Package ‘cencrne’

January 9, 2023

Title Consistent Estimation of the Number of Communities via Regularized Network Embedding

Version 1.0.0

Description The network analysis plays an important role in numerous application domains including biomedicine.
Estimation of the number of communities is a fundamental and critical issue in network analysis. Most existing studies assume that the number of communities is known a priori, or lack of rigorous theoretical guarantee on the estimation consistency. This method proposes a regularized network embedding model to simultaneously estimate the community structure and the number of communities in a unified formulation.
The proposed model equips network embedding with a novel composite regularization term, which pushes the embedding vector towards its center and collapses similar community centers with each other. A rigorous theoretical analysis is conducted, establishing asymptotic consistency in terms of community detection and estimation of the number of communities.
Reference:
Ren, M., Zhang S. and Wang J. (2022). ``Consistent Estimation of the Number of Communities via Regularized Network Embedding''. Biometrics, <doi:10.1111/biom.13815>.

License GPL-2

Encoding UTF-8

Imports MASS, Matrix

LazyData true

LazyLoad yes

RoxygenNote 7.1.2

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Mingyang Ren [aut, cre] (<https://orcid.org/0000-0002-8061-9940>),
Sanguo Zhang [aut],
Junhui Wang [aut]

Maintainer Mingyang Ren <renmingyang17@mails.ucas.ac.cn>

Repository CRAN

Date/Publication 2023-01-09 10:00:05 UTC

R topics documented:

`evaluation` *Consistent Estimation of the Number of Communities via Regularized Network Embedding.*

Description

The evaluation function for Consistent Estimation of the Number of Communities via Regularized Network Embedding.

Usage

```
evaluation(Z.hat, Z.true, cluster.matrix.hat, cluster.matrix.true,
                  P.true, Theta.true, K.hat=4, K.true=4)
```

Arguments

`Z.hat`	A n * r matrix, the estimated embedding vectors corresponding to n nodes.
`Z.true`	A n * r matrix, the true embedding vectors corresponding to n nodes.
`cluster.matrix.hat`	A n * n estimated membership matrix, whose (i,j)-element is 1, if nodes i and j are estimated to belong to the same community, and 0, otherwise.
`cluster.matrix.true`	A n * n true membership matrix, whose (i,j)-element is 1, if nodes i and j belong to the same community, and 0, otherwise.
`P.true`	A n * n true probability matrix.
`Theta.true`	A n * n true matrix: Z.true %*% t(Z.true).
`K.hat`	The true number of communities.
`K.true`	The estimated number of communities.

Value

A vector including five evaluation index. prop. 1: the estimated and actual number of communities are equal; 0: not equal.

Author(s)

Mingyang Ren.

`example.data` *Some example data*

Description

Some example data

Format

A list including: A: An observed n * n adjacency matrix of undirected graph, n=360. K.true: The estimated number of communities. Z.true: A n * r matrix, the true embedding vectors corresponding to n nodes, n=360, r=5. B.true: A n * r matrix, the true community centers corresponding to n nodes. P.true: A n * n true probability matrix. Theta.true: A n * n true matrix: Z.true %*% t(Z.true). cluster.matrix.true: A n * n true membership matrix, whose (i,j)-element is 1, if nodes i and j belong to the same community, and 0, otherwise.

Source

Simulated data

Examples

```
data(example.data)
```

`gen.int` *Consistent Estimation of the Number of Communities via Regularized Network Embedding.*

Description

The function generating the initial values.

Usage

```
gen.int(A, R=8, K.max0=8,rand.seed=123,
                lambda3=0, a=3, kappa=1, alpha=1,
                eps = 1e-2, niter = 20, niter.Z=5)
```

Arguments

`A`	An observed n * n adjacency matrix of undirected graph.
`R`	Int, the relatively large dimension of embedding vectors given in advance.
`K.max0`	The relatively large upper bound of the number of communities given in advance to generate initial values of B.
`rand.seed`	The random seed of generating initial value.

