Package ‘M2SMJF’

November 23, 2020

Title Multi-Modal Similarity Matrix Joint Factorization
Version 1.0
Description A new method to implement clustering from multiple modality data of certain samples, the function M2SMJF() jointly factorizes multiple similarity matrices into a shared sub-matrix and several modality private sub-matrices, which is further used for clustering. Along with this method, we also provide function to calculate the similarity matrix and function to evaluate the best cluster number from the original data.

Imports dplyr, MASS, stats
Depends R (>= 3.4.0)
License GPL (>= 2)
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Xiaoyao Yin [aut, cre]
Maintainer Xiaoyao Yin <yinxy1992@sina.com>
Repository CRAN
Date/Publication 2020-11-23 08:40:06 UTC

R topics documented:

affinityMatrix ......................................................... 2
Cal_NMI ................................................................. 3
cost ................................................................. 3
dist2bin ............................................................... 4
dist2chi ............................................................... 5
dist2eu ............................................................... 6
initialization ......................................................... 6
initialize_WL ......................................................... 7
affinityMatrix

To calculate the similarity matrix

description

calculate the affinity matrix from the diff matrix with 20 neighbors

usage

affinityMatrix(Diff, K = 20, sigma = 0.5)

arguments

Diff A diff matrix
K The number of neighbors in consideration
sigma A parameter to determine the scale

value

W The similarity matrix

author(s)

Xiaoyao Yin

examples

data_list <- simu_data_gen()
Diff <- dist2eu(Standard_Normalization(data_list[[1]]), Standard_Normalization(data_list[[1]]))
simi <- affinityMatrix(Diff, 20, 0.5)
Cal_NMI

Description

calculate the normalized mutual information of two vectors x and y.

Usage

Cal_NMI(x, y)

Arguments

x A vector
y A vector as long as x

Value

A number between 0 and 1 indicating the normalized mutual information

Author(s)

Xiaoyao Yin

Examples

x <- c(0.1, 0.2, 0.3, 0.4)
y <- c(0.1, 0.2, 0.3, 0.4)
NMI <- Cal_NMI(x, y)

cost

Calculate the cost

Description

A function to calculate the cost of the objective function

Usage

cost(new_WL_list, init_list, lambda)

Arguments

new_WL_list A list of matrices factorized from the similarity matrices list WL
init_list A list containing the updated result in this iteration
lambda A parameter to set the relative weight of the group sparsity constraints
Value

A number indicating the total cost of the objective function

Author(s)

Xiaoyao Yin

Examples

```r
WL <- simu_data_gen()
WL[[1]] <- affinityMatrix(dist2eu(Standard_Normalization(WL[[1]]), Standard_Normalization(WL[[1]])))
WL[[2]] <- affinityMatrix(dist2eu(Standard_Normalization(WL[[2]]), Standard_Normalization(WL[[2]])))
new_WL_list <- initialize_WL(WL)
k <- 5
lambda <- 0.25
init_list <- initialization(new_WL_list, k)
update_L_list <- update_L(new_WL_list, init_list)
update_alpha_list <- update_alpha(new_WL_list, update_L_list, lambda)
init_list <- update_alpha_list
new_loss <- cost(new_WL_list, init_list, lambda)
```

---

**dist2bin**

*Calculate the agreement-based measurement*

Description

Calculate the agreement-based measurement of two any pair-wise samples $x_i$ and $x_j$ for binary variables

Usage

`dist2bin(X, C)`

Arguments

- `X` A sample-feature matrix with rows as samples and columns as features

- `C` The same as `X`

Value

A matrix whose elements at `(i, j)` is the agreement-based measurement of two any pair-wise samples $x_i$ and $x_j$

Author(s)

Xiaoyao Yin
**Examples**

```r
data_list <- simu_data_gen()
X <- data_list[[1]]
C <- X
Diff <- dist2bin(X,C)
```

---

**dist2chi**

*Calculate the chi-squared distance*

**Description**

Calculate the chi-squared distance of two any pair-wise samples \(x_i\) and \(x_j\) for discrete variables

**Usage**

```r
dist2chi(X, C)
```

**Arguments**

- **X**: A sample-feature matrix with rows as samples and columns as features
- **C**: The same as \(X\)

**Value**

A matrix whose elements at \((i,j)\) is the chi-squared distance of two any pair-wise samples \(x_i\) and \(x_j\)

**Author(s)**

Xiaoyao Yin

**Examples**

```r
data_list <- simu_data_gen()
X <- data_list[[1]]
C <- X
Diff <- dist2chi(X,C)
```
dist2eu  

*Calculate the Euclidean distance*

Description

Calculate the Euclidean distance of two any pair-wise samples $x_i$ and $x_j$ for continuous variables.

Usage

```
dist2eu(X, C)
```

Arguments

- **X**: A sample-feature matrix with rows as samples and columns as features.
- **C**: The same as X.

Value

A matrix whose elements at (i,j) is the Euclidean distance of two any pair-wise samples $x_i$ and $x_j$.

Author(s)

Xiaoyao Yin

Examples

```
data_list <- simu_data_gen()
X <- data_list[[1]]
C <- X
Diff <- dist2eu(X, C)
```

initialization  

*initialize the sub-matrix $C_i$ into $\alpha \cdot L_i$ by SVD*

Description

$Li$ takes the first k columns of matrix d in SVD, while $\alpha$ is the mean of all the u of SVD result in each modality.

