Package ‘ClusterR’

August 22, 2018

Type Package

Title Gaussian Mixture Models, K-Means, Mini-Batch-Kmeans and K-Medoids Clustering

Version 1.1.4

Date 2018-08-21

Author Lampros Mouselimis <mouselimislampros@gmail.com>

Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

BugReports https://github.com/mlampros/ClusterR/issues

URL https://github.com/mlampros/ClusterR

Description Gaussian mixture models, k-means, mini-batch-kmeans and k-medoids clustering with the option to plot, validate, predict (new data) and estimate the optimal number of clusters. The package takes advantage of 'RcppArmadillo' to speed up the computationally intensive parts of the functions. For more information, see (i) "Clustering in an Object-Oriented Environment" by Anja Struyf, Mia Hubert, Peter Rousseeuw (1997), Journal of Statistical Software, <doi:10.18637/jss.v001.i04>; (ii) "Web-scale k-means clustering" by D. Sculley (2010), ACM Digital Library, <doi:10.1145/1772690.1772862>; (iii) "Armadillo: a template-based C++ library for linear algebra" by Sanderson et al (2016), The Journal of Open Source Software, <doi:10.21105/joss.00026>.

License MIT + file LICENSE

LazyData TRUE

Depends R(>= 3.2.3), gtools

Imports Rcpp (>= 0.12.5), OpenImageR, graphics, grDevices, utils, gmp, FD, stats, ggplot2

LinkingTo Rcpp, RcppArmadillo (>= 0.7.2)

Suggests testthat, covr, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-08-22 16:10:07 UTC
### R topics documented:

- `center_scale` .................................................. 2
- Clara_Medoids .................................................. 3
- Cluster_Medoids ............................................... 5
- dietary_survey_IBS ............................................ 6
- distance_matrix .............................................. 7
- external_validation .......................................... 8
- GMM .............................................................. 9
- KMeans_arma ................................................... 10
- KMeans_rcpp ................................................... 12
- MiniBatchKmeans ............................................. 13
- mushroom ...................................................... 15
- Optimal_Clusters_GMM ....................................... 17
- Optimal_Clusters_KMeans .................................... 18
- Optimal_Clusters_Medoids ................................... 20
- plot_2d ......................................................... 21
- predict_GMM .................................................. 22
- predict_KMeans ................................................ 23
- predict_MBatchKMeans ....................................... 24
- predict_Medoids ............................................. 25
- Silhouette_Dissimilarity_Plot .............................. 26
- soybean ......................................................... 27

### Index

- `center_scale` .................................................. 29

---

**center_scale**

*Function to scale and/or center the data*

**Description**

Function to scale and/or center the data

**Usage**

```r
center_scale(data, mean_center = TRUE, sd_scale = TRUE)
```

**Arguments**

- `data` : matrix or data frame
- `mean_center` : either TRUE or FALSE. If mean_center is TRUE then the mean of each column will be subtracted
- `sd_scale` : either TRUE or FALSE. See the details section for more information
Details
If sd_scale is TRUE and mean_center is TRUE then each column will be divided by the standard
deviation. If sd_scale is TRUE and mean_center is FALSE then each column will be divided by
sqrt( sum(x^2) / (n-1) ). In case of missing values the function raises an error. In case that the
standard deviation equals zero then the standard deviation will be replaced with 1.0, so that NaN’s
can be avoided by division.

Value
a matrix

Examples

data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = center_scale(dat, mean_center = TRUE, sd_scale = TRUE)

---

Clara_Medoids

Clustering large applications

Description
Clustering large applications

Usage
Clara_Medoids(data, clusters, samples, sample_size,
distance_metric = "euclidean", minkowski_p = 1, threads = 1,
swap_phase = TRUE, fuzzy = FALSE, verbose = FALSE, seed = 1)

Arguments
data matrix or data frame
clusters the number of clusters
samples number of samples to draw from the data set
sample_size fraction of data to draw in each sample iteration. It should be a float number
greater than 0.0 and less or equal to 1.0
distance_metric a string specifying the distance method. One of, euclidean, manhattan, chebyshev, canberra, braycurtis, pearson_correlation, simple_matching_coefficient, minkowski, hamming, jaccard_coefficient, Rao_coefficient, mahalanobis
Clara_Medoids

minkowski_p  a numeric value specifying the minkowski parameter in case that distance_metric = "minkowski"

threads  an integer specifying the number of cores to run in parallel. Openmp will be utilized to parallelize the number of the different sample draws

swap_phase  either TRUE or FALSE. If TRUE then both phases ('build' and 'swap') will take place. The `swap_phase` is considered more computationally intensive.

fuzzy  either TRUE or FALSE. If TRUE, then probabilities for each cluster will be returned based on the distance between observations and medoids

verbose  either TRUE or FALSE, indicating whether progress is printed during clustering

seed  integer value for random number generator (RNG)

Details

The Clara_Medoids function is implemented in the same way as the 'clara' (clustering large applications) algorithm (Kaufman and Rousseeuw(1990)). In the 'Clara_Medoids' the 'Cluster_Medoids' function will be applied to each sample draw.

