Package 'NCutYX'

January 20, 2025

Type Package

Title Clustering of Omics Data of Multiple Types with a Multilayer Network Representation

Version 0.1.0

Author Sebastian J. Teran Hidalgo, Shuangge Ma, Ruofan Bie

Maintainer Sebastian J. Teran Hidalgo <sebastianteranhidalgo@gmail.com>

Description Omics data come in different forms: gene expression, methylation, copy number, protein measurements and more.
'NCutYX' allows clustering of variables, of samples, and both variables and samples (biclustering),
while incorporating the dependencies across multiple types of Omics data.
(SJ Teran Hidalgo et al (2017), <doi:10.1186/s12864-017-3990-1>).

License GPL-3

Depends R (>= 3.4)

Imports Rcpp (>= 0.12.2), glmnet (>= 2.0-5), MASS (>= 7.3-47), mvtnorm (>= 1.0-6), fields (>= 9.0)

LinkingTo Rcpp, RcppEigen

RoxygenNote 6.0.1

URL https://github.com/Seborinos/NCutYX

BugReports https://github.com/Seborinos/NCutYX/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-09 18:27:31 UTC

Contents

`ancut` *Cluster the Columns of Y into K Groups with the Help of External Features X.*

Description

This function will output K clusters of the columns of Y using the help of X.

Usage

```
ancut(Y, X, K = 2, B = 3000, L = 1000, alpha = 0.5, nlambdas = 100,
  sampling = "equal", ncv = 5, dist = "correlation", sigma = 0.1)
```

Arguments

`Y`	is a n x p matrix of p variables and n observations. The columns of Y will be clustered into K groups.
`X`	is a n x q matrix of q variables and n observations.
`K`	is the number of clusters.
`B`	is the number of iterations in the simulated annealing algorithm.
`L`	is the temperature coefficient in the simulated annealing algorithm.
`alpha`	is the coefficient of the elastic net penalty.
`nlambdas`	is the number of tuning parameters in the elastic net.
`sampling`	if 'equal' then the sampling probabilities is the same during the simulated annealing algorithm, if 'size' the probabilites are proportional the the sizes of the clusters in the current iterations.
`ncv`	is the number of cross-validations in the elastic net.

`dist` is the type of distance metric for the construction of the similarity matrix. Options are 'gaussian', 'euclidean' and 'correlation', the latter being the default.

`sigma` is the parameter for the gaussian kernel distance which is ignored if 'gaussian' is not chosen as distance measure.

Details

The algorithm minimizes a modified version of NCut through simulated annealing. The modified NCut uses in the numerator the similarity matrix of the original data `Y` and the denominator uses the similarity matrix of the prediction of `Y` using `X`. The clusters correspond to partitions that minimize this objective function. The external information of `X` is incorporated by using elastic net to predict `Y`.

Value

A list with the final value of the objective function, the clusters and the lambda penalty chosen through cross-validation.

A list with the following components:

loss a vector of length `N` which contains the loss at each iteration of the simulated annealing algorithm.

cluster a matrix representing the clustering result of dimension `p` times `K`, where `p` is the number of columns of `Y`.

lambda.min is the optimal lambda chosen through cross-validation for the elastic net for predicting `Y` with `Y`.

Author(s)

Sebastian Jose Teran Hidalgo and Shuangge Ma. Maintainer: Sebastian Jose Teran Hidalgo. sebastianteranhidalgo@gmail.com.

References

Hidalgo, Sebastian J. Teran, Mengyun Wu, and Shuangge Ma. Assisted clustering of gene expression data using ANCut. BMC genomics 18.1 (2017): 623.

