Package 'loe'

October 13, 2022

Type Package

Title Local Ordinal Embedding

Version 1.1

Date 2016-02-09

Author Yoshikazu Terada, Ulrike von Luxburg

Maintainer Yoshikazu Terada <terada@nict.go.jp>

Depends R (>= 3.2.3), MASS

Suggests igraph, Matrix, vegan

Description
Local Ordinal embedding (LOE) is one of graph embedding methods for unweighted graphs.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-02-09 14:36:35

R topics documented:

loe-package *Local Ordinal Embedding*

Description

Soft Ordinal Embedding (SOE) is a simple new method for ordinal embedding problem. SOE can deal with a part of ordinal comparisons: $dissimilarity(i, j) < dissimilarity(k, l)$ for certain quadruples of indices (i, j, k, l).

The local point of view suggests ordinal embedding as an interesting alternative to graph drawing algorithms. If vertex i is connected by an edge to vertex j, but not to vertex k, we interpret this constellation as a constraint of the form $\xi_{ij} < \xi_{ik}$. With this interpretation, graph embedding (graph drawing) becomes a special case of ordinal embedding. We call SOE for graph embedding problem Local Ordinal Embedding (LOE). For the connectivity matrix of given unweighted graph, LOE provides a low-dimensional embedding which preserves the original graph structure.

This package provides SOE and LOE.

Details

Package:	loe
Type:	Package
Version:	1.0
Date:	2014-01-28
License:	GPL-2

The functions for SOE on ordinal comparison matrices and LOE on unweighted adjacency matrices are `SOE()` and `LOE()`, respectively.

Acknowledgement

The authors wishes to express their thanks to Dr. Michio Yamamoto and Mr. Yuichi Saito for their help in creating this package. This work was partially supported by Grant-in-Aid for JSPS Fellows Number 24-2466.

Author(s)

Yoshikazu Terada and Ulrike von Luxburg

Maintainer: Yoshikazu Terada <terada@sigmath.es.osaka-u.ac.jp>

GARI *Graph Adjusted Rand Index*

Description

Computes the graph adjusted Rand index measuring a recovery rate of ordinal information in an unweighted graph.

Usage

```
GARI(ADM, EADM)
```

Arguments

ADM	The given unweighted adjacency matrix
EADM	An recovered unweighted adjacency matrix from an embedding. The size of EADM should be same as that of ADM.

Details

GARI is bounded from above by 1, and $\mathrm{GARI}(A_n, \hat{A}_n) = 1 \iff A_n = \hat{A}_n$. A high GARI score implies that many of the ordinal constraints have been satisfied by the solution.

Value

The graph adjusted rand index measuring a recovery rate of ordinal information (a scalar).

Author(s)

Yoshikazu Terada

Examples

```
library(igraph)
############
#Based on the distance matrix of an embedding,
#this function provides the recovered adjacency matrix for given number of nearest neighbors.
############
rec.graph <- function(DM, k, symm =FALSE, weight=FALSE ) {
N <- nrow(DM)
ADM <- matrix(0, N, N)
#Search kNN point
if(weight==TRUE){
 for (i in 1:N) {
  nid <- order(DM[i,])
  ADM[ i, nid[2:(k[i]+1)] ] <- DM[ i, nid[2:(k[i]+1)] ]
 }
}else{
 for (i in 1:N) {
  nid <- order(DM[i,])
  ADM[i,nid[2:(k[i]+1)] ] <- 1
 }
}
if(symm==TRUE){
 SADM <- ADM+t(ADM)
```

```
 SADM[SADM==2*ADM] <- ADM[SADM==2*ADM]
 ADM <- SADM
}
return(ADM)
}


ADM <- as.matrix( get.adjacency(graph.famous("Thomassen")) )
#Apply LOE
result.LOE <- LOE(ADM=ADM,p=2,c=0.1,method="BFGS",report=1000)

#Compute the vector of numbers of nearest neighbors with each verteces
true.nn <- apply(X=ADM,1,sum)

#Reconstracte the adjacency matrix based on the result embedding
EDM <- as.matrix( dist(result.LOE$X) )
EADM <- rec.graph(EDM,k=true.nn )

#Compute GARI between ADM and EADM
GARI(ADM,EADM)
```

`get.order` *Making ordinal information.*

Description

For a given dissimilarity matrix $\Xi = (\xi_{ij})_{n\times n}$, creating the ordinal information matrix whose row vectors are ordinal quadruples of indices: a row vector (i, j, k, l) means $\xi_{ij} < \xi_{kl}$.