Value

a list with the following attributes : medoids, medoid_indices, sample_indices, best_dissimilarity, clusters, fuzzy_probs (if fuzzy = TRUE), clustering_stats, dissimilarity_matrix, silhouette_matrix

Author(s)

Lampros Mouselimis

References

Anja Struyf, Mia Hubert, Peter J. Rousseeuw, (Feb. 1997), Clustering in an Object-Oriented Environment, Journal of Statistical Software, Vol 1, Issue 4

Examples

data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = center_scale(dat)
clm = Clara_Medoids(dat, clusters = 3, samples = 5, sample_size = 0.2, swap_phase = TRUE)
Cluster_Medoids  

Cluster_Medoids  

**Description**

Partitioning around medoids

**Usage**

Cluster_Medoids(data, clusters, distance_metric = "euclidean",
minkowski_p = 1, threads = 1, swap_phase = TRUE, fuzzy = FALSE,
verbose = FALSE, seed = 1)

**Arguments**

data  
matrix or data frame. The data parameter can be also a dissimilarity matrix, where the main diagonal equals 0.0 and the number of rows equals the number of columns

clusters  
the number of clusters

distance_metric  
a string specifying the distance method. One of, euclidean, manhattan, chebyshev, canberra, braycurtis, pearson_correlation, simple_matching_coefficient, minkowski, hamming, jaccard_coefficient, Rao_coefficient, mahalanobis

minkowski_p  
a numeric value specifying the minkowski parameter in case that distance_metric = "minkowski"

threads  
an integer specifying the number of cores to run in parallel

swap_phase  
either TRUE or FALSE. If TRUE then both phases (‘build’ and ‘swap’) will take place. The ‘swap_phase’ is considered more computationally intensive.

fuzzy  
either TRUE or FALSE. If TRUE, then probabilities for each cluster will be returned based on the distance between observations and medoids

verbose  
either TRUE or FALSE, indicating whether progress is printed during clustering

seed  
integer value for random number generator (RNG)

**Details**

The Cluster_Medoids function is implemented in the same way as the ‘pam’ (partitioning around medoids) algorithm (Kaufman and Rousseeuw(1990)). In comparison to k-means clustering, the function Cluster_Medoids is more robust, because it minimizes the sum of unsquared dissimilarities. Moreover, it doesn’t need initial guesses for the cluster centers.

**Value**

a list with the following attributes: medoids, medoid_indices, best_dissimilarity, dissimilarity_matrix, clusters, fuzzy_probs (if fuzzy = TRUE), silhouette_matrix, clustering_stats
Author(s)
Lampros Mouselimis

References
Anja Struyf, Mia Hubert, Peter J. Rousseeuw. (Feb. 1997), Clustering in an Object-Oriented Environment, Journal of Statistical Software, Vol 1, Issue 4

Examples

data(dietary_survey_IBS)

(dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)])

(dat = center_scale(dat))

(cm = Cluster_Medoids(dat, clusters = 3, distance_metric = 'euclidean', swap_phase = TRUE))

---

dietary_survey_IBS Synthetic data using a dietary survey of patients with irritable bowel syndrome (IBS)

Description
The data are based on the article "A dietary survey of patients with irritable bowel syndrome". The mean and standard deviation of the table 1 (Foods perceived as causing or worsening irritable bowel syndrome symptoms in the IBS group and digestive symptoms in the healthy comparative group) were used to generate the synthetic data.

Usage
data(dietary_survey_IBS)

Format
A data frame with 400 Instances and 43 attributes (including the class attribute, "class")

Details
The predictors are: bread, wheat, pasta, breakfast_cereal, yeast, spicy_food, curry, chinese_takeaway, chilli, cabbage, onion, garlic, potatoes, pepper, vegetables_unspecified, tomato, beans_and_pulses, mushroom, fatty_foods_unspecified, sauces, chocolate, fries, crisps, desserts, eggs, red_meat, processed_meat, pork, chicken, fish_shellfish, dairy_products_unspecified, cheese, cream, milk, fruit_unspecified, nuts_and_seeds, orange, apple, banana, grapes, alcohol, caffeine

The response variable ("class") consists of two groups: healthy-group (class == 0) vs. the IBS-patients (class == 1)
distance_matrix

References


Examples

data(dietary_survey_IBS)

X = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]

y = dietary_survey_IBS[, ncol(dietary_survey_IBS)]

distance_matrix  Distance matrix calculation

Description

Distance matrix calculation

Usage

distance_matrix(data, method = "euclidean", upper = FALSE, diagonal = FALSE, minkowski_p = 1, threads = 1)

Arguments

data matrix or data frame

method a string specifying the distance method. One of, euclidean, manhattan, chebyshev, canberra, braycurtis, pearson_correlation, simple_matching_coefficient, minkowski, hamming, jaccard_coefficient, Rao_coefficient, mahalanobis

upper either TRUE or FALSE specifying if the upper triangle of the distance matrix should be returned. If FALSE then the upper triangle will be filled with NA's

diagonal either TRUE or FALSE specifying if the diagonal of the distance matrix should be returned. If FALSE then the diagonal will be filled with NA's

minkowski_p a numeric value specifying the minkowski parameter in case that method = "minkowski"

threads the number of cores to run in parallel (if OpenMP is available)

Value

a matrix
external_validation

Examples

data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = distance_matrix(dat, method = 'euclidean', upper = TRUE, diagonal = TRUE)

Description

external clustering validation

Usage

external_validation(true_labels, clusters, method = "adjusted_rand_index", summary_stats = FALSE)

Arguments

true_labels  a numeric vector of length equal to the length of the clusters vector
clusters     a numeric vector (the result of a clustering method) of length equal to the length of the true_labels
method       one of rand_index, adjusted_rand_index, jaccard_index, fowlkes_Mallows_index, mirkin_metric, purity, entropy, nmi (normalized mutual information), var_info (variation of information), and nvi (normalized variation of information)
summary_stats besides the available methods the summary_stats parameter prints also the specificity, sensitivity, precision, recall and F-measure of the clusters

Details

This function uses external validation methods to evaluate the clustering results

Value

if summary_stats is FALSE the function returns a float number, otherwise it returns also a summary statistics table

Author(s)

Lampros Mouselimis
Examples

data(dietary_survey_IBS)

dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]

X = center_scale(dat)

km = KMeans_rcpp(X, clusters = 2, num_init = 5, max_iters = 100, initializer = 'kmeans++')

res = external_validation(dietary_survey_IBS$class, km$clusters, method = "adjusted_rand_index")

GMM

Gaussian Mixture Model clustering

Description

Gaussian Mixture Model clustering

Usage

GMM(data, gaussian_comps = 1, dist_mode = "eucl_dist",
     seed_mode = "random_subset", km_iter = 10, em_iter = 5,
     verbose = FALSE, var_floor = 1e-10, seed = 1)

Arguments

data             matrix or data frame

 gaussian_comps  the number of gaussian mixture components

 dist_mode        the distance used during the seeding of initial means and k-means clustering.
                  One of, eucl_dist, maha_dist.

 seed_mode        how the initial means are seeded prior to running k-means and/or EM algorithms. One of, static_subset, random_subset, static_spread, random_spread.

 km_iter          the number of iterations of the k-means algorithm

 em_iter          the number of iterations of the EM algorithm

 verbose         either TRUE or FALSE; enable or disable printing of progress during the k-
                  means and EM algorithms

 var_floor        the variance floor (smallest allowed value) for the diagonal covariances

 seed             integer value for random number generator (RNG)
Details

This function is an R implementation of the 'gmm_diag' class of the Armadillo library. The only exception is that user defined parameter settings are not supported, such as seed_mode = 'keepExisting'. For probabilistic applications, better model parameters are typically learned with dist_mode set to maha_dist. For vector quantisation applications, model parameters should be learned with dist_mode set to eucl_dist, and the number of EM iterations set to zero. In general, a sufficient number of k-means and EM iterations is typically about 10. The number of training samples should be much larger than the number of Gaussians. Seeding the initial means with static_spread and random_spread can be much more time consuming than with static_subset and random_subset. The k-means and EM algorithms will run faster on multi-core machines when OpenMP is enabled in your compiler (eg. -fopenmp in GCC)

Value

a list consisting of the centroids, covariance matrix (where each row of the matrix represents a diagonal covariance matrix), weights and the log-likelihoods for each gaussian component. In case of Error it returns the error message and the possible causes.