Examples

```
#This sets up the initial parameters for the simulation.
library(MASS)#for mvrnorm
library(fields)
n=30 #Sample size
B=50 #Number of iterations in the simulated annealing algorithm.
L=10000 #Temperature coefficient.
p=50 #Number of columns of Y.
q=p #Number of columns of X.
h1=0.15
h2=0.25

S=matrix(0.2,q,q)
```

```
S[1:(q/2),(q/2+1):q]=0
S[(q/2+1):q,1:(q/2)]=0
S=S-diag(diag(S))+diag(q)

mu=rep(0,q)

W0=matrix(1,p,p)
W0[1:(p/2),1:(p/2)]=0
W0[(p/2+1):p,(p/2+1):p]=0
Denum=sum(W0)

B2=matrix(0,q,p)
for (i in 1:(p/2)){
   B2[1:(q/2),i]=runif(q/2,h1,h2)
   in1=sample.int(q/2,6)
   B2[-in1,i]=0
}

for (i in (p/2+1):p){
   B2[(q/2+1):q,i]=runif(q/2,h1,h2)
   in2=sample(seq(q/2+1,q),6)
   B2[-in2,i]=0
}

X=mvrnorm(n, mu, S)
Z=X%*%B2
Y=Z+matrix(rnorm(n*p,0,1),n,p)
#Our method
Res=ancut(Y=Y,X=X,B=B,L=L,alpha=0,ncv=3)
Cx=Res[[2]]
f11=matrix(Cx[,1],p,1)
f12=matrix(Cx[,2],p,1)

errorL=sum((f11%*%t(f11))*W0)/Denum+sum((f12%*%t(f12))*W0)/Denum
#This is the true error of the clustering solution.
errorL

par(mfrow=c(1,2))
#Below is a plot of the simulated annealing path.
plot(Res[[1]],type='l',ylab='')
#Cluster found by ANCut
image.plot(Cx)
```

---

awncut *Cluster the Rows of X into K Clusters Using the AWNCut Method.*

Description

Builds similarity matrices for the rows of X and the rows of an assisted dataset Z. Clusters them into K groups while conducting feature selection based on the AWNCut method.

Usage

```
awncut(X, Z, K, lambda, Tau, B = 500, L = 1000)
```

Arguments

X	is an n x p1 matrix of n observations and p1 variables.
Z	is an n x p2 matrix of n observations and p2 variables. Z is the assistant dataset.
K	is the number of clusters.
lambda	is a vector of tuning parameter lambda in the objective function.
Tau	is a vector of tuning parameters tau to be used in the objective function.
B	is the number of iterations in the simulated annealing algorithm.
L	is the temperature coefficient in the simulated annealing algorithm.

Details

The algorithm maximizes a sum of the weighed NCut measure for X and assisted dataset Z, with the addition of a correlation measure between the two datasets. Feature selection is implemented by using the average correlation of each feature as a criterion.

Value

A list with the following components:

lambda the value of tuning parameter lambda for the result

tau the value of tuning parameter tau for the result

Cs a matrix of the clustering result

ws a vector of the feature selection result

OP.value the value of the objective function

Author(s)

Ruofan Bie. Maintainer: Sebastian Jose Teran Hidalgo sebastianteranhidalgo@gmail.com.

References

Li, Yang; Bie, Ruofan; Teran Hidalgo, Sebastian; Qin, Yinchen; Wu, Mengyun; Ma, Shuangge. Assisted gene expression-based clustering with AWNCut. (Submitted.)

Examples

```
set.seed(123456)
#This sets up the initial parameters for the simulation.
lambda <- seq(2,6,1) #Tuning parameter lambda
Tau    <- seq(0.2,0.8,0.2) #Tuning parameter tau

n=30; n1=10; n2=10; n3=n-n1-n2 #Sample size
p1=10; p2=10; r1=8; r2=8; #Number of variables and noises in each dataset
```

```
K=3; #Number of clusters

mu=1; #Mean of the marginal distribution
u1=0.5; #Range of enties in the coefficient matrix

library(mvtnorm)
epsilon <- matrix(rnorm(n*(p1-r1),0,1), n, (p1-r1)) # Generation of random error

Sigma1 <- matrix(rep(0.8,(p1-r1)^2),(p1-r1),(p1-r1)) # Generation of the covariance matrix
diag(Sigma1) <- 1

# Generation of the original distribution of the three clusters
T1 <- matrix(rmvnorm(n1,mean=rep(-mu,(p1-r1)),sigma=Sigma1),n1,(p1-r1))
T2 <- matrix(rmvnorm(n2,mean=rep(0,(p1-r1)),sigma=Sigma1),n2,(p1-r1))
T3 <- matrix(rmvnorm(n3,mean=rep(mu,(p1-r1)),sigma=Sigma1),n3,(p1-r1))

X1 <- sign(T1)*(exp(abs(T1))) #Generation of signals in X
X2 <- sign(T2)*(exp(abs(T2)))
X3 <- sign(T3)*(exp(abs(T3)))
ep1 <- (matrix(rnorm(n*r1,0,1),n,r1)) #Generation of noises in X
X <- rbind(X1,X2,X3)

beta1 <- matrix(runif((p1-r1)*(p2-r2),-u1,u1),(p1-r1),(p2-r2)) #Generation of the coefficient matrix
Z     <- X%*%beta1+epsilon #Generation of signals in Z
ep2   <- (matrix(rnorm(n*r2,0.5,1),n,r2)) #Generation of noises in Z

X <- cbind(X,ep1)
Z <- cbind(Z,ep2)
#our method
Tune1         <- awncut.selection(X, Z, K, lambda, Tau, B = 20, L = 1000)
awncut.result <- awncut(X, Z, 3, Tune1$lam, Tune1$tau, B = 20, L = 1000)
ErrorRate(awncut.result[[1]]$Cs, n1, n2)
```

`awncut.selection` *This Function Outputs the Selection of Tuning Parameters for the AWNCut Method.*

Description

This Function Outputs the Selection of Tuning Parameters for the AWNCut Method.