Usage

```
initialization(WL, k)
```
**initialize_WL**

**Arguments**

- **WL**: A list of similarity matrices
- **k**: A parameter to specify the cluster number

**Value**

A list with N+2 elements, the former N as modality private sub-matrices, the Nth as the shared sub-matrix and the last one as 1

**Author(s)**

Xiaoyao Yin

**Examples**

```r
WL <- simu_data_gen()
new_WL_list <- initialize_WL(WL)
k <- 5
init_list <- initialization(new_WL_list,k)
```

---

**Description**

Initialize from the similarity matrix list

Factorize the each of the similarity matrix $S_i$ into $C_i^t(C_i)$ by SVD

**Usage**

`initialize_WL(WL)`

**Arguments**

- **WL**: A list of similarity matrices

**Value**

A list as long as `WL` with elements satisfying `res[[i]]`

**Author(s)**

Xiaoyao Yin

**Examples**

```r
WL <- simu_data_gen()
new_WL_list <- initialize_WL(WL)
```
M2SMJF

the main part for M2SMJF and clustering result

Description

jointly factorize multiple matrices into a shared sub-matrix and multiple private sub-matrices

Usage

M2SMJF(WL, lambda = 0.25, theta = 10^-4, k)

Arguments

WL: A list of similarity matrices
lambda: A parameter to set the relative weight of the group sparsity constraints
theta: A parameter to determine the convergence
k: A parameter to specify the cluster number

Value

A list containing the clustering result

sub_matrices: a list containing all the sub-matrices
cluster_res: the clustering result which is as long as the number of samples

Author(s)

Xiaoyao Yin

Examples

WL <- simu_data_gen()
res <- M2SMJF(WL, 0.25, 10^-4, 5)
new_modularity

Calculate the modularity

Description
A function to calculate the modularity for weighted graph

Usage
new_modularity(init_list, WL)

Arguments
init_list A list with N+2 elements, the former N as modality private sub-matrices, the
Nth as the shared sub-matrix and the last one as the current loss
WL A list of similarity matrices

Value
A single value indicating the modularity of current factorization and clustering

Author(s)
Xiaoyao Yin

Examples
WL <- simu_data_gen()
WL[[1]] <- affinityMatrix(dist2eu(StandardNormalization(WL[[1]]), StandardNormalization(WL[[1]])))
WL[[2]] <- affinityMatrix(dist2eu(StandardNormalization(WL[[2]]), StandardNormalization(WL[[2]])))
new_WL_list <- initialize_WL(WL)
init_list <- initialization(new_WL_list, 5)
res <- M2SMJF(WL, 0.25, 10^-4, 5)
init_list <- res[[1]]
modularity <- new_modularity(init_list, WL)

simu_data_gen
Generate simulated data

Description
A function to generate simulated data with two modularities and five clusters

Usage
simu_data_gen()
Value

A list with two elements, which are the sample-feature matrices from different modality

Author(s)

Xiaoyao Yin

Examples

data_list <- simu_data_gen()

Standard_Normalization

Normalize the input matrix by column

Description

Normalize each column of x to have mean 0 and standard deviation 1.

Usage

Standard_Normalization(x)

Arguments

x

A sample-feature matrix with rows as samples and columns as features

Value

A sample-feature matrix with rows as samples and columns as features, each column of the matrix have mean 0 and standard deviation 1

Author(s)

Xiaoyao Yin

Examples

data_list <- simu_data_gen()
x <- data_list[[1]]
data_matrix <- Standard_Normalization(x)
**update_alpha**  

*the function to update alpha*

---

**Description**

update the sub-matrix alpha to convergence to its local minimum gradually

**Usage**

```r
update_alpha(WL, update_L_list, lambda)
```

**Arguments**

- **WL**
  - A list of similarity matrices
- **update_L_list**
  - A list with N+2 elements, the former N as modality private sub-matrices, the Nth as the shared sub-matrix and the last one as the current loss
- **lambda**
  - A parameter to set the relative weight of the group sparsity constraints

**Value**

A list containing the updated result in this iteration

**Author(s)**

Xiaoyao Yin

**Examples**

```r
WL <- simu_data_gen()
WL[[1]] <- affinityMatrix(dist2eu(Standard_Normalization(WL[[1]]), Standard_Normalization(WL[[1]])))
WL[[2]] <- affinityMatrix(dist2eu(Standard_Normalization(WL[[2]]), Standard_Normalization(WL[[2]])))
new_WL_list <- initialize_WL(WL)
k <- 5
lambda <- 0.25
init_list <- initialization(new_WL_list,k)
update_L_list <- update_L(new_WL_list,init_list)
update_alpha_list <- update_alpha(WL,update_L_list,lambda)
```
update_L

the function to update \( L_i \), for \( i=1,2,...,N \)

Description
update the sub-matrix \( L_i \), for \( i=1,2,...,N \) to convergence to its local minimum gradually

Usage
update_L(WL, init_list)

Arguments

\( WL \)  
A list of similarity matrices

\( init_list \)  
A list with \( N+2 \) elements, the former \( N \) as modality private sub-matrices, the \( N \)th as the shared sub-matrix and the last one as 1

Value
A list containing the updated result in this iteration

Author(s)
Xiaoyao Yin

Examples
WL <- simu_data_gen()
WL[[1]] <- affinityMatrix(dist2eu(Standard_Normalization(WL[[1]]),Standard_Normalization(WL[[1]])))
WL[[2]] <- affinityMatrix(dist2eu(Standard_Normalization(WL[[2]]),Standard_Normalization(WL[[2]])))
new_WL_list <- initialize_WL(WL)
k <- 5
init_list <- initialization(new_WL_list,k)
update_L_list <- update_L(WL,init_list)
Index

affinityMatrix, 2
Cal_NMI, 3
cost, 3
dist2bin, 4
dist2chi, 5
dist2eu, 6
initialization, 6
initialize_WL, 7
M2SMJF, 8
new_modularity, 9
simu_data_gen, 9
Standard_Normalization, 10
update_alpha, 11
update_L, 12