References

http://arma.sourceforge.net/docs.html

Examples

```r
data(dietary_survey_IBS)
dat = as.matrix(dietary_survey_IBS[, -ncol(dietary_survey_IBS)])
dat = center_scale(dat)
gmm = GMM(dat, 2, "maha_dist", "random_subset", 10, 10)
```

Description

k-means using the Armadillo library

Usage

```r
KMeans_arma(data, clusters, n_iter = 10, seed_mode = "random_subset", verbose = FALSE, CENTROIDS = NULL, seed = 1)
```
**Arguments**

- **data**: matrix or data frame
- **clusters**: the number of clusters
- **n_iter**: the number of clustering iterations (about 10 is typically sufficient)
- **seed_mode**: how the initial centroids are seeded. One of, *keep_existing, static_subset, random_subset, static_spread, random_spread*.
- **verbose**: either TRUE or FALSE, indicating whether progress is printed during clustering
- **CENTROIDS**: a matrix of initial cluster centroids. The rows of the CENTROIDS matrix should be equal to the number of clusters and the columns should be equal to the columns of the data. CENTROIDS should be used in combination with seed_mode 'keep_existing'.
- **seed**: integer value for random number generator (RNG)

**Details**

This function is an R implementation of the 'kmeans' class of the Armadillo library. It is faster than the KMeans_rcpp function but it lacks some features. For more info see the details section of the KMeans_rcpp function. The number of columns should be larger than the number of clusters or CENTROIDS. If the clustering fails, the means matrix is reset and a bool set to false is returned. The clustering will run faster on multi-core machines when OpenMP is enabled in your compiler (eg. -fopenmp in GCC)

**Value**

the centroids as a matrix. In case of Error it returns the error message, whereas in case of an empty centroids-matrix it returns a warning-message.

**References**

http://arma.sourceforge.net/docs.html

**Examples**

```r
data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = center_scale(dat)
km = KMeans_arma(dat, clusters = 2, n_iter = 10, "random_subset")
```
KMeans_rcpp

k-means using RcppArmadillo

Description

k-means using RcppArmadillo

Usage

KMeans_rcpp(data, clusters, num_init = 1, max_iters = 100,
initializer = "kmeans++", fuzzy = FALSE, verbose = FALSE,
CENTROIDS = NULL, tol = 1e-04, tol_optimal_init = 0.3, seed = 1)

Arguments

data matrix or data frame
clusters the number of clusters
num_init number of times the algorithm will be run with different centroid seeds
max_iters the maximum number of clustering iterations
initializer the method of initialization. One of, optimal_init, quantile_init, kmeans++ and random. See details for more information
fuzzy either TRUE or FALSE. If TRUE, then prediction probabilities will be calculated using the distance between observations and centroids
verbose either TRUE or FALSE, indicating whether progress is printed during clustering.
CENTROIDS a matrix of initial cluster centroids. The rows of the CENTROIDS matrix should be equal to the number of clusters and the columns should be equal to the columns of the data.
tol a float number. If, in case of an iteration (iteration > 1 and iteration < max_iters) ‘tol’ is greater than the squared norm of the centroids, then kmeans has converged
tol_optimal_init tolerance value for the ‘optimal_init’ initializer. The higher this value is, the far apart from each other the centroids are.
seed integer value for random number generator (RNG)

Details

This function has the following features in comparison to the KMeans_arma function:
It allows for multiple initializations (which can be parallelized if Openmp is available).
Besides optimal_init, quantile_init, random and kmeans++ initializations one can specify the centroids using the CENTROIDS parameter.
The running time and convergence of the algorithm can be adjusted using the num_init, max_iters and tol parameters.
If num_init > 1 then KMeans_rcpp returns the attributes of the best initialization using as criterion the within-cluster-sum-of-squared-error.

Initializers

- **optimal_init**: this initializer adds rows of the data incrementally, while checking that they do not already exist in the centroid-matrix [experimental]
- **quantile_init**: initialization of centroids by using the cumulative distance between observations and by removing potential duplicates [experimental]
- **random**: random selection of data rows as initial centroids

Value

a list with the following attributes: clusters, fuzzy_clusters (if fuzzy = TRUE), centroids, total_SSE, best_initialization, WCSS_per_cluster, obs_per_cluster, between.SS_DIV_total.SS

Author(s)

Lampros Mouselimis

Examples

data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = center_scale(dat)

km = KMeans_rcpp(dat, clusters = 2, num_init = 5, max_iters = 100, initializer = 'kmeans++')

Description

Mini-batch-k-means using RcppArmadillo

Usage

MiniBatchKmeans(data, clusters, batch_size = 10, num_init = 1,
max_iters = 100, init_fraction = 1, initializer = "kmeans++",
early_stop_iter = 10, verbose = FALSE, CENTROIDS = NULL, tol = 1e-04,
tol_optimal_init = 0.3, seed = 1)
Arguments

data: matrix or data frame
clusters: the number of clusters
batch_size: the size of the mini batches
num_init: number of times the algorithm will be run with different centroid seeds
max_iters: the maximum number of clustering iterations
init_fraction: percentage of data to use for the initialization centroids (applies if initializer is kmeans++ or optimal_init). Should be a float number between 0.0 and 1.0.
initializer: the method of initialization. One of, optimal_init, quantile_init, kmeans++ and random. See details for more information
early_stop_iter: continue that many iterations after calculation of the best within-cluster-sum-of-squared-error
verbose: either TRUE or FALSE, indicating whether progress is printed during clustering

DETAILS

This function performs k-means clustering using mini batches.