Usage

```
awncut.selection(X, Z, K, lambda, Tau, B = 500, L = 1000)
```

Arguments

X	is an n x p1 matrix of n observations and p1 variables.
Z	is an n x p2 matrix of n observations and p2 variables. Z is the assistant dataset.

K	is the number of clusters.
lambda	is a vector of tuning parameter lambda in the objective function.
Tau	is a vector of tuning parameter tau in the objective function.
B	is the number of iterations in the simulated annealing algorithm.
L	is the temperature coefficient in the simulated annealing algorithm. #' @return A list with the following components:

num is the position of the max DBI

Table is the Table of the DBI for all possible combination of the parameters

lam is the best choice of tuning parameter lambda

tau is the best choice of tuning parameter lambda

DBI is the max DBI

References

Li, Yang; Bie, Ruofan; Teran Hidalgo, Sebastian; Qin, Yinchen; Wu, Mengyun; Ma, Shuangge. Assisted gene expression-based clustering with AWNCut. (Submitted.)

Examples

```
set.seed(123456)
#This sets up the initial parameters for the simulation.
lambda <- seq(2,6,1) #Tuning parameter lambda
Tau    <- seq(0.2,0.8,0.2) #Tuning parameter tau

n=30; n1=10; n2=10; n3=n-n1-n2 #Sample size
p1=10; p2=10; r1=8; r2=8; #Number of variables and noises in each dataset

K=3; #Number of clusters

mu=1; #Mean of the marginal distribution
u1=0.5; #Range of enties in the coefficient matrix

library(mvtnorm)
epsilon <- matrix(rnorm(n*(p1-r1),0,1), n, (p1-r1)) # Generation of random error

Sigma1 <- matrix(rep(0.8,(p1-r1)^2),(p1-r1),(p1-r1)) # Generation of the covariance matrix
diag(Sigma1) <- 1

# Generation of the original distribution of the three clusters
T1 <- matrix(rmvnorm(n1,mean=rep(-mu,(p1-r1)),sigma=Sigma1),n1,(p1-r1))
T2 <- matrix(rmvnorm(n2,mean=rep(0,(p1-r1)),sigma=Sigma1),n2,(p1-r1))
T3 <- matrix(rmvnorm(n3,mean=rep(mu,(p1-r1)),sigma=Sigma1),n3,(p1-r1))

X1 <- sign(T1)*(exp(abs(T1))) #Generation of signals in X
X2 <- sign(T2)*(exp(abs(T2)))
X3 <- sign(T3)*(exp(abs(T3)))
ep1 <- (matrix(rnorm(n*r1,0,1),n,r1)) #Generation of noises in X
X <- rbind(X1,X2,X3)
```

```
beta1 <- matrix(runif((p1-r1)*(p2-r2),-u1,u1),(p1-r1),(p2-r2)) #Generation of the coefficient matrix
Z     <- X%*%beta1+epsilon #Generation of signals in Z
ep2   <- (matrix(rnorm(n*r2,0.5,1),n,r2)) #Generation of noises in Z

X <- cbind(X,ep1)
Z <- cbind(Z,ep2)
#our method
Tune1         <- awncut.selection(X, Z, K, lambda, Tau, B = 20, L = 1000)
awncut.result <- awncut(X, Z, 3, Tune1$lam, Tune1$tau, B = 20, L = 1000)
ErrorRate(awncut.result[[1]]$Cs, n1, n2)
```

`brca.data.cna` *Data on copy number aberrations from breast cancer patients.*

Description

A dataset containing copy number aberrations measurements from TCGA.

Usage

```
brca.data.cna
```

Format

A data frame with 873 patients and 515 gene names.

Source

https://cancergenome.nih.gov/

`brca.data.ge` *Data on gene expression from breast cancer patients.*

Description

A dataset containing gene expression measurements from TCGA.

Usage

```
brca.data.ge
```

Format

A data frame with 873 patients and 334 gene names.

