Package ‘HPdcluster’

February 7, 2014

Type Package

Title Distributed Clustering for Big Data

Version 0.4.0

Date 2014-01-22

Author Arash Fard

Maintainer Arash Fard <afard@vertica.com>

Depends R (>= 3.0.1), distributedR, MatrixHelper

Description It provides distributed Clustering. It is written based on the infrastructure created in HP-Labs for distributed computing in R.

License GPL (>= 2) + file LICENSE

R topics documented:

`HPdcluster-package` *Distributed clustering for Big Data*

Description

HPdcluster provides a few distributed clustering functions. It is written based on the infrastructure created in HP-Labs for distributed computing in R.

Details

Package:	HPdcluster
Type:	Package
Version:	0.4.0
Date:	2014-01-22

Main Functions:

- hpdkmeans: It is a distributed version of kmeans.
- sampleLoader: It is an example for loading a set of samples stored in a table to a darray.
- centerLoader: It is an example for loading a set of centers stored in a table to a matrix.

Author(s)

Arash Fard <afard@vertica.com>

References

1. Using R for Iterative and Incremental Processing. Shivaram Venkataraman, Indrajit Roy, Alvin AuYoung, Rob Schreiber. HotCloud 2012, Boston, USA.
2. http://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

---

`centerLoader` *A simple loader for initial centers of a cluster*

Description

centerLoader function is an example for loading a set of centers stored in a table to a matrix. It is assumed that centers are stored in a single table. All the rows of the table will be read, and each row will be a center.

Usage

```
centerLoader(tableName, features = list(...), conf)
```

Arguments

`tableName`	it is the name of the table in the database in string format.
`features`	the list of the name of columns corresponding to the features of a center.
`conf`	the name of configuration in ODBC.INI file for connecting to the database.

Value

`X`	the matrix of centers

Author(s)

Arash Fard <afard@vertica.com>

Examples

```
 ## Not run:
    # Assuming that centers are stored in a table named "centers",
    # and the names of the columns are "col1", "col2", "col3", and "col4".
    loadedCenters <- centerLoader ("centers", list(col1", "col2",
    "col3", "col4"), conf="RDev")

## End(Not run)
```

hpdapply — *Cluster labeling*

Description

hpdapply function finds cluster label of a set of samples according to a given set of centers.

Usage

```
hpdapply(newdata, centers, trace=FALSE)
```

Arguments

`newdata`	a darray (dense or sparse) or a matrix which contains the samples.
`centers`	a matrix of cluster centres. Each row represents a center. Each sample in newdata will be assigned a label which indicates the row number of its corresponding center.
`trace`	when this argument is true, intermediate steps of the progress are displayed.

Details

This function applies the centers found by hpdkmeans on a new set of samples in order to label them.

Value

hpdapply returns a darray or a matrix based on the type of newdata which contains the corresponding label of each sample.

Author(s)

Arash Fard <afard@vertica.com>

Examples

```
 ## Not run:
    iris2 <- iris
    iris2$Species <- NULL
    centers <- matrix(c(5.901613,5.006000,6.850000,2.748387,3.428000,
3.073684,4.393548,1.462000,5.742105,1.433871,0.246000,2.071053),3,4)
    dimnames(centers) <- list(1L:3, colnames(iris2))

    library(distributedR)
    distributedR_start()
    (drs <- distributedR_status())
    nblocks = sum(drs$Inst)
    X <- as.darray(as.matrix(cbind(iris2$Sepal.Length,iris2$Sepal.Width,
iris2$Petal.Length,iris2$Petal.Width)),c(ceiling(length(iris2$Sepal.Length)
/nblocks),4))
    colnames(X) <- colnames(iris2)

    library(HPdcluster)
    mykm <- hpdkmeans(X,centers=3)
    newdata <- matrix(c(5,4,3,5,7,1,0,8),2,4)
    labels <- hpdapply(newdata,mykm$centers)

## End(Not run)
```

`hpdkmeans` *Distributed kmeans*

Description

hpdkmeans function is intended to be a distributed alternative for kmeans function.