- __optimal_init__ : this initializer adds rows of the data incrementally, while checking that they do not already exist in the centroid-matrix [ experimental ]
- __quantile_init__ : initialization of centroids by using the cumulative distance between observations and by removing potential duplicates [ experimental ]
- __random__ : random selection of data rows as initial centroids

Value

a list with the following attributes: centroids, WCSS_per_cluster, best_initialization, iters_per_initialization

Author(s)

Lampros Mouselimis
The mushroom data

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like 'leaflets three, let it be' for Poisonous Oak and Ivy.

Usage

data(mushroom)

Format

A data frame with 8124 Instances and 23 attributes (including the class attribute, "class")

Details

The column names of the data (including the class) appear in the following order:
1. class: edible=e, poisonous=p
2. cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
3. cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
4. cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
5. bruises: bruises=t, no=f
6. odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
7. gill-attachment: attached=a, descending=d, free=f, notched=n
8. gill-spacing: close=c, crowded=w, distant=d
9. gill-size: broad=b, narrow=n
10. gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
11. stalk-shape: enlarging=e, tapering=t
12. stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
13. stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
14. stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
15. stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16. stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
17. veil-type: partial=p, universal=u
18. veil-color: brown=n, orange=o, white=w, yellow=y
19. ring-number: none=n, one=o, two=t
20. ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
21. spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
22. population: abundant=a, clustered=c, numerous=n, scattered=s, several=s, solitary=y
23. habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

References

Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a gp.cs.cmu.edu)
download source: https://archive.ics.uci.edu/ml/datasets/Mushroom

Examples

data(mushroom)

X = mushroom[, -1]

y = mushroom[, 1]
Optimal number of Clusters for the gaussian mixture models

Description

Optimal number of Clusters for the gaussian mixture models

Usage

```r
Optimal_Clusters_GMM(data, max_clusters, criterion = "AIC",
  dist_mode = "eucl_dist", seed_mode = "random_subset", km_iter = 10,
  em_iter = 5, verbose = FALSE, var_floor = 1e-10, plot_data = TRUE,
  seed = 1)
```

Arguments

- `data`: matrix or data frame
- `max_clusters`: the maximum number of clusters
- `criterion`: one of 'AIC' or 'BIC'
- `dist_mode`: the distance used during the seeding of initial means and k-means clustering. One of, `eucl_dist, maha_dist`.
- `seed_mode`: how the initial means are seeded prior to running k-means and/or EM algorithms. One of, `static_subset, random_subset, static_spread, random_spread`.
- `km_iter`: the number of iterations of the k-means algorithm
- `em_iter`: the number of iterations of the EM algorithm
- `verbose`: either TRUE or FALSE; enable or disable printing of progress during the k-means and EM algorithms
- `var_floor`: the variance floor (smallest allowed value) for the diagonal covariances
- `plot_data`: either TRUE or FALSE indicating whether the results of the function should be plotted
- `seed`: integer value for random number generator (RNG)

Details

AIC: the Akaike information criterion
BIC: the Bayesian information criterion

Value

a vector with either the AIC or BIC for each iteration. In case of Error it returns the error message and the possible causes.

Author(s)

Lampros Mouselimis
Examples

data(dietary_survey_IBS)

dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]

dat = center_scale(dat)

opt_gmm = Optimal_Clusters_GMM(dat, 10, criterion = "AIC", plot_data = FALSE)

Optimal_Clusters_KMeans

Optimal number of Clusters for k-means

Description

Optimal number of Clusters for k-means

Usage

Optimal_Clusters_KMeans(data, max_clusters, criterion = "variance_explained",
fk_threshold = 0.85, num_init = 1, max_iters = 200,
initializer = "kmeans++", tol = 1e-04, plot_clusters = TRUE,
verbose = FALSE, tol_optimal_init = 0.3, seed = 1)

Arguments

data matrix or data frame
max_clusters the maximum number of clusters
criterion one of variance_explained, WCSSE, dissimilarity, silhouette, distortion_fK, AIC,
BIC and Adjusted_Rsquared. See details for more information.
fk_threshold a float number used in the 'distortion_fK' criterion
num_init number of times the algorithm will be run with different centroid seeds
max_iters the maximum number of clustering iterations
initializer the method of initialization. One of, optimal_init, quantile_init, kmeans++ and
random. See details for more information
tol a float number. If, in case of an iteration (iteration > 1 and iteration < max_iters)
'tol' is greater than the squared norm of the centroids, then kmeans has con-
verged
plot_clusters either TRUE or FALSE, indicating whether the results of the Optimal_Clusters_KMeans
function should be plotted
verbose either TRUE or FALSE, indicating whether progress is printed during clustering
tol_optimal_init tolerance value for the 'optimal_init' initializer. The higher this value is, the far
apart from each other the centroids are.
seed integer value for random number generator (RNG)
Details

