Package ‘uHMM’

October 12, 2022

Type Package
Title Construct an Unsupervised Hidden Markov Model
Version 1.0
Date 2016-03-18
Author Emilie POISSON-CAILLAULT [aut],
Paul TERNYNCK [aut, cre]
Maintainer Paul TERNYNCK <ternynck@lisic.univ-littoral.fr>
Description Construct a Hidden Markov Model with states learnt by unsupervised classification.
License GPL (>= 2)
RoxygenNote 5.0.1
LazyData TRUE
Depends R (>= 3.0.0), stats, grDevices
Imports tcltk, tcltk2, tkrplot, HMM, clValid, class, cluster,
FactoMineR, corrplot, chron
NeedsCompilation no
Repository CRAN
Date/Publication 2016-04-22 16:25:47

R topics documented:

uHMM-package .................................................... 2
computeGap ....................................................... 2
cutCalculation ................................................... 4
emissionMatrix .................................................... 5
FastSpectralNJW .................................................... 5
HMMparams ......................................................... 7
KmeansAutoElbow .................................................. 8
KpartitionNJW ....................................................... 9
MarelCarnot ....................................................... 10
selfKNN .......................................................... 11
spectralPamClusteringNg .......................................... 12
computeGap

Description

Find the highest gap between eigenvalues of a similarity matrix. The 2 first eigenvalues are considered as equal to each other (the gap between the 2 first eigenvalues is set to 0).
computeGap

Usage

computeGap(similarity, Gmax)

Arguments

similarity a similarity matrix.

Gmax the maximum gap value allowed (only the first Gmax eigenvalues will be taken into account).

Value

The function returns a list containing the following components:

gap a vector indicating the gap between similarity matrix eigenvalues (the gap between the 2 first eigenvalues is set to 0)

Kmax an integer indicating the index of the highest gap (the highest gap is between the Kmax-th and the (Kmax+1)-th eigenvalues)

Examples

```r
x <- cbind(matrix(rnorm(50, mean = 0, sd = 0.3), ncol = 2),
           matrix(rnorm(50, mean = 2, sd = 0.3), ncol = 2),
           matrix(rnorm(50, mean = 4, sd = 0.3), ncol = 2))
similarity<-ZPGaussianSimilarity(x,7)
Gap<-computeGap(similarity,10)
plot(1:length(Gap$gap),Gap$gap,type="h",
     main=paste("Gap criteria =",Gap$K),ylab="gap value",xlab="eigenvalues")
```

```r
x=(runif(1000)*4)-2;y=(runif(1000)*4)-2
keep<-which((x**2+y**2<0.5)|(x**2+y**2>1.5**2 & x**2+y**2<2**2 ))
data<-data.frame(x,y)[keep,]
plot(data)
similarity<-ZPGaussianSimilarity(data,1)
Gap<-computeGap(similarity,10)
plot(1:length(Gap$gap),Gap$gap,type="h",
     main=paste("Gap criteria =",Gap$K),ylab="gap value",xlab="eigenvalues")
```
cutCalculation function

Description

Compute intra and inter-cluster cuts from the similarity matrix of a dataset.

Usage

cutCalculation(similarity, label, K)

Arguments

- similarity: a similarity matrix.
- label: vector of cluster sequencing.
- K: number of clusters. (= nbCluster CALCULE DANS LA FONCTION ???)

Details

intra cluster cut :

\[ \text{Cut}(g_k, g_l) = \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w(x(i), x(j)) \]

Value

- mncut: the inter-cluster cut, i.e. K-sum(ratioCutVol).
- ratioCutVol: vector of intra-cluster cuts, one component per cluster.

Examples

```r
x<-rbinding(matrix(runif(100),ncol=2),matrix(runif(100)+2,ncol=2),matrix(runif(20)*3,ncol=2))
similarity<-ZPGaussianSimilarity(x,7)%*%t(ZPGaussianSimilarity(x,7))
km<-kmeans(similarity,2)
label<-km$cluster
plot(x,col=km$cluster)
cutCalculation(similarity,label,length(unique(label)))
```
emissionMatrix

### Description

This function estimates the emission matrix of a Hidden Markov Model from vectors of state and symbol sequencing.

