Package 'CEC'

July 27, 2018

Title Cross-Entropy Clustering

Version 0.10.2

Date 2018-07-26

Author Konrad Kamieniecki [aut, cre], Przemyslaw Spurek [ctb]

Maintainer Konrad Kamieniecki <konrad.kamieniecki@alumni.uj.edu.pl>

Description CEC divides data into Gaussian type clusters. The implementation allows the simultaneous use of various type Gaussian mixture models, performs the reduction of unnecessary clusters and it's able to discover new groups. Based on Spurek, P. and Tabor, J. (2014) <doi:10.1016/j.patcog.2014.03.006>.

ByteCompile true

URL https://github.com/azureblue/cec

Encoding UTF-8

NeedsCompilation yes

SystemRequirements C++11

License GPL-3

Imports graphics, methods, stats, utils

Repository CRAN

Date/Publication 2018-07-26 22:00:06 UTC

R topics documented:

`CEC-package` *Cross-Entropy Clustering*

Description

CEC divides data into Gaussian type clusters. The implementation allows the simultaneous use of various type Gaussian mixture models, performs the reduction of unnecessary clusters and it's able to discover new groups. Based on Spurek, P. and Tabor, J. (2014) <doi:10.1016/j.patcog.2014.03.006> `cec`.

Details

Package:	CEC
Type:	Package
Version:	0.10.2
Date:	2018-07-26
License:	GPL-3

Author(s)

Konrad Kamieniecki

See Also

`cec`

`ball` *Ball*

Description

Generates points that form a ball with uniform density.

Usage

```
ball(n, r, dim)
```

Arguments

n	Number of points to generate.
r	Radius of the ball.
dim	Dimension of the points.

Value

Matrix of points with n rows and dim cols.

See Also

mouseset

Examples

```
M = ball(4000, 0.9)
plot(M, cex = 0.5, pch = 19)
```

---

cec *Cross-Entropy Clustering*

Description

Performs Cross-Entropy Clustering on a data matrix.

Usage

```
cec(x, centers, type = c("covariance", "fixedr", "spherical", "diagonal",
"eigenvalues", "mean", "all"), iter.max = 25, nstart = 1, param,
centers.init = c("kmeans++", "random"), card.min = "5%", keep.removed = F,
interactive = F, threads = 1, split = F, split.depth = 8, split.tries = 5,
split.limit = 100, split.initial.starts = 1,readline = T)
```

Arguments

x	Numeric matrix of data.
centers	Either a matrix of initial centers or the number of initial centers (k, single number cec(data, 4, ...) or a vector for variable number of centers cec(data, 3:10, ...)). If centers is a vector, length(centers) clusterings will be performed for each start (nstart argument) and the total number of clusterings will be length(centers) * nstart. If centers is a number or a vector, initial centers will be generated using a method depending on the centers.init argument.
type	Type (or types) of clustering (density family). This can be either a single value or a vector of length equal to the number of centers. Possible values are: "covariance", "fixedr", "spherical", "diagonal", "eigenvalues", "all" (default). Currently, if the centers argument is a vector, only signle type can be used.

`iter.max` Maximum number of iterations at each clustering.

`nstart` The number of clusterings to perform (with different initial centers). Only the best clustering (with the lowest cost) will be returned. Value grater then one is valid only if the `centers` argument is a number or a vector.

If the `centers` argument is a vector, `length(centers)` clusterings will be performed for each start and the total number of clusterings will be `length(centers) * nstart`.

If the split mode is on (`split = T`), it's rarely desired use change this parameter as the whole procedure (initial clustering + split) will be performed `nstart` times.

`centers.init` Centers initialization method. Possible values are: "kmeans++" (default), "random".

`param` Parameter (or parameters) specific to a particular type of clustering. Not all types of clustering require parameter. Types that require parameter: "covariance" (matrix parameter), "fixedr" (numeric parameter), "eigenvalues" (vector parameter). This can be a vector or a list (when one of the parameters is a matrix or a vector).

`card.min` Minimal cluster cardinality. If cluster cardinality becomes less than card.min, cluster is removed. This argument can be either an integer number or a string ended with a percent sign (e.g. "5%").

`keep.removed` If this parameter is TRUE, removed clusters will be visible in the results as NA in centers matrix (as well as corresponding values in the list of covariances).

`interactive` Interactive mode. If TRUE, the result of clustering will be plotted after every iteration.

`threads` Specifies the number of threads to use or "auto" to use default number of threads (usually the number of available processing units/cores) when performing multiple starts (`nstart` parameter).

The execution of a single start is always performed by a single thread, thus for `nstart = 1` only one thread will be used regardless of the value of this parameter.

`split` Enables split mode. This mode discovers new clusters after initial clustering, by trying to split single clusters into two to lower the cost function.