`lambda3`	A float value, the tuning parameter for sparsity of Z.
`a`	A float value, regularization parameter in MCP, the default setting is 3.
`kappa`	A float value, the penalty parameter in ADMM algorithm, the default setting is 1.
`alpha`	A float value, the step size of coordinate descent algorithm updating Z, the default setting is 1.
`eps`	A float value, algorithm termination threshold.
`niter`	Int, maximum number of cycles of the overall ADMM algorithm.
`niter.Z`	Int, maximum number of cycles of coordinate descent algorithm updating Z.

Value

A list including all estimated parameters and the BIC values with all choices of given tuning parameters, and the selected optional parameters. Opt_Z: A n * r matrix, the estimated embedding vectors corresponding to n nodes; Opt_B: A n * r matrix, the estimated community centers corresponding to n nodes; Opt_K: Int, the estimated number of communities; Opt_member: A n-dimensional vector, describing the membership of n nodes; Opt_cluster.matrix: A n * n membership matrix, whose (i,j)-element is 1, if nodes i and j belong to the same community, and 0, otherwise.

Author(s)

Mingyang Ren.

References

Ren, M., Zhang S., Zhang Q. and Ma S. (2022). Consistent Estimation of the Number of Communities via Regularized Network Embedding.

Examples

```
library(cencrne)
data(example.data)
A                   = example.data$A
K.true              = example.data$K.true
Z.true              = example.data$Z.true
B.true              = example.data$B.true
P.true              = example.data$P.true
Theta.true          = example.data$Theta.true
cluster.matrix.true = example.data$cluster.matrix.true

n              = dim(A)[1]
sample.index.n = rbind(combn(n,2),1:(n*(n-1)/2))
int.list       = gen.int(A)
Z.int          = int.list$Z.int
B.int          = int.list$B.int
```

genelambda.obo *Generate tuning parameters*

Description

Generating a sequence of the tuning parameters (lambda1, lambda2, and lambda3).

Usage

```
genelambda.obo(nlambda1=10,lambda1_max=1,lambda1_min=0.05,
                           nlambda2=10,lambda2_max=1,lambda2_min=0.01,
                           nlambda3=10,lambda3_max=5,lambda3_min=0.5)
```

Arguments

nlambda1	The numbers of lambda 1.
lambda1_max	The maximum values of lambda 1.
lambda1_min	The minimum values of lambda 1.
nlambda2	The numbers of lambda 2.
lambda2_max	The maximum values of lambda 2.
lambda2_min	The minimum values of lambda 2.
nlambda3	The numbers of lambda 3.
lambda3_max	The maximum values of lambda 3.
lambda3_min	The minimum values of lambda 3.

Value

A sequence of the tuning parameters (lambda1, lambda2, and lambda3).

Author(s)

Mingyang Ren

Examples

```
lambda <- genelambda.obo(nlambda1=5,lambda1_max=0.5,lambda1_min=0.1, nlambda2=15,lambda2_max=1.5,
                            lambda2_min=0.1, nlambda3=10,lambda3_max=3.5,lambda3_min=0.5)
lambda
```

`network.comm.num` *Consistent Estimation of the Number of Communities via Regularized Network Embedding.*

Description

The main function for Consistent Estimation of the Number of Communities via Regularized Network Embedding.

Usage

```
network.comm.num(A, sample.index.n, lambda, Z.int, B.int,
                        a=3, kappa=1, alpha=1, eps=5e-2, niter=20,
               niter.Z=5, update.B="ADMM",local.oppro=FALSE, merge.all=TRUE,
               ad.BIC=FALSE, Fully.Connected=TRUE, trace=FALSE,
               line.search=TRUE, ad.BIC.B=FALSE)
```