### Usage

```r
emissionMatrix(states, symbols)
```

### Arguments

- `states` a numeric vector of state sequencing.
- `symbols` a numeric vector of symbol sequencing.

### Value

Estimated emission matrix.

### See Also

HMMparams

### Examples

```r
states<-c(1,1,3,2,1,2,1,3)
symbols<-c(4,1,3,1,4,4,4,2)
B<-emissionMatrix(states,symbols)
B
```

FastSpectralNJW

### Description

Perform the Jordan spectral algorithm for large databases. Data are sampled, using K-means with Elbow criteria, before being classified.

### Usage

```r
FastSpectralNJW(data, nK = NULL, Kech = 2000, StopCriteriaElbow = 0.97, neighbours = 7, method = "", nb.iter = 10, uHMMinterface = FALSE, console = NULL, tm = NULL)
```
Arguments

- **data** numeric matrix or dataframe.
- **nK** number of clusters desired. If NULL, optimal number of clusters will be computed using gap criteria.
- **Kech** maximum number of representative points in sampled data.
- **StopCriteriaElbow** maximum (minimum ?) de variance expliquees des points representatifs souhaite.
- **neighbours** number of neighbours considered for the computation of local scale parameters.
- **method** string specifying the spectral classification method desired, either "PAM" (for spectral kmedoids) or "" (for "spectral kmeans").
- **nb.iter** number of iterations.
- **uHMMinterface** logical indicating whether the function is used via the uHMMinterface.
- **console** frame of the uHMM interface in which messages should be displayed (only if uHMMinterface=TRUE).
- **tm** a one row dataframe containing text to display in the uHMMinterface (only if uHMMinterface=TRUE).

Details

Algorithme de Jordan pour un grand jeu de donnees : echantillonage puis spectral

Value

The function returns a list containing:

- **sim** similarity matrix of representative points, multiplied by its transpose (ZPGaussianSimilarity).
- **label** vector of cluster sequencing.
- **gap** number of clusters.
- **labelElbow** vector of prototype sequencing.
- **vpK** matrix containing, in columns, the K first normalised eigen vectors of the data similarity matrix.
- **valp** vector containing the K first eigen values of the data similarity matrix.
- **echantillons** matrix of prototypes coordinates.
- **label.echantillons** vector containing the cluster of each prototype.
- **numSymbole** vector containing the nearest prototype of each data item.

See Also

KmeansAutoElbow ZPGaussianSimilarity knn silhouette dunn connectivity dist
**HMMparams**  

*Hidden Markov Model parameter estimation*

**Description**

This function is used by the *uHMMinterface* to estimate parameters of a Hidden Markov Model.

**Usage**

```r
HMMparams(stateSeq, symbolSeq)
```

**Arguments**

- `stateSeq` a numeric vector of state sequencing.
- `symbolSeq` a numeric vector of symbol sequencing.

**Value**

HMMparams returns a list containing:

- `trans` The transition matrix.
- `emis` The emission matrix.
- `startProb` The vector of initial probability distribution (initial states are supposed equiprobable).

**See Also**

*transitionMatrix emissionMatrix*
KmeansAutoElbow

**Description**

KmeansAutoElbow performs k-means clustering on a dataframe with selection of optimal number of clusters using elbow criteria.

**Usage**

KmeansAutoElbow(features, Kmax, StopCriteria = 0.99, graph = FALSE)

**Arguments**

- **features**: dataframe or matrix of raw data.
- **Kmax**: maximum number of clusters allowed.
- **StopCriteria**: elbow method cumulative explained variance > criteria to stop K-search.
- **graph**: boolean, if TRUE figures are plotted.