For each start (`nstart`), initial clustering will be performed and then split. The number of starts in the initial clustering before split is driven by the `split.initial.starts` parameter.

`split.depth` Cluster subdivision depth used in split mode. Usually a value less than 10 is sufficient (when after each subdivision, new clusters have similar sizes). For some data, subdivisions may often produce a cluster (one of the two) that will not be split further, in that case a higher value of the `split.depth` is required.

`split.tries` The number of attempts that are made when trying to split a cluster in split mode.

`split.limit` Maximum number of centers to be discovered in split mode.

`split.initial.starts`
The number of 'standard' starts performed before starting split.

`readline` Used only in the interactive mode. If `readline` is TRUE, at each iteration, before plotting it will wait for the user to press <Return> instead of standard "before plotting" (`par(ask = TRUE)`) waiting.

Details

In the context of implementation, Cross-Entropy Clustering (CEC) aims to partition *m* points into *k* clusters so as to minimize the cost function (energy **E** of the clustering) by switching the points between clusters. The presented method is based on the adapted Hartigan approach, where we reduce clusters which cardinalities decreased below some small prefixed level.

The energy function **E** is given by:

$$E(Y_1, \mathcal{F}_1; ...; Y_k, \mathcal{F}_k) = \sum_{i=1}^{k} p(Y_i) \cdot (-ln(p(Y_i)) + H^{\times}(Y_i \| \mathcal{F}_i))$$

where *Yi* denotes the *i*-th cluster, *p(Yi)* is the ratio of the number of points in *i*-th cluster to the total number points, **H***(Yi|Fi)* is the value of cross-entropy, which represents the internal cluster energy function of data *Yi* defined with respect to a certain Gaussian density family *Fi*, which encodes the type of clustering we consider.

The value of the internal energy function **H** depends on the covariance matrix (computed using maximum-likelihood method) and the mean (in case of the *mean* model) of the points in the cluster. Seven implementations of **H** have been proposed (expressed as a type - model - of the clustering):

- "all" - All Gaussian densities. Data will form ellipsoids with arbitrary radiuses.
- "covariance" - Gaussian densities with a fixed given covariance. The shapes of clusters depend on the given covariance matrix (additional parameter).
- "fixedr" - Special case of "covariance", where the covariance matrix equals *rI* for the given *r* (additional parameter). The clustering will have a tendency to divide data into balls with approximate radius proportional to the square root of *r*.
- "spherical" - Spherical (radial) Gaussian densities (covariance proportional to the identity). Clusters will have a tendency to form balls of arbitrary sizes.
- "diagonal" - Gaussian densities with diagonal covariane. Data will form ellipsoids with radiuses parallel to the coordinate axes.
- "eigenvalues" - Gaussian densities with covariance matrix having fixed eigenvalues (additional parameter). The clustering will try to divide the data into fixed-shaped ellipsoids rotated by an arbitrary angle.
- "mean" Gaussian densities with a fixed mean. Data will be covered with ellipsoids with fixed centers.

The implementation of `cec` function allows mixing of clustering types.

Value

Returns an object of class "cec" with available components: "data", "cluster", "probabilities", "centers", "cost.function", "nclusters", "iterations", "cost", "covariances", "covariances.model", "time".

Author(s)

Konrad Kamieniecki, Jacek Tabor, Przemysław Spurek

References

Spurek, P. and Tabor, J. (2014) Cross-Entropy Clustering *Pattern Recognition* **47, 9** 3046–3059

See Also

CEC-package.

Examples

```
#
# Cross-Entropy Clustering
#

## Example of clustering random data set of 3 Gaussians,
## 10 random initial centers and 7% as minimal cluster size.

m1 = matrix(rnorm(2000, sd=1), ncol=2)
m2 = matrix(rnorm(2000, mean = 3, sd = 1.5), ncol = 2)
m3 = matrix(rnorm(2000, mean = 3, sd = 1), ncol = 2)
m3[,2] = m3[,2] - 5
m = rbind(m1, m2, m3)
par(ask = TRUE)
plot(m, cex = 0.5, pch = 19)
## Clustering result:
Z = cec(m, 10, iter.max = 100, card.min="7%")
plot(Z)
# Result:
Z
## Example of clustering mouse-like set using spherical Gaussian densities.
m = mouseset(n=7000, r.head=2, r.left.ear=1.1, r.right.ear=1.1, left.ear.dist=2.5,
right.ear.dist=2.5, dim=2)
plot(m, cex = 0.5, pch = 19)
## Clustering result:
Z = cec(m, 3, type="sp", iter.max = 100, nstart=4, card.min="5%")
plot(Z)
# Result:
Z

## Example of clustering data set "Tset" using "eigenvalues" clustering type.
data(Tset)
plot(Tset, cex = 0.5, pch = 19)
centers = init.centers(Tset, 2)
## Clustering result:
Z <- cec(Tset, 5, "eigenvalues", param=c(0.02,0.002), nstart=4)
plot(Z)
# Result:
Z

## Example of using CEC split method starting with a single cluster.
data(mixShapes)
plot(mixShapes, cex = 0.5, pch = 19)
## Clustering result:
```

```
Z <- cec(mixShapes, 1, split=TRUE)
plot(Z)
# Result:
Z
```

`fourGaussians` *fourGaussians*

Description

Matrix of 2-dimensional points of four Gaussians.