Source

https://cancergenome.nih.gov/

brca.data.rppa *Data on protein measurements from breast cancer patients.*

Description

A dataset containing protein measurements from TCGA.

Usage

```
brca.data.rppa
```

Format

A data frame with 873 patients and 164 gene names.

Source

https://cancergenome.nih.gov/

cesc.data.cna *Data on copy number aberrations from cervical cancer patients.*

Description

A dataset containing copy number aberrations from TCGA.

Usage

```
cesc.data.cna
```

Format

A data frame with 164 patients and 488 gene names.

Source

https://cancergenome.nih.gov/

`cesc.data.ge` *Data on gene expression from breast cancer patients.*

Description

A dataset containing gene expression measurements from TCGA.

Usage

```
cesc.data.ge
```

Format

A data frame with 164 patients and 325 gene names.

Source

https://cancergenome.nih.gov/

`cesc.data.rppa` *Data on protein measurements from cervical cancer patients.*

Description

A dataset containing protein measurements from TCGA.

Usage

```
cesc.data.rppa
```

Format

A data frame with 164 patients and 144 gene names.

Source

https://cancergenome.nih.gov/

ErrorRate *This Function Calculates the True Error Rate of a Clustering Result, Assuming that There are Three Clusters.*

Description

This Function Calculates the True Error Rate of a Clustering Result, Assuming that There are Three Clusters.

Usage

```
ErrorRate(X, n1, n2)
```

Arguments

X	is a clustering result in matrix format.
n1	is the size of the first cluster.
n2	is the size of the second cluster.

Value

err is the true error rate of a clustering result.

References

Li, Yang; Bie, Ruofan; Teran Hidalgo, Sebastian; Qin, Yinchen; Wu, Mengyun; Ma, Shuangge. Assisted gene expression-based clustering with AWNCut. (Submitted.)

Examples

```
set.seed(123456)
#This sets up the initial parameters for the simulation.
lambda <- seq(2,6,1) #Tuning parameter lambda
Tau    <- seq(0.2,0.8,0.2) #Tuning parameter tau

n=30; n1=10; n2=10; n3=n-n1-n2 #Sample size
p1=10; p2=10; r1=8; r2=8; #Number of variables and noises in each dataset

K=3; #Number of clusters

mu=1; #Mean of the marginal distribution
u1=0.5; #Range of enties in the coefficient matrix

library(mvtnorm)
epsilon <- matrix(rnorm(n*(p1-r1),0,1), n, (p1-r1)) # Generation of random error

Sigma1 <- matrix(rep(0.8,(p1-r1)^2),(p1-r1),(p1-r1)) # Generation of the covariance matrix
diag(Sigma1) <- 1
```

```
# Generation of the original distribution of the three clusters
T1 <- matrix(rmvnorm(n1,mean=rep(-mu,(p1-r1)),sigma=Sigma1),n1,(p1-r1))
T2 <- matrix(rmvnorm(n2,mean=rep(0,(p1-r1)),sigma=Sigma1),n2,(p1-r1))
T3 <- matrix(rmvnorm(n3,mean=rep(mu,(p1-r1)),sigma=Sigma1),n3,(p1-r1))

X1 <- sign(T1)*(exp(abs(T1))) #Generation of signals in X
X2 <- sign(T2)*(exp(abs(T2)))
X3 <- sign(T3)*(exp(abs(T3)))
ep1 <- (matrix(rnorm(n*r1,0,1),n,r1)) #Generation of noises in X
X <- rbind(X1,X2,X3)

beta1 <- matrix(runif((p1-r1)*(p2-r2),-u1,u1),(p1-r1),(p2-r2)) #Generation of the coefficient matrix
Z     <- X%*%beta1+epsilon #Generation of signals in Z
ep2   <- (matrix(rnorm(n*r2,0.5,1),n,r2)) #Generation of noises in Z

X <- cbind(X,ep1)
Z <- cbind(Z,ep2)
#our method
Tune1         <- awncut.selection(X, Z, K, lambda, Tau, B = 20, L = 1000)
awncut.result <- awncut(X, Z, 3, Tune1$lam, Tune1$tau, B = 20, L = 1000)
ErrorRate(awncut.result[[1]]$Cs, n1, n2)
```

`mlbncut` *The MLBNCut Clusters the Columns and the Rows Simultaneously of Data from 3 Different Sources.*

Description

It clusters the columns of Z,Y and X into K clusters and the samples into R clusters by representing each data type as one network layer. It represents the Z layer depending on Y, and the Y layer depending on X.