**Details**

KmeansAutoElbow returns partition and K number of groups according to kmeans clustering and Elbow method.

**Value**

The function returns a list containing the following components:

- **K**: number of clusters in data according to explained variance and kmeans algorithm.
- **res.kmeans**: an object of class "kmeans" (see kmeans) containing classification results.

**See Also**

kmeans

**Examples**

```r
x <- rbind(matrix(rnorm(300, mean = 0, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
km <- KmeansAutoElbow(x, round(dim(x)/25,0)[1], StopCriteria = 0.99, graph = TRUE)
plot(x, col = km$res.kmeans$cluster)
points(km$res.kmeans$centers, col = 1:km$K, pch = 16)
```
**KpartitionNJW**

---

**KpartitionNJW function**

**Description**

Perform spectral classification on the similarity matrix of a dataset (Ng et al. (2001) algorithm), using kmeans algorithm on data projected in the space of its $K$ first eigen vectors.

**Usage**

\[ \text{KpartitionNJW}(\text{similarity}, K) \]

**Arguments**

- **similarity**: matrix of similarity.
- **$K$**: number of clusters.

**Value**

The function returns a list containing:

- **label**: vector of cluster sequencing.
- **centres**: matrix of cluster centers in the space of the $K$ first normalised eigen vectors.
- **vecteursPropresProjK**: matrix containing, in columns, the $K$ first normalised eigen vectors of the similarity matrix.
- **valeursPropresK**: vector containing the $K$ first eigen values of the similarity matrix.
- **vecteursPropres**: matrix containing, in columns, eigen vectors of the similarity matrix.
- **valeursPropres**: vector containing eigen values of the similarity matrix.
- **inertieZ**: vector of within-cluster sum of squares, one component per cluster.

**References**


**Examples**

```r
####
x <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
similarity <- ZPGaussianSimilarity(x, 7)
```
```r
similarity=similarity%*%t(similarity)
sp<-KpartitionNJW(similarity,3)
plot(x,col=sp$label)

x<-rbind(data.frame(x=1:100+(runif(100)-0.5)*2,y=runif(100)/5),
          data.frame(x=1:100+(runif(100)-0.5)*2,y=runif(100)/5+1),
          data.frame(x=1:100+(runif(100)-0.5)*2,y=runif(100)/5+2))

similarity<-ZPGaussianSimilarity(x,7)
similarity=similarity%*%t(similarity)
sp<-KpartitionNJW(similarity,3)
plot(x,col=sp$label)

x=(runif(1000)*4)-2;y=(runif(1000)*4)-2
keep<-which((x**2+y**2<0.5)|(x**2+y**2>1.5**2 & x**2+y**2<2**2 ))
data<-data.frame(x,y)[keep,]
similarity=ZPGaussianSimilarity(data, 7)
similarity=similarity%*%t(similarity)
sp<-KpartitionNJW(similarity,2)
plot(data,col=sp$label)
```

---

**MarelCarnot**

**MarelCarnot dataset**

**Description**

The MarelCarnot data set gives the measurements of 14 physico-chemical and biological parameters performed by the Marel-Carnot station (Boulogne-sur-Mer, France), at high frequency resolution.

**Usage**

MarelCarnot

**Format**

A data frame with 131487 rows and 16 columns.

**Details**

<table>
<thead>
<tr>
<th></th>
<th>description</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dates</td>
<td>date of measurement</td>
<td>(YYYY:MM:DD)</td>
</tr>
<tr>
<td>Hours</td>
<td>time of measurement</td>
<td>(HH:MM:SS)</td>
</tr>
<tr>
<td>C_NI1</td>
<td>nitrate concentration</td>
<td>(µmol/L)</td>
</tr>
<tr>
<td>C_PO1</td>
<td>phosphate concentration</td>
<td>(µmol/L)</td>
</tr>
<tr>
<td>C_O21</td>
<td>corrected dissolved oxygen</td>
<td>(mg/L)</td>
</tr>
</tbody>
</table>
### Description

This function performs the k-Nearest Neighbour algorithm without class estimation, but only computation of distances and neighbours.