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Explained</td>
<td>The sum of the within-cluster-sum-of-squares-of-all-clusters divided by the</td>
</tr>
<tr>
<td></td>
<td>total sum of squares</td>
</tr>
<tr>
<td>WCSSE</td>
<td>The sum of the within-cluster-sum-of-squares-of-all-clusters</td>
</tr>
<tr>
<td>Dissimilarity</td>
<td>The average intra-cluster-dissimilarity of all clusters (the distance</td>
</tr>
<tr>
<td></td>
<td>metric defaults to euclidean)</td>
</tr>
<tr>
<td>Silhouette</td>
<td>The average silhouette width of all clusters (the distance metric defaults</td>
</tr>
<tr>
<td></td>
<td>to euclidean)</td>
</tr>
<tr>
<td>Distortion_FK</td>
<td>This criterion is based on the following paper, 'Selection of K in K-means</td>
</tr>
<tr>
<td></td>
<td>clustering' (<a href="https://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf">https://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf</a>)</td>
</tr>
<tr>
<td>AIC</td>
<td>The Akaike information criterion</td>
</tr>
<tr>
<td>BIC</td>
<td>The Bayesian information criterion</td>
</tr>
<tr>
<td>Adjusted Rsquared</td>
<td>The adjusted R^2 statistic</td>
</tr>
</tbody>
</table>

Initializers

<table>
<thead>
<tr>
<th>Initializer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal Init</td>
<td>This initializer adds rows of the data incrementally, while checking that</td>
</tr>
<tr>
<td></td>
<td>they do not already exist in the centroid-matrix [experimental]</td>
</tr>
<tr>
<td>Quantile Init</td>
<td>Initialization of centroids by using the cumulative distance between</td>
</tr>
<tr>
<td></td>
<td>observations and by removing potential duplicates [experimental]</td>
</tr>
<tr>
<td></td>
<td>kMeansPP-soda.pdf AND <a href="http://stackoverflow.com/questions/5466323/how-exactly-">http://stackoverflow.com/questions/5466323/how-exactly-</a></td>
</tr>
<tr>
<td></td>
<td>does-k-means-work</td>
</tr>
<tr>
<td>Random</td>
<td>Random selection of data rows as initial centroids</td>
</tr>
</tbody>
</table>

Value

A vector with the results for the specified criterion. If plot_clusters is TRUE then it plots also the results.

Author(s)

Lampros Mouselimis

Examples

data(dietary_survey_IBS)

dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]

dat = center_scale(dat)

opt = Optimal_Clusters_KMeans(dat, max_clusters = 10, plot_clusters = FALSE)
### Optimal_Clusters_Medoids

**Optimal number of Clusters for the partitioning around Medoids functions**

#### Description

Optimal number of Clusters for the partitioning around Medoids functions

#### Usage

```r
Optimal_Clusters_Medoids(data, max_clusters, distance_metric, criterion = "dissimilarity", clara_samples = 0, clara_sample_size = 0, minkowski_p = 1, swap_phase = TRUE, threads = 1, verbose = FALSE, plot_clusters = TRUE, seed = 1)
```

#### Arguments

- **data**: matrix or data.frame. If both clara_samples and clara_sample_size equal 0, then the data parameter can be also a dissimilarity matrix, where the main diagonal equals 0.0 and the number of rows equals the number of columns.
- **max_clusters**: the maximum number of clusters.
- **distance_metric**: a string specifying the distance method. One of, *euclidean*, *manhattan*, *chebychev*, *canberra*, *braycurtis*, *pearson_correlation*, *simple_matching_coefficient*, *minkowski*, *hamming*, *jaccard_coefficient*, *Rao_coefficient*, *mahalanobis*.
- **criterion**: one of 'dissimilarity' or 'silhouette'.
- **clara_samples**: number of samples to draw from the data set in case of clustering large applications (clara).
- **clara_sample_size**: fraction of data to draw in each sample iteration in case of clustering large applications (clara). It should be a float number greater than 0.0 and less or equal to 1.0.
- **minkowski_p**: a numeric value specifying the minkowski parameter in case that distance_metric = "minkowski".
- **swap_phase**: either TRUE or FALSE. If TRUE then both phases ('build' and 'swap') will take place. The 'swap_phase' is considered more computationally intensive.
- **threads**: an integer specifying the number of cores to run in parallel. Openmp will be utilized to parallelize the number of sample draws.
- **verbose**: either TRUE or FALSE, indicating whether progress is printed during clustering.
- **plot_clusters**: TRUE or FALSE, indicating whether the iterative results should be plotted. See the details section for more information.
- **seed**: integer value for random number generator (RNG).
Details