Usage

```
hpdkmeans(X, centers, iter.max = 10, nstart = 1,
             sampling_threshold = 1e+06, trace = FALSE,
             na_action = c("exclude","fail"))
```

Arguments

`X`	a darray (dense or sparse) which contains the samples.
`centers`	either the number of clusters, say k, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) samples in X is chosen as the initial centres.
`iter.max`	the maximum number of iterations allowed.
`nstart`	when the value specified for 'centers' argument is a number, clustering will be performed several times and the best result is reported. The best result would be the one with highest value of 'withinss' regardless of its number of iterations. 'nstart' gives the number of times that a random set of centers is chosen and clustering is performed. When 'centers' argument is a matrix of centers, 'nstart' will be discarded.
`sampling_threshold`	threshold for the method which Randomly finds centers (centralized or distributed). It should be always smaller than 1e9. When (blockSize > sampling_threshold || nSample > 1e9), the distributed sampling is selected, in which first a set of blocks are randomly chosen, and then the centers are randomly selected from the samples of those bloks. Here, blockSize is the number of samples in each partition of X, and nSample is the total number of samples in X.
`trace`	when this argument is true, intermediate steps of the progress are displayed.
`na_action`	it indicates what should happen when the data contain missed values. Values of NA, NaN, and Inf in samples are treated as missed values. There are two options for this argument exclude and fail. When exclude is selected (the default choice), any sample with missed values will be ignored in the clustering process. In the darray which will be created for cluster, the value corresponding to these samples will be NA. When fail is selected, the function will stop in the case of any missed value in the dataset.

Details

The data given by X is clustered by the k-means method, which aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centres is minimized. At the minimum, all cluster centres are at the mean of their Voronoi sets (the set of data points which are nearest to the cluster centre).

The algorithm of Lloyd–Forgy (Lloyd 1957 and Forgy 1965) is used at the current version. If an initial matrix of centres is supplied, it is possible that no point will be closest to one or more centres, which currently generates a warning message.

Value

hpdkmeans returns an object of class "hpdkmeans" which has a print and a fitted method. It is a list with components:

`cluster`	A darray of integers (from 1:k) indicating the cluster to which each point is allocated.
`centers`	A matrix of cluster centres.
`totss`	The total sum of squares.
`withinss`	Vector of within-cluster sum of squares, one component per cluster.
`tot.withinss`	Total within-cluster sum of squares, i.e., sum(withinss).
`betweenss`	The between-cluster sum of squares, i.e. totss-tot.withinss.
`size`	The number of points in each cluster.
`iter`	The number of iterations used for clustering. Its value will be iter.max+1 when the algorithm is not converged.

Author(s)

Arash Fard <afard@vertica.com>

References

Forgy, E. W. (1965) Cluster analysis of multivariate data: efficiency vs interpretability of classifications. Biometrics 21, 768–769.

Lloyd, S. P. (1957, 1982) Least squares quantization in PCM. Technical Note, Bell Laboratories. Published in 1982 in IEEE Transactions on Information Theory 28, 128–137.

See Also

`kmeans`

Examples

```
 ## Not run:
    iris2 <- iris
    iris2$Species <- NULL
    centers <- matrix(c(5.901613,5.006000,6.850000,2.748387,3.428000,
3.073684,4.393548,1.462000,5.742105,1.433871,0.246000,2.071053),3,4)
    dimnames(centers) <- list(1L:3, colnames(iris2))

    library(distributedR)
    distributedR_start()
    (drs <- distributedR_status())
    nblocks = sum(drs$Inst)
    X <- as.darray(as.matrix(cbind(iris2$Sepal.Length,iris2$Sepal.Width,
iris2$Petal.Length,iris2$Petal.Width)),c(ceiling(length(iris2$Sepal.Length)
/nblocks),4))
    colnames(X) <- colnames(iris2)

    library(HPdcluster)
    mykm1 <- hpdkmeans(X,centers=centers)

    mykm2 <- hpdkmeans(X,centers=3)

## End(Not run)
```

`sampleLoader` *A simple sample loader*

Description

sampleLoader function is an example for loading a set of samples stored in a table to a darray. It is assumed that samples are stored in a single table, and the table contains a column called 'rowid'. It is also assumed that 'rowid' starts from 0, and there is no missed 'rowid'.

Usage

```
sampleLoader(tableName, features = list(...), conf)
```

Arguments

`tableName`	it is the name of the table in the database in string format.
`features`	the list of the name of columns corresponding to the features of a sample.
`conf`	the name of configuration in ODBC.INI file for connecting to the database.

Value

`X`	the darray of samples

Author(s)

Arash Fard <afard@vertica.com>

Examples

```
 ## Not run:
    require(distributedR)
    distributedR_start()
    # Assuming that samples are stored in a table named "samples",
    # and the names of the columns are "col1", "col2", "col3", and "col4".
    loadedSamples <- sampleLoader ("samples", list(col1", "col2",
    "col3", "col4"), conf="RDev")

## End(Not run)
```

Index