### Usage

```r
selfKNN(train, K = 1)
```

### Arguments

- `train`: numeric matrix or data frame.
- `K`: number of neighbours considered.

### Value

The function returns a list with the following components:

- `D`: matrix of squared root of the distances between observations and their nearest neighbours.
- `idx`: Index of K nearest neighbours of each observation.
Examples

x<-matrix(runif(10),ncol=2)
plot(x,pch=c("1","2","3","4","5"))
selfKNN(x,K=4)

description

Perform spectral classification on the similarity matrix of a dataset, using pam algorithm (a more robust version of K-means) on projected data.

Usage

spectralPamClusteringNg(similarity, K)

Arguments

similarity    matrix of similarity
K             number of clusters

Value

The function returns a list containing:

label          vector of cluster sequencing.
centres        matrix of cluster medoids (similar in concept to means, but medoids are members of the dataset) in the space of the K first normalised eigen vectors.
id.med          integer vector of indices giving the medoid observation numbers.
vecteursPropresProjK    matrix containing, in columns, the K first normalised eigen vectors of the similarity matrix.
valeursPropresK       vector containing the K first eigen values of the similarity matrix.
vecteursPropres       matrix containing, in columns, eigen vectors of the similarity matrix.
valeursPropres       vector containing eigen values of the similarity matrix.
cluster.info  matrix, each row gives numerical information for one cluster. These are the cardinality of the cluster (number of observations), the maximal and average dissimilarity between the observations in the cluster and the cluster's medoid, the diameter of the cluster (maximal dissimilarity between two observations of the cluster), and the separation of the cluster (minimal dissimilarity between an observation of the cluster and an observation of another cluster).
transitionMatrix

References

See Also
pam

---

transitionMatrix  
Transition matrix estimation

Description
This function estimates the transition matrix of a (Hidden) Markov Model from a vector of state sequencing.

Usage
transitionMatrix(states)

Arguments
states  
a numeric vector of state sequencing.

Value
Estimated transition matrix.

See Also
HMMparams

Examples
states<-c(1,1,3,2,1,2,1,3)  
A<-transitionMatrix(states)  
A
uHMMinterface

*Graphical Interface to Build an uHMM*

**Description**

A user-friendly interface to detect usual or extreme events in a dataset and to characterize their dynamic, by building an unsupervised Hidden Markov Model.

**Usage**

```r
uHMMinterface(uHMMenv = NULL)
```

**Arguments**

- `uHMMenv`: an environment in which data and results will be stored. If NULL, a local environment will be created.

**Value**

Results are saved in the directory chosen by the user.

**References**


---

ZPGaussianSimilarity

*Similarity matrix with local scale parameter*

**Description**

Compute and return the similarity matrix of a data frame using gaussian kernel with a local scale parameter for each data point, rather than a unique scale parameter.

**Usage**

```r
ZPGaussianSimilarity(data, K)
```

**Arguments**

- `data`: a matrix or numeric data frame.
- `K`: number of neighbours considered to compute scale parameters.

**Value**

The matrix of similarity.
References


Examples

```r
x <- rbind(matrix(rnorm(50, mean = 0, sd = 0.3), ncol = 2))
similarity <- ZPGaussianSimilarity(x, 7)
```
Index

computeGap, 2
connectivity, 6
cutCalculation, 4
dist, 6
dunn, 6
emissionMatrix, 5, 7
FastSpectralNJW, 5
HMMparams, 5, 7, 13
kmeans, 8
KmeansAutoElbow, 6, 8
knn, 6
KpartitionNJW, 9
MarelCarnot, 10
pam, 13
selfKNN, 11
silhouette, 6
spectralPamClusteringNg, 12
transitionMatrix, 7, 13
uHMM-package, 2
uHMMinterface, 2, 7, 14
ZPGaussianSimilarity, 6, 14