In case of plot_clusters = TRUE, the first plot will be either a plot of dissimilarities or both dissimilarities and silhouette widths giving an indication of the optimal number of the clusters. Then, the user will be asked to give an optimal value for the number of the clusters and after that the second plot will appear with either the dissimilarities or the silhouette widths belonging to each cluster.

Value

a list of length equal to the max_clusters parameter (the first sublist equals NULL, as dissimilarities and silhouette widths can be calculated if the number of clusters > 1). If plot_clusters is TRUE then the function plots also the results.

Author(s)

Lampros Mouselimis

Examples

## Not run:
data(soybean)
dat = soybean[, -ncol(soybean)]
opt_md = Optimal_Clusters_Medoids(dat, 10, 'jaccard_coefficient', plot_clusters = FALSE)
## End(Not run)

plot_2d

2-dimensional plots

Description

2-dimensional plots

Usage

plot_2d(data, clusters, centroids_medoids)

Arguments

data a 2-dimensional matrix or data frame
clusters numeric vector of length equal to the number of rows of the data, which is the result of a clustering method
centroids_medoids a matrix of centroids or medoids. The rows of the centroids_medoids should be equal to the length of the unique values of the clusters and the columns should be equal to the columns of the data.
Details
This function plots the clusters using 2-dimensional data and medoids or centroids.

Value
a plot

Author(s)
Lampros Mouselimis

Examples

```r
# data(dietary_survey_IBS)
# dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
# dat = center_scale(dat)
# pca_dat = stats::princomp(dat)$scores[, 1:2]
# km = KMeans_rcpp(pca_dat, clusters = 2, num_init = 5, max_iters = 100)
# plot_2d(pca_dat, km$clusters, km$centroids)
```

predict_GMM
Prediction function for a Gaussian Mixture Model object

Description
Prediction function for a Gaussian Mixture Model object

Usage
predict_GMM(data, CENTROIDS, COVARIANCE, WEIGHTS)

Arguments
- `data`: matrix or data frame
- `CENTROIDS`: matrix or data frame containing the centroids (means), stored as row vectors
- `COVARIANCE`: matrix or data frame containing the diagonal covariance matrices, stored as row vectors
- `WEIGHTS`: vector containing the weights
predict_KMeans

Details
This function takes the centroids, covariance matrix and weights from a trained model and returns the log-likelihoods, cluster probabilities and cluster labels for new data.

Value
a list consisting of the log-likelihoods, cluster probabilities and cluster labels.

Author(s)
Lampros Mouselimis

Examples

data(dietary_survey_IBS)
dat = as.matrix(dietary_survey_IBS[, -ncol(dietary_survey_IBS)])
dat = center_scale(dat)
gmm = GMM(dat, 2, "maha_dist", "random_subset", 10, 10)
# pr = predict_GMM(dat, gmm$centroids, gmm$covariance_matrices, gmm$weights)

predict_KMeans Prediction function for the k-means

Description
Prediction function for the k-means

Usage
predict_KMeans(data, CENTROIDS)

Arguments
data matrix or data frame
CENTROIDS a matrix of initial cluster centroids. The rows of the CENTROIDS matrix should be equal to the number of clusters and the columns should be equal to the columns of the data.

Details
This function takes the data and the output centroids and returns the clusters.
**predict_MBatchKMeans**

**Value**

a vector (clusters)

**Author(s)**

Lampros Mouselimis

**Examples**

```r
data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = center_scale(dat)
km = KMeans_rcpp(dat, clusters = 2, num_init = 5, max_iter = 100, initializer = 'kmeans++')
pr = predict_KMeans(dat, km$centroids)
```

**predict_MBatchKMeans**  
*Prediction function for Mini-Batch-k-means*

**Description**

Prediction function for Mini-Batch-k-means

**Usage**

```r
predict_MBatchKMeans(data, CENTROIDS, fuzzy = FALSE)
```

**Arguments**

- `data`: matrix or data frame
- `CENTROIDS`: a matrix of initial cluster centroids. The rows of the CENTROIDS matrix should be equal to the number of clusters and the columns should equal the columns of the data.
- `fuzzy`: either TRUE or FALSE. If TRUE then prediction probabilities will be calculated using the distance between observations and centroids.

**Details**

This function takes the data and the output centroids and returns the clusters.

