Package 'cclust'

May 8, 2026

Title Convex Clustering Methods and Clustering Indexes

Version 0.6-27

Description Convex Clustering methods, including K-means algorithm, On-line Update algorithm (Hard Competitive Learning) and Neural Gas algorithm (Soft Competitive Learning), and calculation of several indexes for finding the number of clusters in a data set.

Imports stats

License GPL-2

NeedsCompilation yes

Author Evgenia Dimitriadou [aut], Kurt Hornik [ctb, cre] (ORCID: <https://orcid.org/0000-0003-4198-9911>)

Maintainer Kurt Hornik <Kurt.Hornik@R-project.org>

Repository CRAN

Date/Publication 2026-02-18 08:46:18 UTC

Contents

`cclust` *Convex Clustering*

Description

The data given by x is clustered by an algorithm.

If centers is a matrix, its rows are taken as the initial cluster centers. If centers is an integer, centers rows of x are randomly chosen as initial values.

The algorithm stops, if no cluster center has changed during the last iteration or the maximum number of iterations (given by iter.max) is reached.

If verbose is TRUE, only for "kmeans" method, displays for each iteration the number of the iteration and the numbers of cluster indices which have changed since the last iteration is given.

If dist is "euclidean", the distance between the cluster center and the data points is the Euclidian distance (ordinary kmeans algorithm). If "manhattan", the distance between the cluster center and the data points is the sum of the absolute values of the distances of the coordinates.

If method is "kmeans", then we have the kmeans clustering method, which works by repeatedly moving all cluster centers to the mean of their Voronoi sets. If "hardcl" we have the On-line Update (Hard Competitive learning) method, which works by performing an update directly after each input signal, and if "neuralgas" we have the Neural Gas (Soft Competitive learning) method, that sorts for each input signal the units of the network according to the distance of their reference vectors to input signal.

If rate.method is "polynomial", the polynomial learning rate is used, that means $1/t$, where t stands for the number of input data for which a particular cluster has been the winner so far. If "exponentially decaying", the exponential decaying learning rate is used according to $par1 * (par2/par1)^{(iter/itermax)}$ where $par1$ and $par2$ are the initial and final values of the learning rate.

The parameters rate.par of the learning rate, where if rate.method is "polynomial" then by default rate.par=1.0, otherwise rate.par=(0.5,1e-5).

Usage

```
cclust (x, centers, iter.max=100, verbose=FALSE, dist="euclidean",
        method= "kmeans", rate.method="polynomial", rate.par=NULL)
```

Arguments

x	Data matrix where columns correspond to variables and rows to observations
centers	Number of clusters or initial values for cluster centers
iter.max	Maximum number of iterations
verbose	If TRUE, make some output during learning
dist	If "euclidean", then mean square error, if "manhattan ", the mean absolute error is used.
method	If "kmeans", then we have the kmeans clustering method, if "hardcl" we have the On-line Update (Hard Competitive learning) method, and if "neuralgas", we have the Neural Gas (Soft Competitive learning) method. Abbreviations of the method names are accepted.
rate.method	If "kmeans", then k-means learning rate, otherwise exponential decaying learning rate. It is used only for the Hardcl method.
rate.par	The parameters of the learning rate.

Value

`cclust` returns an object of class `"cclust"`.

`centers`	The final cluster centers.
`initcenters`	The initial cluster centers.
`ncenters`	The number of the centers.
`cluster`	Vector containing the indices of the clusters where the data points are assigned to.
`size`	The number of data points in each cluster.
`iter`	The number of iterations performed.
`changes`	The number of changes performed in each iteration step with the Kmeans algorithm.
`dist`	The distance measure used.
`method`	The algorithm method being used.
`rate.method`	The learning rate being used by the Hardcl clustering method.
`rate.par`	The parameters of the learning rate.
`call`	Returns a call in which all of the arguments are specified by their names.
`withinss`	Returns the sum of square distances within the clusters.

Author(s)

Evgenia Dimitriadou

See Also

`predict.cclust`

Examples

```
## a 2-dimensional example
x<-rbind(matrix(rnorm(100,sd=0.3),ncol=2),
         matrix(rnorm(100,mean=1,sd=0.3),ncol=2))
cl<-cclust(x,2,20,verbose=TRUE,method="kmeans")
plot(x, col=cl$cluster)

## a 3-dimensional example
x<-rbind(matrix(rnorm(150,sd=0.3),ncol=3),
         matrix(rnorm(150,mean=1,sd=0.3),ncol=3),
         matrix(rnorm(150,mean=2,sd=0.3),ncol=3))
cl<-cclust(x,6,20,verbose=TRUE,method="kmeans")
plot(x, col=cl$cluster)

## assign classes to some new data
y<-rbind(matrix(rnorm(33,sd=0.3),ncol=3),
         matrix(rnorm(33,mean=1,sd=0.3),ncol=3),
         matrix(rnorm(3,mean=2,sd=0.3),ncol=3))
         ycl<-predict(cl, y)
         plot(y, col=ycl$cluster)
```

clustIndex *Cluster Indexes*

Description

y is the result of a clustering algorithm of class such as "cclust". This function is calculating the values of several clustering indexes. The values of the indexes can be independently used in order to determine the number of clusters existing in a data set.

Usage

```
clustIndex ( y, x, index = "all" )
```

Arguments

y	Object of class "cclust" returned by a clustering algorithm such as kmeans
x	Data matrix where columns correspond to variables and rows to observations
index	The indexes that are calculated "calinski", "cindex", "db", "hartigan", "ratkowsky", "scott", "marriot", "ball", "trcovw", "tracew", "friedman", "rubin", "ssi", "likelihood", and "all" for all the indexes. Abbreviations of these names are also accepted.

