

Package ‘RJcluster’

April 6, 2021

Title RJ Clustering Algorithm

Version 2.5.0

Author Rachael Shudde [aut, cre],
Shahina Rahman [aut],
Valen Johnson [aut]

Maintainer Rachael Shudde <rachael.shudde@gmail.com>

Description Clustering algorithm for high dimensional data. This algorithm is ideal for data where $N \ll P$. Assuming that P feature measurements on N objects are arranged in an $N \times P$ matrix X , this package provides clustering based on the left Gram matrix XX^T . When the P -dimensional feature vectors of objects are drawn independently from a K distinct mixture distribution, the N -dimensional rows of the modified Gram matrix XX^T/P converges almost surely to K distinct cluster means. This transformation/projection thus allows the clusters to be tighter with order of P . To simulate data, type ```help("simulate_HD_data")` and to learn how to use the clustering algorithm, type ```help("RJclust")`.

License GPL (≥ 2)

Encoding UTF-8

Imports Rcpp ($\geq 1.0.2$), matrixStats, infotheo, rlang, stats,
graphics, profvis, mclust, foreach

LinkingTo Rcpp, RcppArmadillo

Suggests testthat ($\geq 2.1.0$), knitr, rmarkdown

RoxygenNote 7.1.1

VignetteBuilder knitr

Depends R (≥ 2.10)

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-04-06 21:00:03 UTC

R topics documented:

RJcluster-package 2

| | |
|------------------------------|---|
| Mutual_Information | 2 |
| RJclust | 3 |
| simulate_HD_data | 5 |

| | |
|--------------|----------|
| Index | 7 |
|--------------|----------|

| | |
|-------------------|--------------------------------|
| RJcluster-package | <i>RJ Clustering Algorithm</i> |
|-------------------|--------------------------------|

Description

Clustering algorithm for high dimensional data. This algorithm is ideal for data where $N \ll P$. Assuming that P feature measurements on N objects are arranged in an $N \times P$ matrix X , this package provides clustering based on the left Gram matrix XX^T . When the P -dimensional feature vectors of objects are drawn independently from a K distinct mixture distribution, the N -dimensional rows of the modified Gram matrix XX^T/P converges almost surely to K distinct cluster means. This transformation/projection thus allows the clusters to be tighter with order of P . To simulate data, type "help('simulate_HD_data')" and to learn how to use the clustering algorithm, type "help('RJclust')".

Details

| | |
|----------|------------|
| Package: | RJcluster |
| Type: | Package |
| Version: | 2.5.0 |
| Date: | 03-31-2021 |
| License: | GPL>=2 |

Author(s)

Rachael Shudde [aut, cre], Shahina Rahman [aut], Valen Johnson [aut]
 Maintainer: Rachael Shudde <rachael.shudde@gmail.com>

| | |
|--------------------|---------------------------|
| Mutual_Information | <i>Mutual_Information</i> |
|--------------------|---------------------------|

Description

Calculates normalized mutual information and adjusted mutual information. The value for both will be a value between 0 and 1 that measures how close the classification between the two clusters is. A value closer to 1 means the labels are more similar across $v1$ and $v2$, and a value closer to 0 means the labels are not as similar.

Usage

```
Mutual_Information(v1, v2)
```

Arguments

v1 vector containing first classification labels
v2 vector containing second classification labels

Details

See these links for a more formal definition of [AMI](#) and [NMI](#).

Value

Returns mutual information:

| | |
|-----|-----------|
| nmi | NMI value |
| ami | AMI value |

Examples

```
cluster1 <- sample(1:5, size = 10, replace = TRUE)  
cluster2 <- sample(1:2, size = 10, replace = TRUE)  
Mutual_Information(cluster1, cluster2)
```

RJclust

RJclust

Description

This is a clustering algorithm for data where $p \ll n$. There are four different types of penalty methods that can be used, depending on the size of the data and the accuracy. The first is the default method: the BIC penalty. There is also the AIC penalty, and full covariance. The full covariance method takes longer, but may give a more accurate implementation. Finally, there is also the mclust implementation, but that is not recommended. For all methods, a `C_max` variable is needed that is an upper limit on the possible number of clusters.

