Package 'Rclust'

April 8, 2021

Type Package

Title Randomized spectral clustering for large-scale networks

Version 0.1.0

Author Who wrote it

Maintainer The package maintainer <yourself@somewhere.net>

Description
This package implements spectral clustering for large-scale directed and undirected networks using randomization techniques including the random projection and the random sampling.

License GPL

Encoding UTF-8

LazyData true

Imports Rcpp,
RSpectra,
irlba,
stats,
RcppZiggurat,
Matrix,
Gmedian,
igraph

LinkingTo Rcpp, RcppEigen

SystemRequirements C++11

RoxygenNote 7.1.1

R topics documented:

`rclust` *Randomized spectral clustering using random sampling or random projection*

Description

Randomized spectral clustering for undirected networks. The clusters are computed using two random schemes, namely, the random sampling and the random projection scheme. Can deal with very large networks.

Usage

```
rclust(
  A,
  method = c("rsample", "rproject"),
  k,
  rank,
  p = 10,
  q = 2,
  dist = "normal",
  abs = FALSE,
  P,
  iter.max = 50,
  nstart = 10,
  ...
)
```

Arguments

`A`	The adjacency matrix of an undirected network (binary and symmetric) with type "dgCMatrix".
`method`	The method for computing the randomized eigendecomposition. Random sampling-based eigendecomposition is implemented if `method="rsample"`, and random projection-based eigendecomposition is implemented if `method="rproject"`.
`k`	The number of target clusters.
`rank`	The number of target rank.
`p`	The oversampling parameter in the random projection scheme. Requested only if `method="rproject"`. Default is 10.
`q`	The power parameter in the random projection scheme. Requested only if `method="rproject"`. Default is 2.
`dist`	The distribution of the entry of the random test matrix in the random projection scheme. Requested only if `method="rproject"`. Default is `"normal"`.
`abs`	A logical variable indicating whether the eigen values should be largest in absolute value. Default is `FALSE`, indicating that the eigen values are largest in value.
`P`	The sampling probability in the random sampling scheme. Requested only if `method="rsample"`.
`iter.max`	Maximum number of iterations in the `kmeans`. Default is 50.
`nstart`	The number of random sets in `kmeans`. Default is 10.
`...`	Additional arguments.

Details

This function computes the clusters of undirected networks using randomized spectral clustering algorithms. The random projection-based eigendecomposition or the random sampling-based eigendecomposition is first computed for the adjacency matrix of the undirected network. The k-means is then performed on the randomized eigen vectors.

Value

`cluster`	The cluster vector (from `1:k`) with the numbers indicating which cluster each node is assigned.
`rvectors`	The randomized `rank` eigen vectors computed by `reig.pro` or `reig.sam`.

See Also

`reig.pro`, `reig.sam`.

Examples

```
n <- 100
k <- 2
clustertrue <- rep(1:k, each = n/k)
A <- matrix(0, n, n)
for(i in 1: (n-1)) {
   for(j in (i+1):n) {
       A[i, j] <- ifelse(clustertrue[i] == clustertrue[j], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
       A[j, i] <- A[i, j]
    }
}
A <- as(A, "dgCMatrix")
rclust(A, method = "rsample", k = k, rank = k, P = 0.7)
```

---

reig.pro *Compute randomized eigenvalue decomposition of the adjacency matrix of undirected networks using random projection*

Description

Compute the randomized eigenvalue decomposition of an adjacency matrix (0-1 coded) by random projection. The randomized eigen vectors and eigen values are computed. Can deal with very large data matrix.

Usage

```
reig.pro(
  A,
  rank,
  p = 10,
  q = 2,
  dist = "normal",
  approA = FALSE,
```

```
  nthread = 1,
  abs = FALSE
)
```

Arguments

A	Input data matrix of class "dgCMatrix". The matrix is assumed to be binary, symmetric, and sparse, with zeros on the diagonal.
rank	The target rank of the low-rank decomposition.
p	The oversampling parameter. It need to be a positive integer number. Default value is 10.
q	The power parameter. It need to be a positive integer number. Default value is 2.
dist	The distribution of the entry of the random test matrix. Can be "normal" (standard normal distribution), "unif" (uniform distribution from -1 to 1), or "rademacher" (randemacher distribution). Default is "normal".
approA	A logical variable indicating whether the approximated A is returned. Default is FALSE.
nthread	Maximum number of threads for specific computations that could be implemented in parallel. Default is 1.
abs	A logical variable indicating whether the rank+p eigen values should be largest in absolute value. Default is FALSE, indicating that the eigen values are largest in value.

Details

This function computes the randomized eigen value decomposition of an adjacency matrix using the random projection scheme. The data matrix A is symmetric and binary. It is first compressed to a smaller matrix with its columns (rows) being the linear combinations of the columns (rows) of A. The classical eigen value decomposition is then performed on the smaller matrix. The randomized eigen value decomposition of A are obtained by postprocessing.

Value

vectors	The randomized rank+p eigen vectors.
values	The rank+p eigen values.
approA	The approximated data matrix if requested.