Arguments

`A`	An observed n * n adjacency matrix of undirected graph.
`sample.index.n`	A 3 * (n*(n-1)/2) matrix, all pairs of integers from 1 to n.
`lambda`	A list, the sequences of the tuning parameters (λ_1, λ_2, and λ_3).
`Z.int`	A n * r matrix, the initial values of embedding vectors corresponding to n nodes.
`B.int`	A n * r matrix, the initial values of community centers corresponding to n nodes.
`a`	A float value, regularization parameter in MCP, the default setting is 3.
`kappa`	A float value, the penalty parameter in ADMM algorithm, the default setting is 1.
`alpha`	A float value, the step size of coordinate descent algorithm updating Z, the default setting is 1.
`eps`	A float value, algorithm termination threshold.
`niter`	Int, maximum number of cycles of the overall ADMM algorithm.
`niter.Z`	Int, maximum number of cycles of coordinate descent algorithm updating Z.
`update.B`	The optimization algorithm updating B, which can be selected "ADMM" (default) and "AMA".
`local.oppro`	The logical variable, whether to use local approximations when updating Z, the default setting is F.
`merge.all`	Whether to merge pairs of nodes indirectly connected (but without the direct edge) in the estimated community membership matrix.
`ad.BIC`	Whether to use the adjusted BIC, the default setting is F.
`Fully.Connected`	Whether to use the all pairs (i,j) in fusion penalty, the default setting is T. If F, the pairs (i,j) in fusion penalty will be determined by the observed n * n adjacency matrix A.

`trace`	Whether to output the intermediate process of the algorithm.
`line.search`	Linear search or not, the default setting is T.
`ad.BIC.B`	Whether the BIC criterion contains terms involving the B matrix, the default setting is F.

Value

A list including all estimated parameters and the BIC values with all choices of given tuning parameters, and the selected optional parameters. Opt_Z: A n * r matrix, the estimated embedding vectors corresponding to n nodes; Opt_B: A n * r matrix, the estimated community centers corresponding to n nodes; Opt_K: Int, the estimated number of communities; Opt_member: A n-dimensional vector, describing the membership of n nodes; Opt_cluster.matrix: A n * n membership matrix, whose (i,j)-element is 1, if nodes i and j belong to the same community, and 0, otherwise.

Author(s)

Mingyang Ren, Sanguo Zhang, Junhui Wang. Maintainer: Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Ren, M., Zhang S. and Wang J. (2022). Consistent Estimation of the Number of Communities via Regularized Network Embedding.

Examples

```
library(cencrne)
data(example.data)
A                   = example.data$A
K.true              = example.data$K.true
Z.true              = example.data$Z.true
B.true              = example.data$B.true
P.true              = example.data$P.true
Theta.true          = example.data$Theta.true
cluster.matrix.true = example.data$cluster.matrix.true

n       = dim(A)[1]
lam.max = 3
lam.min = 0.5
lam1.s  = 2/log(n)
lam2.s  = sqrt(8*log(n)/n)
lam3.s  = 1/8/log(n)/sqrt(n)
lambda  = genelambda.obo(nlambda1=3,lambda1_max=lam.max*lam1.s,lambda1_min=lam.min*lam1.s,
                      nlambda2=10,lambda2_max=lam.max*lam2.s,lambda2_min=lam.min*lam2.s,
                       nlambda3=1,lambda3_max=lam.max*lam3.s,lambda3_min=lam.min*lam3.s)

sample.index.n = rbind(combn(n,2),1:(n*(n-1)/2))
int.list       = gen.int(A)
Z.int          = int.list$Z.int
B.int          = int.list$B.int
res            = network.comm.num(A, sample.index.n, lambda, Z.int, B.int)
```

```
K.hat = res$Opt_K # the estimated number of communities
Z.hat = res$Opt_Z # the estimated embedding vectors corresponding to n nodes
cluster.matrix.hat = res$Opt_cluster.matrix # the n * n estimated membership matrix
evaluation(Z.hat, Z.true, cluster.matrix.hat, cluster.matrix.true,
           P.true, Theta.true, K.hat, K.true)
```

Index