Usage

```
mlbncut(Z, Y, X, K = 2, R = 2, B = 30, N = 500, q0 = 0.25,
  scale = TRUE, dist = "gaussian", sigmas = 1, sigmac = 1)
```

Arguments

`Z`	is a n x q matrix of q variables and n observations.
`Y`	is a n x p matrix of p variables and n observations.
`X`	is a n x r matrix of r variables and n observations.
`K`	is the number of column clusters.
`R`	is the number of row clusters.
`B`	is the number of iterations.
`N`	is the number of samples per iterations.

q0	is the quantiles in the cross entropy method.
scale	equals TRUE if data Y is to be scaled with mean 0 and variance 1.
dist	is the type of distance measure use in the similarity matrix. Options are 'gaussian' and 'correlation', with 'gaussian' being the default.
sigmas	is the tuning parameter of the Gaussian kernel of the samples.
sigmac	is the tuning parameter of the Gaussian kernel of the variables.

Details

This function will output K clusters of columns of Z, Y and X and R clusters of the samples.

The algorithm minimizes the NCut through the cross entropy method. The clusters correspond to partitions that minimize this objective function.

Value

A list with the final value of the objective function and the clusters.

References

Sebastian J. Teran Hidalgo and Shuangge Ma. Multilayer Biclustering of Omics Data using MLB-NCut. (Work in progress.)

Examples

```
#This sets up the initial parameters for the simulation.
library(NCutYX)
library(MASS)
library(fields)

n   <- 50
p   <- 50
h   <- 0.15
rho <- 0.15
mu  <- 1

W0 <- matrix(1,p,p)
W0[1:(p/5),1:(p/5)] <- 0
W0[(p/5+1):(3*p/5),(p/5+1):(3*p/5)] <- 0
W0[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)] <- 0
W0[(4*p/5+1):p,(4*p/5+1):p]=0
W0=cbind(W0,W0,W0)
W0=rbind(W0,W0,W0)

W1 <- matrix(1,n,n)
W1[1:(n/2),1:(n/2)] <- 0
W1[(n/2+1):n,(n/2+1):n] <- 0

X <- matrix(0,n,p)
Y <- matrix(0,n,p)
Z <- matrix(0,n,p)
```

```
Sigma=matrix(0,p,p)
Sigma[1:(p/5),1:(p/5)] <- rho
Sigma[(p/5+1):(3*p/5),(p/5+1):(3*p/5)] <- rho
Sigma[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)] <- rho
Sigma[(4*p/5+1):p,(4*p/5+1):p] <- rho
Sigma <- Sigma - diag(diag(Sigma))
Sigma <- Sigma + diag(p)

X[1:(n/2),]   <- mvrnorm(n/2,rep(mu,p),Sigma)
X[(n/2+1):n,] <- mvrnorm(n/2,rep(-mu,p),Sigma)

B11 <- matrix(0,p,p)
B12 <- matrix(0,p,p)
B21 <- matrix(0,p,p)
B22 <- matrix(0,p,p)

B11[1:(p/5),1:(p/5)]                     <- runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.5)
B11[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]   <- runif((2*p/5)^2,h/2,h)*rbinom((2*p/5)^2,1,0.5)
B11[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)] <- runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.5)
B11[(4*p/5+1):p,(4*p/5+1):p]           <- runif((1*p/5)^2,h/2,h)*rbinom((1*p/5)^2,1,0.5)

B12[1:(p/5),1:(p/5)]                     <- runif((p/5)^2,-h,-h/2)*rbinom((p/5)^2,1,0.5)
B12[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]   <- runif((2*p/5)^2,-h,-h/2)*rbinom((2*p/5)^2,1,0.5)
B12[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)] <- runif((p/5)^2,-h,-h/2)*rbinom((p/5)^2,1,0.5)
B12[(4*p/5+1):p,(4*p/5+1):p]         <- runif((1*p/5)^2,-h,-h/2)*rbinom((1*p/5)^2,1,0.5)

B21[1:(p/5),1:(p/5)]                     <- runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.5)
B21[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]   <- runif((2*p/5)^2,h/2,h)*rbinom((2*p/5)^2,1,0.5)
B21[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)] <- runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.5)
B21[(4*p/5+1):p,(4*p/5+1):p]           <- runif((1*p/5)^2,h/2,h)*rbinom((1*p/5)^2,1,0.5)

B22[1:(p/5),1:(p/5)]                     <- runif((p/5)^2,-h,-h/2)*rbinom((p/5)^2,1,0.5)
B22[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]   <- runif((2*p/5)^2,-h,-h/2)*rbinom((2*p/5)^2,1,0.5)
B22[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)] <- runif((p/5)^2,-h,-h/2)*rbinom((p/5)^2,1,0.5)
B22[(4*p/5+1):p,(4*p/5+1):p]         <- runif((1*p/5)^2,-h,-h/2)*rbinom((1*p/5)^2,1,0.5)

Y[1:(n/2),]   <- X[1:(n/2),]%*%B11+matrix(rnorm((n/2)*p,0,0.25),n/2,p)
Y[(n/2+1):n,] <- X[(n/2+1):n,]%*%B12+matrix(rnorm((n/2)*p,0,0.25),n/2,p)

Z[1:(n/2),]   <- Y[1:(n/2),]%*%B21+matrix(rnorm((n/2)*p,0,0.25),n/2,p)
Z[(n/2+1):n,] <- Y[(n/2+1):n,]%*%B22+matrix(rnorm((n/2)*p,0,0.25),n/2,p)

trial <- mlbncut(Z,
                 Y,
                 X,
                 K=4,
                 R=2,
                 B=10,
                 N=50,
                 dist='correlation',
                 q0=0.15,
                 scale=TRUE,
```

```
                  sigmas=0.05,
                  sigmac=1)

plot(trial[[1]],type='l')
image.plot(trial[[2]])
image.plot(trial[[3]])

errorK <- sum((trial[[3]][,1]%*%t(trial[[3]][,1]) +
                  trial[[3]][,2]%*%t(trial[[3]][,2]) +
                  trial[[3]][,3]%*%t(trial[[3]][,3]) +
                  trial[[3]][,4]%*%t(trial[[3]][,4]))*W0)/(3*p)^2 +
            sum((trial[[2]][,1]%*%t(trial[[2]][,1]) +
                  trial[[2]][,2]%*%t(trial[[2]][,2]))*W1)/(n)^2
errorK
```