Details

The description of the indexes is categorized into 3 groups, based on the statistics mainly used to compute them.

The first group is based on the sum of squares within (SSW) and between (SSB) the clusters. These statistics measure the dispersion of the data points in a cluster and between the clusters respectively. These indexes are:

calinski: $(SSB/(k-1))/(SSW/(n-k))$, where n is the number of data points and k is the number of clusters.

hartigan: then $\log(SSB/SSW)$.

ratkowsky: $mean(\sqrt{(varSSB/varSST)})$, where $varSSB$ stands for the SSB for every variable and $varSST$ for the total sum of squares for every variable.

ball: SSW/k, where k is the number of clusters.

The second group is based on the statistics of T, i.e., the scatter matrix of the data points, and W, which is the sum of the scatter matrices in every group. These indexes are:

scott: $n\log(|T|/|W|)$, where n is the number of data points and $|\cdot|$ stands for the determinant of a matrix.

marriot: $k^2|W|$, where k is the number of clusters.

trcovw: $TraceCovW$.

tracew: $TraceW$.

friedman: $TraceW^{(-1)}B$, where B is the scatter matrix of the cluster centers.

rubin: $|T|/|W|$.

The third group consists of four algorithms not belonging to the previous ones and not having anything in common.

cindex: if the data set is binary, then while the C-Index is a cluster similarity measure, is expressed as:
$[d_{(w)} - \min(d_{(w)})]/[\max(d_{(w)}) - \min(d_{(w)})]$, where $d_{(w)}$ is the sum of all $n_{(d)}$ within cluster distances, $\min(d_{(w)})$ is the sum of the $n_{(d)}$ smallest pairwise distances in the data set, and $\max(d_{(w)})$ is the sum of the $n_{(d)}$ biggest pairwise distances. In order to compute the C-Index all pairwise distances in the data set have to be computed and stored. In the case of binary data, the storage of the distances is creating no problems since there are only a few possible distances. However, the computation of all distances can make this index prohibitive for large data sets.

db: $R = (1/n) * sum(R_{(i)})$ where $R_{(i)}$ stands for the maximum value of $R_{(ij)}$ for $i \neq j$, and $R_{(ij)}$ for $R_{(ij)} = (SSW_{(i)} + SSW_{(j)})/DC_{(ij)}$, where $DC_{(ij)}$ is the distance between the centers of two clusters i, j.

likelihood: under the assumption of independence of the variables within a cluster, a cluster solution can be regarded as a mixture model for the data, where the cluster centers give the probabilities for each variable to be 1. Therefore, the negative Log-likelihood can be computed and used as a quantity measure for a cluster solution. Note that the assumptions for applying special penalty terms, like in AIC or BIC, are not fulfilled in this model, and also they show no effect for these data sets.

ssi: this "Simple Structure Index" combines three elements which influence the interpretability of a solution, i.e., the maximum difference of each variable between the clusters, the sizes of the most contrasting clusters and the deviation of a variable in the cluster centers compared to its overall mean. These three elements are multiplicatively combined and normalized to give a value between 0 and 1.

Value

Returns an vector with the indexes values.

Author(s)

Evgenia Dimitriadou and Andreas Weingessel

References

Weingessel A, Dimitriadou E, Dolnicar S (1999). "An Examination of Indexes for Determining the Number of Clusters in Binary Data Sets." Working Paper 29, SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business. doi:10.57938/0409c45b603b4daea2e0893a0cb0737d.

See Also

cclust, kmeans

Examples

```
# a 2-dimensional example
x<-rbind(matrix(rnorm(100,sd=0.3),ncol=2),
         matrix(rnorm(100,mean=1,sd=0.3),ncol=2))
cl<-cclust(x,2,20,verbose=TRUE,method="kmeans")
resultindexes <- clustIndex(cl,x, index="all")
resultindexes
```

`predict.cclust` *Assign clusters to new data*

Description

Assigns each data point (row in `newdata`) the cluster corresponding to the closest center found in `object`.

Usage

```
## S3 method for class 'cclust'
predict(object, newdata, ...)
```

Arguments

`object`	Object of class `"cclust"` returned by a clustering algorithm such as `cclust`
`newdata`	Data matrix where columns correspond to variables and rows to observations
`...`	currently not used

Value

`predict.cclust` returns an object of class `"cclust"`. Only `size` is changed as compared to the argument `object`.

`cluster`	Vector containing the indices of the clusters where the data is mapped.
`size`	The number of data points in each cluster.

Author(s)

Evgenia Dimitriadou

See Also

`cclust`

Examples

```
# a 2-dimensional example
x<-rbind(matrix(rnorm(100,sd=0.3),ncol=2),
         matrix(rnorm(100,mean=1,sd=0.3),ncol=2))
cl<-cclust(x,2,20,verbose=TRUE,method="kmeans")
plot(x, col=cl$cluster)

# a 3-dimensional example
x<-rbind(matrix(rnorm(150,sd=0.3),ncol=3),
         matrix(rnorm(150,mean=1,sd=0.3),ncol=3),
         matrix(rnorm(150,mean=2,sd=0.3),ncol=3))
cl<-cclust(x,6,20,verbose=TRUE,method="kmeans")
plot(x, col=cl$cluster)

# assign classes to some new data
y<-rbind(matrix(rnorm(33,sd=0.3),ncol=3),
         matrix(rnorm(33,mean=1,sd=0.3),ncol=3),
         matrix(rnorm(3,mean=2,sd=0.3),ncol=3))
ycl<-predict(cl, y)
plot(y, col=ycl$cluster)
```

Index