Package 'CEC'

October 11, 2024

Type Package

Title Cross-Entropy Clustering

Version 0.11.2

Date 2024-10-10

Maintainer Simon Garnier <garnier@njit.edu>

Description Splits data into Gaussian type clusters using the Cross-Entropy Clustering ('CEC') method. This method allows for the simultaneous use of various types of Gaussian mixture models, for performing the reduction of unnecessary clusters, and for discovering new clusters by splitting them. 'CEC' is based on the work of Spurek, P. and Tabor, J. (2014) <doi:10.1016/j.patcog.2014.03.006>.

ByteCompile true

URL https://github.com/swarm-lab/cec, https://swarm-lab.github.io/cec/

Encoding UTF-8

LazyData true

Imports graphics, methods, stats, utils

NeedsCompilation yes

License GPL-3

RoxygenNote 7.3.2

Suggests covr

Author Kamieniecki Konrad [aut],
Spurek Przemyslaw [ctb],
Simon Garnier [cre, ctb] (<https://orcid.org/0000-0002-3886-3974>)

Repository CRAN

Date/Publication 2024-10-10 22:10:06 UTC

Contents

cec *Cross-Entropy Clustering*

Description

cec performs Cross-Entropy Clustering on a data matrix. See Details for an explanation of Cross-Entropy Clustering.

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 = FALSE,
  interactive = FALSE,
  threads = 1,
  split = FALSE,
  split.depth = 8,
  split.tries = 5,
  split.limit = 100,
  split.initial.starts = 1,
  readline = TRUE
)
```

Arguments

x	A numeric matrix of data. Each row corresponds to a distinct observation; each column corresponds to a distinct variable/dimension. It must not contain NA values.
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, ...)). It must not contain NA values.

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` The 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 a single type can be used.

`iter.max` The maximum number of iterations of the clustering algorithm.

`nstart` The number of clusterings to perform (with different initial centers). Only the best clustering (with the lowest cost) will be returned. A value grater than 1 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 = TRUE`), the whole procedure (initial clustering + split) will be performed `nstart` times, which may take some time.

`param` The parameter (or parameters) specific to a particular type of clustering. Not all types of clustering require parameters. The types that require parameter are: "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).

`centers.init` The method used to automatically initialize the centers. Possible values are: "kmeans++" (default) and "random".

`card.min` The minimal cluster cardinality. If the number of observations in a cluster becomes lower than card.min, the cluster is removed. This argument can be either an integer number or a string ending with a percent sign (e.g. "5%").

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

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

`threads` The number of threads to use or "auto" to use the 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` If `TRUE`, the function will attempt to discover new clusters after the initial clustering, by trying to split single clusters into two and check whether it lowers the cost function.

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

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

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

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

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

`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 the standard 'before plotting' waiting (`graphics::par(ask = TRUE)`).

Details

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 Hartigan approach, where we remove 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) 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

An object of class cec with the following attributes: data, cluster, probability, centers, cost.function, nclusters, iterations, cost, covariances, covariances.model, time.

References

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

See Also

CEC-package, plot.cec, print.cec

Examples

```
## Example of clustering a random data set of 3 Gaussians, with 10 random
## initial centers and a minimal cluster size of 7% of the total data set.

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)

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` *Four Gaussian Clusters*

Description

Matrix of 2-dimensional points forming four Gaussian clusters.

Examples

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

`init.centers` *Cluster Center Initialization*

Description

`init.centers` automatically initializes the centers of the clusters before running the Cross-Entropy Clustering algorithm.

Usage

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

Arguments

`x`	A numeric matrix of data. Each row corresponds to a distinct observation; each column corresponds to a distinct variable/dimension. It must not contain `NA` values.
`k`	An integer indicating the number of cluster centers to initialize.
`method`	A character string indicating the initialization method to use. It can take the following values: **"kmeans++":** the centers are selected using the k-means++ algorithm. **"random":** the centers are randomly selected among the values in `x`

Value

A matrix with `k` rows and `ncol(x)` columns.

References

Arthur, D., & Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 1027–1035.

Examples

```
## See the examples provided with the cec() function.
```

`mixShapes` *Mixed Shapes Clusters*

Description

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

Examples

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

`mouseset` *Mouse*

Description

`mouseset` generates a cluster of points uniformly distributed inside a "mouse head" shape.

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` The number of points (default: 4000).

`r.head` The radius of the mouse's head (default: 2).

`r.left.ear`, `r.right.ear`
The radii of the left and right ear of the mouse's head (default: 1.1).

`left.ear.dist`, `right.ear.dist`
The distance between the center of the mouse's head and the center of the left and right ear (default: 2.5).

`dim` The dimensionality of the mouse's head (default: 2).

Value

A matrix with `n` rows and `dim` columns.

Examples

```
plot(mouseset())
```

plot.cec *Plot CEC Objects*

Description

plot.cec presents the results from the cec function in the form of a plot. The colors of the data points represent the cluster they belong to. Ellipses are drawn to represent the covariance (of either the model or the 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 = TRUE,
  ellipses.lwd = 4,
  ellipses.lty = 2,
  model = TRUE,
  xlab,
  ylab,
  ...
)
```

Arguments

x	A cec object resulting from the cec function.
col	A specification for the default plotting color of the points in the clusters. See par for more details.
cex	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. See par for more details.
pch	Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par for more details.
cex.centers	The same as cex, except that it applies only to the centers' means.
pch.centers	The same as pch, except that it applies only to the centers' means.
ellipses	If this parameter is TRUE, covariance ellipses will be drawn.
ellipses.lwd	The line width of the covariance ellipses. See lwd in par for more details.
ellipses.lty	The line type of the covariance ellipses. See lty in par for more details.
model	If this parameter is TRUE, the model (expected) covariance will be used for each cluster instead of the sample covariance (MLE) of the points in the cluster, when drawing the covariance ellipses.

`xlab`	A label for the x axis. See plot for more details.
`ylab`	A label for the y axis. See plot for more details.
`...`	Additional arguments passed to `plot` when drawing data points.

Value

This function returns nothing.

See Also

`cec`, `print.cec`

Examples

```
## See the examples provided with the cec() function.
```

`print.cec` *Printing Cross Entropy Clusters*

Description

Print objects of class `cec`.

Usage

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

Arguments

`x`	An object produced by `cec`.
`...`	Ignored.

Value

This function returns nothing.

See Also

`cec`, `plot.cec`

Examples

```
## See the examples provided with the cec() function.
```

threeGaussians *Three Gaussian Clusters*

Description

Matrix of 2-dimensional points forming three Gaussian clusters.

Examples

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

Tset *T-Shaped Clusters*

Description

Matrix of 2-dimensional points that form the letter T.

Examples

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

Index