---

muncut *MuNCut Clusters the Columns of Data from 3 Different Sources.*

Description

It clusters the columns of Z,Y and X into K clusters by representing each data type as one network layer. It represents the Z layer depending on Y, and the Y layer depending on X. Elastic net can be used before the clustering procedure by using the predictions of Z and Y instead of the actual values to improve the cluster results. This function will output K clusters of columns of Z, Y and X.

Usage

```
muncut(Z, Y, X, K = 2, B = 3000, L = 1000, alpha = 0.5, ncv = 3,
  nlambdas = 100, scale = FALSE, model = FALSE, gamma = 0.5,
  sampling = "equal", dist = "gaussian", sigma = 0.1)
```

Arguments

Z	is a n x q matrix of q variables and n observations.
Y	is a n x p matrix of p variables and n observations.
X	is a n x r matrix of r variables and n observations.
K	is the number of column clusters.
B	is the number of iterations in the simulated annealing algorithm.
L	is the temperature coefficient in the simulated annealing algorithm.
alpha	is the tuning parameter in the elastic net penalty, only used when model=T.
ncv	is the number of cross-validations used to choose the tuning parameter lambda in the elastic net penalty, only used when model=T.
nlambdas	number of tuning parameters lambda used during cross-validation, only when model=T.
scale	when TRUE the Z, Y and X are scaled with mean 0 and standard deviation equal 1.

model	when TRUE the the relationship between Z and Y, and between Y and X are modeled with the elastic net. The predictions of Z, and Y from the models are used in the clustering algorithm.
gamma	is the tuning parameter of the clustering penalty. Larger values give more importance to within layer effects and less to across layer effects.
sampling	if 'equal' then the sampling distribution is discrete uniform over the number of clusters, if 'size' the probabilities are inversely proportional to the size of each cluster.
dist	is the type of distance measure use in the similarity matrix. Options are 'gaussian' and 'correlation', with 'gaussian' being the default.
sigma	is the bandwidth parameter when the dist metric chosen is gaussian.

Details

The algorithm minimizes a modified version of NCut through simulated annealing. The clusters correspond to partitions that minimize this objective function. The external information of X is incorporated by using ridge regression to predict Y.

References

Sebastian J. Teran Hidalgo and Shuangge Ma. Clustering Multilayer Omics Data using MuNCut. (Revise and resubmit.)