Usage

```
data(fourGaussians)
```

Examples

```
data(fourGaussians)
plot(fourGaussians, cex = 0.5, pch = 19);
```

`init.centers` *Center initialization*

Description

Creates a matrix of k points (centers) based on a given matrix of points. One of two method can be used: Kmeans++ centers initialization method or a random choice of data points.

Usage

```
init.centers(x, k, method = c("kmeans++", "random"))
```

Arguments

x	Dataset as a matrix of n-dimensional points.
k	Number of points (centers) to generate.
method	Generation method. Possible values are: "kmeans++", "random.points".

Value

Matrix points (centers) with k rows.

Examples

```
m = matrix(runif(3000), 1000, 3)
init.centers(m, 3, method = "km")
```

mixShapes *mixShapes*

Description

Matrix of 2-dimensional points that form circular and elliptical patterns.

Usage

```
data(mixShapes)
```

Examples

```
data(mixShapes)
plot(mixShapes, cex = 0.5, pch = 19);
```

mouseset *Mouse set*

Description

Creates a matrix of `dim`-dimensional points that form a "mouse-like" set with uniform density.

Usage

```
mouseset(n = 4000, r.head = 2, r.left.ear = 1.1, r.right.ear = 1.1, left.ear.dist = 2.5,
right.ear.dist = 2.5, dim = 2)
```

Arguments

`n`	Number of points to generate.
`r.head`	Radius of mouse head.
`r.left.ear`	Radius of mouse left ear.
`r.right.ear`	Radius of mouse right ear.
`left.ear.dist`	Distance between the center of the head and the center the left ear.
`right.ear.dist`	Distance between the center of the head and the center the right ear.
`dim`	Dimension of points.

Value

Matrix of points with `n` rows and `dim` cols.

See Also

`ball`

Examples

```
M = mouseset(n=7000, r.head=2, r.left.ear=1.1, r.right.ear=1.1, left.ear.dist=2.5,
right.ear.dist=2.5, dim=2)
plot(M, cex = 0.5, pch = 19)
```

`plot.cec` *Plot CEC.*

Description

Presents the results of `cec` function in the form of a plot. Colors of data points depend of the cluster they belong to. Ellipses are drawn with regards to the covariance (either model or sample) of each cluster.

Usage

```
## S3 method for class 'cec'
plot(x, col, cex = 0.5, pch = 19, cex.centers = 1, pch.centers = 8,
ellipses.lwd = 4, ellipses = TRUE, model = T, xlab, ylab, ...)
```

Arguments

`x`	The result of `cec` function.
`col`	Use this argument to change default colors of points in the clusters.
`cex`	Basically the size of the points, see points.
`pch`	See points.
`cex.centers`	The same as `cex` parameter, except that it's related to the centers' means.
`pch.centers`	The same as `pch` parameter, except that it's related to the centers' means.
`ellipses.lwd`	Width of ellipses, points.
`ellipses`	If this parameter is TRUE, ellipses will be drawn.
`model`	If this parameter is TRUE, the model (expected) covariance will be used for each cluster insted of sample covariance (MLE) of the points in the cluster, when drawing ellipses.
`xlab`	See plot.
`ylab`	See plot.
`...`	Arguments are passed to `plot` function when drawing data points.

See Also

`print.cec`

Examples

```
## See the examples of function cec.
```

print.cec *Print CEC.*

Description

Presents a structure of the cec results object in the form of text.

Usage

```
## S3 method for class 'cec'
print(x, ...)
```

Arguments

x	Result of the cec function.
...	Ignored.

See Also

plot.cec

Examples

```
## See the examples of function cec.
```

run.cec.tests *CEC package tests.*

Description

This function is used to run cec package "unit test"-like system. The set of tests is located in inst/cec_tests directory and it consists of .R files defining each test case. This is also used for R CMD check.

Usage

```
run.cec.tests()
```

threeGaussians *threeGaussians*

Description

Matrix of 2-dimensional points from three Gaussians with means equal (0, 0).

Usage

```
data(threeGaussians)
```

Examples

```
data(threeGaussians)
plot(threeGaussians, cex = 0.5, pch = 19);
```

Tset *Tset*

Description

Matrix of 2-dimensional points that form T letter.

Usage

```
data(Tset)
```

Examples

```
data(Tset)
plot(Tset, cex = 0.5, pch = 19);
```

Index