrix(runif(n*n, 0, 1), nrow = n);
Adj = Omega - Adj;
Adj = 1*(Adj >= 0)
diag(Adj) = 0
Adj[lower.tri(Adj)] = t(Adj)[lower.tri(Adj)]
library(igraph)
is.igraph(Adj) # [1] FALSE
ix = components(graph.adjacency(Adj))
componentLabel = ix$membership
giantLabel = which(componentLabel == which.max(ix$csize))
Giant = Adj[giantLabel, giantLabel]
SCORE(Giant, 2)
```

Index