Examples

```
library(NCutYX)
library(MASS)
library(fields) #for image.plot

#parameters#
set.seed(777)
n=50
p=50
h=0.5
rho=0.5

W0=matrix(1,p,p)
W0[1:(p/5),1:(p/5)]=0
W0[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]=0
W0[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)]=0
W0[(4*p/5+1):p,(4*p/5+1):p]=0
W0=cbind(W0,W0,W0)
W0=rbind(W0,W0,W0)

Y=matrix(0,n,p)
Z=matrix(0,n,p)
Sigma=matrix(rho,p,p)
Sigma[1:(p/5),1:(p/5)]=2*rho
Sigma[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]=2*rho
Sigma[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)]=2*rho
```

```
Sigma=Sigma-diag(diag(Sigma))
Sigma=Sigma+diag(p)

X=mvrnorm(n,rep(0,p),Sigma)
B1=matrix(0,p,p)
B2=matrix(0,p,p)

B1[1:(p/5),1:(p/5)]=runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.2)
B1[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]=runif((2*p/5)^2,h/2,h)*rbinom((2*p/5)^2,1,0.2)
B1[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)]=runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.2)

B2[1:(p/5),1:(p/5)]=runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.2)
B2[(p/5+1):(3*p/5),(p/5+1):(3*p/5)]=runif((2*p/5)^2,h/2,h)*rbinom((2*p/5)^2,1,0.2)
B2[(3*p/5+1):(4*p/5),(3*p/5+1):(4*p/5)]=runif((p/5)^2,h/2,h)*rbinom((p/5)^2,1,0.2)

Y=X%*%B1+matrix(rnorm(n*p,0,0.5),n,p)
Y2=X%*%B1

Z=Y%*%B2+matrix(rnorm(n*p,0,0.5),n,p)
Z2=Y%*%B2

#Computing our method
clust <- muncut(Z,
                Y,
                X,
                K        = 4,
                B        = 10000,
                L        = 500,
                sampling = 'size',
                alpha    = 0.5,
                ncv      = 3,
                nlambdas = 20,
                sigma    = 10,
                scale    = TRUE,
                model    = FALSE,
                gamma    = 0.1)

A <- clust[[2]][,1]%*%t(clust[[2]][,1])+
     clust[[2]][,2]%*%t(clust[[2]][,2])+
     clust[[2]][,3]%*%t(clust[[2]][,3])+
     clust[[2]][,4]%*%t(clust[[2]][,4])

errorK=sum(A*W0)/(3*p)^2
errorK
plot(clust[[1]],type='l')
image.plot(A)
```

---

ncut *Cluster the Columns of Y into K Groups Using the NCut Graph Measure.*

Description

Builds a similarity matrix for the columns of Y and clusters them into K groups based on the NCut graph measure. Correlation, Euclidean and Gaussian distances can be used to construct the similarity matrix.

Usage

```
ncut(Y, K = 2, B = 30, N = 500, dist = "correlation", scale = TRUE,
  q = 0.1, sigma = 1)
```

Arguments

Y	is a n x p matrix of p variables and n observations. The p columns of Y will be clustered into K groups using NCut.
K	is the number of clusters.
B	is the number of iterations.
N	is the number of samples per iterations.
dist	is the type of distance metric for the construction of the similarity matrix. Options are 'gaussian', 'euclidean' and 'correlation', the latter being the default.
scale	equals TRUE if data Y is to be scaled with mean 0 and variance 1.
q	is the quantile used for the top results at each iterations.
sigma	is the bandwidth parameter when the dist metric chosen is 'gaussian' (default=0.1).

Details

The algorithm minimizes the NCut through the cross entropy method. The edges of the graph correspond to the entries of a similarity matrix constructed based on a correlation, euclidean or gaussian distance metric. The clusters correspond to partitions that minimize this NCut objective function.

Value

A list with the following components:

quantile a vector of length N which contains the quantiles q at each iteration of the optimization algorithm.

cluster a matrix representing the clustering result of dimension p times K, where p is the number of columns of Y.

ncut the NCut measure for the cluster result.

Author(s)

Sebastian Jose Teran Hidalgo. Maintainer: Sebastian Jose Teran Hidalgo sebastianteranhidalgo@gmail.com.

References

Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007): 395-416.

Kroese, D. P., Rubinstein, R. Y., Cohen, I., Porotsky, S., & Taimre, T. (2013). "Cross-entropy method." In Encyclopedia of Operations Research and Management Science (pp. 326-333). Springer US.