References

N. Halko, P.-G. Martinsson, and J. A. Tropp. (2011) *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, *SIAM review, Vol. 53(2), 217-288*
https://epubs.siam.org/doi/10.1137/090771806

Examples

```
set.seed(123)
n <- 100
rank <- 2
clustertrue <- rep(1:rank, each = n/rank)
A <- matrix(0, n, n)
for(i in 1:(n - 1)) {
    for(j in (i + 1):n) {
        A[i, j] <- ifelse(clustertrue[i] == clustertrue[j], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
    }
}
diag(A) <- 0
A <- A + t(A)
A <- as(A, "dgCMatrix")
reig.pro(A, rank)
```

`reig.sam` *Compute randomized eigenvalue decomposition of the adjacency matrix of undirected networks using random sampling*

Description

Compute the randomized eigenvalue decomposition of the adjacency matrix (0-1 coded) by random sampling. The randomized eigen vectors and eigen values are computed. Can deal with very large data matrix.

Usage

```
reig.sam(A, P, use_lower = TRUE, k, tol = 1e-05, abs = FALSE, ...)
```

Arguments

`A`	An input sparse and binary data matrix of type "dgCMatrix". `A` is not necessarily a symmetric matrix, see the parameter `use_lower`.
`P`	The sampling probability. Should be between 0 and 1.
`use_lower`	If `TRUE/FALSE`, only the lower/upper triangular part of `A` is used for sampling and the following eigendecomposition steps.
`k`	Number of eigen values requested.
`tol`	Precision parameter of the iterative algorithm. Default is 1e-5.
`abs`	A logical variable indicating whether the `k` eigen values should be largest in absolute value. If `FALSE`, then `eigs_sym`) is used as the iterative algorithm. If `TRUE`, then `svds`) is used as the iterative algorithm.Default is `FALSE`.
`...`	Additional arguments of function `svds` or `eigs_sym`.

Details

This function computes the randomized eigenvalue decomposition of a data matrix (0-1 coded) using the random sampling scheme. The data matrix `A` is first sampled to obtain a sparsified matrix. An iterative algorithm (`svds` or `eigs_sym`) for computing the leading eigen vectors is then performed on the sparsified matrix to obtain the randomized eigen vectors and eigen values.

Value

vectors	The randomized k eigen vectors.
values	The k eigen values.
sparA	The sparsified data matrix obtained via rsample_sym(A,P)/P.

See Also

rsample_sym, svds, eigs_sym.

Examples

```
n <- 100
k <- 2
clustertrue <- rep(1:k, each = n/k)
A <- matrix(0, n, n)
for(i in 1:(n-1)) {
    for(j in (i+1):n) {
        A[i, j] <- ifelse(clustertrue[i] == clustertrue[j], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
        A[j, i] <- A[i, j]
    }
}
diag(A) <- 0
A <- as(A, "dgCMatrix")
reig.sam(A, P = 0.7, use_lower = TRUE, k = k)
```

---

rsample *Sample a sparse matrix*

Description

Sample a sparse matrix

Usage

```
rsample(A, P)
```

Arguments

A	A sparse matrix of type "dgCMatrix".
P	The probability that each edge is kept.

Value

A binary sparse matrix of type "dgCMatrix".

Examples

```
library(Matrix)
set.seed(123)
n = 20
A = matrix(rbinom(n^2, 1, 0.5), 20, 20)
diag(A) = 0
A = as(A, "dgCMatrix")
A
rsample(A, 0.5)
```

`rsample_sym` *Sample a symmetric sparse matrix*

Description

Sample a symmetric sparse matrix

Usage

```
rsample_sym(A, P, use_lower = TRUE)
```

Arguments

`A`	A sparse matrix of type "dgCMatrix". `A` does not need to be symmetric, see the parameter `use_lower`.
`P`	The probability that each edge is kept.
`use_lower`	If `TRUE`/`FALSE`, only the lower/upper triangular part of `A` is used for sampling.

Value

A lower triangular, binary, and sparse matrix of type "dgCMatrix". The diagonal elements are all zeros.

Examples

```
library(Matrix)
set.seed(123)
n = 20
A = matrix(rbinom(n^2, 1, 0.5), 20, 20)
A = as(A, "dgCMatrix")
A
rsample_sym(A, 0.5, use_lower = TRUE)
rsample_sym(A, 0.5, use_lower = FALSE)
```

`youtubeNetwork` *Youtube social network*

Description

This is a Youtube social network where users form friendship each other.

Usage

```
data(youtubeNetwork)
```

Format

The `youtubeNetwork` object is a sparse matrix representing the adjacency matrix of the Youtube social network.

Details

There is 1134890 nodes and 2987624 edges.

Source

http://snap.stanford.edu/data/com-Youtube.html

References

J. Yang and J. Leskovec. (2012) *Defining and Evaluating Network Communities based on Ground-truth*, *ICDM*

Examples

```
data(youtubeNetwork)
A <- youtubeNetwork
reig.sam(A, P=0.7, k = 4)
reig.pro(A, rank = 4)
```

Index