Usage

```
get.order(DM)
```

Arguments

DM	A dissimilarity matrix.

Value

The matrix whose row vectors are ordinal quadruples of indices: a row vector (i,j,k,l) means $\xi_{ij} < \xi_{kl}$.

Author(s)

Yoshikazu Terada

Examples

```
data(eurodist)
OIF <- get.order(as.matrix(eurodist))
```

LOE *LOE for a unweighted Graph.*

Description

Performs LOE for a given unweighted grah.

Usage

```
LOE(ADM, p=2, c=0.1,eps= 1e-5,maxit =1000,method=c("BFGS","SD","MM"),
iniX = "auto",report=100,DEL=1,H=0.5)
```

Arguments

ADM	The adjacency matrix.
p	Number of dimensions.
c	Scale parameter which only takes strictly positive value.
eps	Convergence criterion for the majorization algorithm or the steepest descent algorithm.
maxit	Maximum number of iteretions.
method	If "BFGS", then the BFGS method is used for optimizing the stress function. If "SD", then the steepest descent method is used. If "MM", then the majorization minimization algortihm is used.
iniX	Matrix with starting values for embedding (optional). If "auto", then Laplacian eigenmaps is used as a starting value.
report	The frequency of reports. Defaults to every 100 iterations.
DEL	The initial step size in the steepest descent algorithm. Defaults to 1.
H	The rate parameter of the backtracking line search in the steepest descent algorithm. This parameter only takes value in $(0, 1)$. Defaults to 0.5.

Value

LOE returns a list with components:

X	The best corrdinate matrix with p columns whose rows give the coordinates of the vertexes.
str	If method is "BFGS", then the value of the stress function of LOE corresponding to X is returned. If "SD" or "MM", then the vector of values on each iteration is returned.

Author(s)