Examples

```
# This sets up the initial parameters for the simulation.
library(MASS)
n=100 # Sample size
B=30 # Number of iterations in the simulated annealing algorithm.
p=50 # Number of columns of Y.

S=matrix(0.2,p,p)
S[1:(p/2),(p/2+1):p]=0
S[(p/2+1):p,1:(p/2)]=0
S=S-diag(diag(S))+diag(p)
mu=rep(0,p)

W0=matrix(1,p,p)
W0[1:(p/2),1:(p/2)]=0
W0[(p/2+1):p,(p/2+1):p]=0
Denum=sum(W0)

Y=mvrnorm(n, mu, S)
# NCut
Res=ncut(Y,
K=2,
B=30,
N=1000,
dist='correlation',
scale=TRUE,
q=0.2,
sigma=0.1)
Cx=Res[[2]]
f11=matrix(Cx[,1],p,1)
f12=matrix(Cx[,2],p,1)

errorL=sum((f11%*%t(f11))*W0)/Denum+sum((f12%*%t(f12))*W0)/Denum
# This is the true error of the clustering solution.
errorL
```

---

pwncut *Cluster the Columns of X into K Clusters by Giving a Weighted Cluster Membership while shrinking Weights Towards Each Other.*

Description

This function will output K channels of variables.

Usage

```
pwncut(X, K = 2, B = 3000, L = 1000, scale = TRUE, lambda = 1,
  epsilon = 0, nstarts = 3, start = "default", dist = "gaussian",
  sigma = 0.1, beta = 1)
```

Arguments

X	is a n x p matrix of p variables and n observations.
K	is the number of clusters.
B	is the number of iterations in the simulated annealing algorithm.
L	is the temperature coefficient in the simulated annealing algorithm.
scale	equals TRUE if data X is to be scaled with mean 0 and variance 1.
lambda	the tuning parameter of the penalty. Larger values shrink the weighted cluster membership closer together (default = 1).
epsilon	values in the similarity matrix less than epsilon are set to 0 (default = 0).
nstarts	the number of starting values also corresponding how many times simulated annealing is run. Larger values provide better results but takes longer.
start	if it equals 'default' then the starting value for all weights is 1/K. If 'random' then weights are sampled from a uniform distribution and then scaled to sum 1 per variable.
dist	specifies the distance metric used for constructing the similarity matrix. Options are 'gaussian', 'correlation' and 'euclidean' (default = 'gaussian').
sigma	is the bandwidth parameter when the dist metric chosen is 'gaussian' (default = 0.1).
beta	when dist='correlation', beta is the exponent applied to each entry of the similarity matrix.

Details

The algorithm minimizes a modified version of NCut through simulated annealing. The clusters correspond to partitions that minimize this objective function.

References

Sebastian J. Teran Hidalgo, Mengyun Wu and Shuangge Ma. Penalized and weighted clustering of gene expression data using PWNCut. (Submitted.)

Examples

```
# This sets up the initial parameters for the simulation.
n <- 100 # Sample size
p <- 100 # Number of columns of Y.
K <- 3

C0            <- matrix(0,p,K)
C0[1:25,1]    <- matrix(1,25,1)
C0[26:75,1:3] <- matrix(1/3,50,3)
C0[76:100,3]  <- matrix(1,25,1)

A0 <- C0[ ,1]%*%t(C0[ ,1]) + C0[ ,2]%*%t(C0[ ,2]) +
      C0[ ,3]%*%t(C0[ ,3])
A0 <- A0 - diag(diag(A0)) + diag(p)

Z1 <- rnorm(n,0,2)
Z2 <- rnorm(n,0,2)
Z3 <- rnorm(n,0,2)

Y <- matrix(0,n,p)
Y[ ,1:25]   <-  matrix(rnorm(n*25, 0, 2), n, 25) + matrix(Z1, n, 25, byrow=FALSE)
Y[ ,26:75]  <-  matrix(rnorm(n*50, 0, 2), n, 50) + matrix(Z1, n, 50, byrow=FALSE) +
                matrix(Z2, n, 50, byrow=FALSE) + matrix(Z3, n, 50, byrow=FALSE)
Y[ ,76:100] <-  matrix(rnorm(n*25, 0, 2), n, 25) + matrix(Z3, n, 25, byrow=FALSE)

trial <- pwncut(Y,
                K       = 3,
                B       = 10000,
                L       = 1000,
                lambda  = 1.5,
                start   = 'default',
                scale   = TRUE,
                nstarts = 1,
                epsilon = 0,
                dist    = 'correlation',
                sigma   = 10)

A1 <- trial[[2]][ ,1]%*%t(trial[[2]][ ,1]) +
      trial[[2]][ ,2]%*%t(trial[[2]][ ,2]) +
      trial[[2]][ ,3]%*%t(trial[[2]][ ,3])

A1 <- A1 - diag(diag(A1)) + diag(p)

plot(trial[[1]], type='l')
errorL <- sum(abs(A0-A1))/p^2
errorL
```

Index