Usage

```
RJclust(  
  data,  
  penalty = "bic",  
  C_max = 10,  
  criterion = "VVI",  
  n_bins = NULL,
```

```

    seed = 1,
    verbose = FALSE
  )

```

Arguments

| | |
|------------------------|--|
| <code>data</code> | Data input, must be in matrix form. Currently no support for missing values |
| <code>penalty</code> | A string of possible vectors. Options include: "bic", "aic", "full_covariance", "mclust" (default = "bic") |
| <code>C_max</code> | Maximum number of clusters to look for (default is 10) |
| <code>criterion</code> | Model of covariance structure (default = "VVI") |
| <code>n_bins</code> | Number of cuts if penalty = "scale" for the scaled RJ algorithm (default = \sqrt{p}) |
| <code>seed</code> | Seed (default = 1) |
| <code>verbose</code> | Should progress be printed? (default = FALSE) |

Details

All implementation except the mclust and full covariance method use C++ to increase runtime.

`model_names` controls the type of covariance structure. See [Mclust Documentation](#) for more information. Note criterion "kmeans" is the same as "EEI". It is not suggested to use "kmeans" if it is suspected the classes are imbalanced

Value

Returns RJ algorithm result for "aic", "bic" ("mclust" and "scale" will return an mclust object:

| | |
|----------------------|---|
| <code>K</code> | number of clusters found |
| <code>class</code> | Class labels |
| <code>penalty</code> | Penalty values at each iteration |
| <code>mean</code> | Mean matrix |
| <code>prob</code> | Probability values |
| <code>z</code> | Z values from mclust (NULL penalty = "full_covariance") |

Examples

```

X = simulate_HD_data()
X = X$X
clust = RJclust(X, penalty = "bic", C_max = 10)

```

simulate_HD_data *simulate_HD_data*

Description

This is simulation data to check performance of RJcluster. Data can be simulated for any n , P , and size of clusters. The data has two types of data: noisy data and signal data. The percent of the data that is noisy is controlled by the sparsity parameter. The noisy data has two parts: half of it is $N(0, 1)$ and half is $N(0, noise_variance)$. The signal data is divided in two as well, half of it is $N(\mu[1], signal_variance)$ and half $N(\mu[2], signal_variance)$.

Usage

```
simulate_HD_data(
  size_vector = c(20, 20, 20, 20),
  p = 220,
  mu = matrix(c(1.5, 2.5, 0, 1.5, 0, -1.5, -2.5, -1.5), ncol = 2, byrow = TRUE),
  signal_variance = 1,
  noise_variance = 1,
  sparsity = 0.09,
  seed = 1234
)
```

Arguments

| | |
|-----------------|--|
| size_vector | A list of the size of the different clusters. (default = a balanced case of 4 clusters of size 20, c(20, 20, 20, 20)) |
| p | The number of columns in the simulated matrix (default = 220) |
| mu | The matrix of means, of dimension length(size_vector)x2. The first column of means is for the first half informative features, the second columns of mean is for the second half of the informative features (default is described in RJcluster paper) |
| signal_variance | Variance of the signal part of the generated data. A value of 1 indicates a high SNR, a value of 2 indicates a low SNR (default = 1) |
| noise_variance | Variance of the noisy part of the generated data (Default = 1) |
| sparsity | What percent of the data should be informative? A value between 0 and 1, a higher value means more data is informative (default = 0.09) |
| seed | Random seed. Change if generating multiple simulation datasets (default = 1234) |

Details

The data in the paper is generated with number of clusters = 4, a balanced case of c(20, 20, 20, 20) and an unbalanced case of c(20, 20, 200, 200), with $p = 220$ in both cases. The default is a balanced, high signal case with μ as the matrix in the RJcluster paper.

Value

Returns simulation data for X and Y values

X Matrix of dimension $\text{sum}(\text{size_vector}) \times p$

Y Vector of class labels of length $\sum(\text{size_vector})$, with unique values of $1:\text{length}(\text{size_vector})$

Examples

```
data = simulate_HD_data()
X = data$X
Y = data$Y
print(head(X))
```

Index

Mutual_Information, [2](#)

RJclust, [3](#)

RJcluster (RJcluster-package), [2](#)

RJcluster-package, [2](#)

simulate_HD_data, [5](#)