Yoshikazu Terada

Examples

```
#################
#Realizable case#
#################
library(igraph)
###############################
#Create a toy data
x <- seq(-5,5,by=1)
y <- seq(1,6,by=1)
hx1 <- seq(-3.5,-1.5,by=0.5)
hx2 <- seq(1.5,3.5,by=0.5)
hy <- seq(2.5,4.5,by=0.5)
D1 <- matrix(0,66,2)
for(i in 1:11){
for(j in 1:6){
D1[i+11*(j-1),] <- c(x[i],y[j])
}
}
D2n <- matrix(0,25,2)
D2p <- matrix(0,25,2)
for(i in 1:5){
for(j in 1:5){
D2n[i+5*(j-1),] <- c(hx1[i],hy[j])
D2p[i+5*(j-1),] <- c(hx2[i],hy[j])
}
}
D2n <- D2n[-c(7,9,17,19),]
D2p <- D2p[-c(7,9,17,19),]
Data <- rbind(D1,D2n,D2p)
Data <- scale(Data[order(Data[,1]),], scale=FALSE)
N <- nrow(Data)
#Visualization of the original data
plot(Data,pch=20,xlab="",ylab="",cex=1,col=rainbow(N,start=.7,end=.9),
xlim=c(-7,7),ylim=c(-7,7),main="Original data")

#Creating a k-NN graph based on Data
DM <- as.matrix(dist(Data))
ADM <- make.kNNG(DM,k=25)

#plot of the adjacency matrix
AADM <- ADM
diag(AADM) <- NA
image(AADM[N:1,],col=topo.colors(3),ann=FALSE,axes=FALSE)

#Apply some graph embedding methods
LE <-spec.emb(A=ADM,2,norm=FALSE)
result.LOE <- LOE(ADM=ADM,p=2,c=0.1,method="BFGS",report=1000,iniX=LE)

#Procrustes transform
library(vegan)
LOEX <- procrustes(X=Data,Y=result.LOE$X)$Yrot
plot(LOEX,pch=20,xlab="",ylab="",cex=1,col=rainbow(N,start=.7,end=.9),
```

```
xlim=c(-7,7),ylim=c(-7,7),main="LOE")
################################

#############
#This function provide appropriate vectors of xlim and ylim
#for given embedding matrix X.
#############
make.lim <- function(X){
mima <- matrix(0, 2,2)
mima[,1] <- apply(X, 2, min)
mima[,2] <- apply(X, 2, max)
han <- mima[,2] - mima[,1]
cent <-  (mima[,2] + mima[,1])/2
tmpr <- max(han)+max(han)*0.05
for(s in 1:2){
mima[s,] <- c(cent[s]-tmpr/2,cent[s]+tmpr/2)
}
return(mima)
}
##############################
#Standered graph-drawing task#
##############################
################################
ADM <- as.matrix( get.adjacency(graph.famous("Thomassen")) )
TG <- graph.adjacency(ADM)

#Apply some graph embedding methods
LE <-spec.emb(A=ADM,2,norm=FALSE)
KK <- layout.kamada.kawai(TG,maxiter=1000,start=LE)
FR <- layout.fruchterman.reingold(TG,maxiter=1000,start=LE)
result.LOE <- LOE(ADM=ADM,p=2,c=0.1,method="MM",report=1000,maxit=2000)


#Visualization of embeddings
par(mfrow=c(2,3),oma = c(0, 0, 4, 0))
#plot of the adjacency matrix
AADM <- ADM
N <- nrow(AADM)
diag(AADM) <- NA
image(AADM[N:1,],col=topo.colors(3),ann=FALSE,axes=FALSE)
#plot of Laplacian eigenmaps
tmplim <- make.lim(LE)
vsize <- (tmplim[1,2] -tmplim[1,1])*4
plot(TG, layout=LE, main="Laplacian eigenmaps",
vertex.size=vsize,vertex.color="blue",vertex.label.color="white",vertex.label=NA,
edge.arrow.size=0.1,xlim=tmplim[1,],ylim=tmplim[2,],axes=TRUE,rescale=FALSE)
#plot of Kamada and Kawai
tmplim <- make.lim(KK)
vsize <- (tmplim[1,2] -tmplim[1,1])*4
plot(TG, layout=KK, main="Kamada and Kawai",
vertex.size=vsize,vertex.color="blue",vertex.label.color="white",vertex.label=NA,
edge.arrow.size=0.1,xlim=tmplim[1,],ylim=tmplim[2,],axes=TRUE,rescale=FALSE)
#plot of Fruchterman Reingold
```

```
tmplim <- make.lim(FR)
vsize <- (tmplim[1,2] -tmplim[1,1])*4
plot(TG, layout=FR, main="Fruchterman Reingold",
vertex.size=vsize,vertex.color="blue",vertex.label.color="white",vertex.label=NA,
edge.arrow.size=0.1,xlim=tmplim[1,],ylim=tmplim[2,],axes=TRUE,rescale=FALSE)
#plot of LOE
tmplim <- make.lim(result.LOE$X)
vsize <- (tmplim[1,2] -tmplim[1,1])*4
plot(TG, layout=result.LOE$X, main="LOE",
vertex.size=vsize,vertex.color="blue",vertex.label.color="white",vertex.label=NA,
edge.arrow.size=0.1,xlim=tmplim[1,],ylim=tmplim[2,],axes=TRUE,rescale=FALSE)
plot(result.LOE$str,type="l",xlab="Number of iter.", ylab="Stress")
#Make title
mtext(side = 3, line=1, outer=TRUE, text = "Thomassen", cex=2)
################################
```

`make.distmat` *Euclidean distance matrix computation*

Description

This function computes and returns the distance matrix computed by using the Euclidean distance between the row of a data matrix.

Usage

```
make.distmat(X)
```

Arguments

`X`	A numeric matrix.

Value

The Euclidean distance matrix based on a given corrdinate matrix `X`.

Author(s)

Yoshikazu Terada

Examples

```
#Create a toy data
x <- seq(-5,5,by=1)
y <- seq(1,6,by=1)
hx1 <- seq(-3.5,-1.5,by=0.5)
hx2 <- seq(1.5,3.5,by=0.5)
hy <- seq(2.5,4.5,by=0.5)
D1 <- matrix(0,66,2)
for(i in 1:11){
```

```
for(j in 1:6){
D1[i+11*(j-1),] <- c(x[i],y[j])
}
}
D2n <- matrix(0,25,2)
D2p <- matrix(0,25,2)
for(i in 1:5){
for(j in 1:5){
D2n[i+5*(j-1),] <- c(hx1[i],hy[j])
D2p[i+5*(j-1),] <- c(hx2[i],hy[j])
}
}
D2n <- D2n[-c(7,9,17,19),]
D2p <- D2p[-c(7,9,17,19),]
Data <- rbind(D1,D2n,D2p)

#Creating a k-NN graph based on Data
DM <- make.distmat(Data)
```

---

make.kNNG *Making the kNN Graph*

Description

Making the kNN graph from given distance matrix.

Usage

```
make.kNNG(DM, k = as.integer(2 * log(nrow(DM))), symm = FALSE, weight = FALSE)
```

Arguments

DM	A distance matrix.
k	Number of neighbers
symm	If TRUE, then the connectivity matrix is symmetrized.
weight	If TRUE, then the weighted kNN graph is created.

Value

The adjacency matrix of a kNN graph.

Author(s)

Yoshikazu Terada

Examples

```
#Create a toy data
x <- seq(-5,5,by=1)
y <- seq(1,6,by=1)
hx1 <- seq(-3.5,-1.5,by=0.5)
hx2 <- seq(1.5,3.5,by=0.5)
hy <- seq(2.5,4.5,by=0.5)
D1 <- matrix(0,66,2)
for(i in 1:11){
for(j in 1:6){
D1[i+11*(j-1),] <- c(x[i],y[j])
}
}
D2n <- matrix(0,25,2)
D2p <- matrix(0,25,2)
for(i in 1:5){
for(j in 1:5){
D2n[i+5*(j-1),] <- c(hx1[i],hy[j])
D2p[i+5*(j-1),] <- c(hx2[i],hy[j])
}
}
D2n <- D2n[-c(7,9,17,19),]
D2p <- D2p[-c(7,9,17,19),]
Data <- rbind(D1,D2n,D2p)
Data <- scale(Data[order(Data[,1]),], scale=FALSE)

#Visualization of Data
plot(Data,pch=20,xlab="",ylab="",cex=1,col=rainbow(108,start=.7,end=.9),
xlim=c(-7,7),ylim=c(-7,7))

#Creating a k-NN graph based on Data
DM <- as.matrix(dist(Data))
ADM <- make.kNNG(DM,k=25)
```

SOE *Soft Ordinal Embedding (SOE)*

Description

Performs SOE for a given part of ordinal informaiton.

Usage

```
SOE(CM, N, p = 2, c = 0.1, maxit = 1000, report = 100, iniX = "rand",rnd=10000)
```

Arguments

CM	A matrix whose rows represent ordinal information: A row vector (i, j, k, l) means $\xi_{ij} < \xi_{kl}$, where ξ_{ij} is a dissimilarity between object i and object j.

	If the number of given ordinal comparisons is more than 100000, then SOE randomly chooses rnd ordinal comarisons from given comparisons.
N	The number of objects.
p	The number of dimensions.
c	Scale parameter which only takes strictly positive value.
maxit	Maximum number of iteretions.
report	The frequency of reports. Defaults to every 100 iterations.
iniX	Matrix with starting values for embedding (optional).
rnd	The number of ordinal comarisons which are randomly chosen from given ordinal comparisons CM. rnd is used only if the number of given ordinal comparisons is more than 100000.

Value

SOE returns a list with components:

X	The best corrdinate matrix with p columns whose rows give the coordinates of objects.
str	The value of the stress function of SOE corresponding to X.

Author(s)

Yoshikazu Terada

Examples

```
library(MASS)
data(eurodist)
OIF <- get.order(as.matrix(eurodist))
sid <- sample(1:nrow(OIF))
POI <- OIF[sid[1:1000],]
result <- SOE(CM=POI, N=nrow(as.matrix(eurodist)),p=2, c=0.1,maxit =1000,report=100)
plot(result$X,type="n",xlab="Dim. 1", ylab="Dim. 2")
text(result$X,rownames(as.matrix(eurodist)),cex=1)
```

---

spec.emb *Spectral embedding with the normalized or the unnormalized Laplacian (Laplacian eigenmaps).*

Description

Performs spectral embedding for a given adjacency matrix.

Usage

```
spec.emb(A, p, norm = TRUE)
```

Arguments

A	
p	The number of dimensions.
norm	If TRUE, then the normalized Laplacian is used for embedding. If FASLE, then the unnormalized Laplacian is used for embedding.

Value

The corrdinate matrix with p columns whose rows give the coordinates of the vertexes.

Author(s)

Yoshikazu Terada

Examples

```
library(igraph)
ADM <- as.matrix( get.adjacency(graph.famous("Icosahedral")) )

#Apply some graph embedding methods
LE <-spec.emb(A=ADM,2,norm=FALSE)
```

Index