**Value**

if `fuzzy = TRUE` the function returns a list with two attributes: a vector with the clusters and a matrix with cluster probabilities. Otherwise, it returns a vector with the clusters.
**predict_Medoids**

**Author(s)**
Lampros Mouselimis

**Examples**

```r
data(dietary_survey_IBS)
dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]
dat = center_scale(dat)
Mb = MiniBatchKmeans(dat, clusters = 2, batch_size = 20, num_init = 5, early_stop_iter = 10)
pr = predict_MBatchKMeans(dat, Mb$centroids, fuzzy = FALSE)
```

---

**Description**

Predictions for the Medoid functions

**Usage**

```r
predict_Medoids(data, MEDOIDS = NULL, distance_metric = "euclidean", fuzzy = FALSE, minkowski_p = 1, threads = 1)
```

**Arguments**

- **data**: matrix or data frame
- **MEDOIDS**: a matrix of initial cluster medoids (data observations). The rows of the MEDOIDS matrix should be equal to the number of clusters and the columns of the MEDOIDS matrix should be equal to the columns of the data.
- **distance_metric**: a string specifying the distance method. One of, euclidean, manhattan, chebyshev, canberra, braycurtis, pearson_correlation, simple_matching_coefficient, minkowski, hamming, jaccard_coefficient, Rao_coefficient, mahalanobis
- **fuzzy**: either TRUE or FALSE. If TRUE, then probabilities for each cluster will be returned based on the distance between observations and medoids.
- **minkowski_p**: a numeric value specifying the minkowski parameter in case that distance_metric = "minkowski"
- **threads**: an integer specifying the number of cores to run in parallel. Openmp will be utilized to parallelize the number of initializations (num_init)
Silhouette_Dissimilarity_Plot

Value

a list with the following attributes will be returned: clusters, fuzzy_clusters (if fuzzy = TRUE), dissimilarity.

Author(s)

Lampros Mouselimis

Examples

data(dietary_survey_IBS)

dat = dietary_survey_IBS[, -ncol(dietary_survey_IBS)]

dat = center_scale(dat)

cm = Cluster_Medoids(dat, clusters = 3, distance_metric = 'euclidean', swap_phase = TRUE)

pm = predict_Medoids(dat, MEDOIDS = cm$medoids, 'euclidean', fuzzy = TRUE)

Silhouette_Dissimilarity_Plot

Plot of silhouette widths or dissimilarities

Description

Plot of silhouette widths or dissimilarities

Usage

Silhouette_Dissimilarity_Plot(evaluation_object, silhouette = TRUE)

Arguments

evaluation_object

the output of either a Cluster_Medoids or Clara_Medoids function

silhouette

either TRUE or FALSE, indicating whether the silhouette widths or the dissimilarities should be plotted

Details

This function takes the result-object of the Cluster_Medoids or Clara_Medoids function and depending on the argument silhouette it plots either the dissimilarities or the silhouette widths of the observations belonging to each cluster.

Value

TRUE if either the silhouette widths or the dissimilarities are plotted successfully, otherwise FALSE
soybean

Author(s)
Lampros Mouselimis

Examples

# data(soybean)
# dat = soybean[, -ncol(soybean)]
# cm = Cluster_Medoids(dat, clusters = 5, distance_metric = 'jaccard_coefficient')
# plt_sd = Silhouette_Dissimilarity_Plot(cm, silhouette = TRUE)

soybean

The soybean (large) data set from the UCI repository

Description
There are 19 classes, only the first 15 of which have been used in prior work. The folklore seems to be that the last four classes are unjustified by the data since they have so few examples. There are 35 categorical attributes, some nominal and some ordered. The value 'dna' means does not apply. The values for attributes are encoded numerically, with the first value encoded as '0', the second as '1', and so forth. Unknown values were imputed using the mice package.

Usage
data(soybean)

Format
A data frame with 307 Instances and 36 attributes (including the class attribute, "class")

Details
The column names of the data (including the class) appear in the following order:
date, plant-stand, precip, temp, hail, crop-hist, area-damaged, severity, seed-tmt, germination, plant-growth, leaves, leafspots-halo, leafspots-marg, leafspot-size, leaf-shread, leaf-malf, leaf-mild, stem, lodging, stem-cankers, canker-lesion, fruiting-bodies, external decay, mycelium, int-discolor, sclerotia, fruit-pods, fruit spots, seed, mold-growth, seed-discolor, seed-size, shriveling, roots, class
References


Donor: Ming Tan & Jeff Schlimmer (Jeff.Schlimmer cs.cmu.edu)
download source: https://archive.ics.uci.edu/ml/datasets/Soybean+(Large)

Examples

data(soybean)

X = soybean[, -ncol(soybean)]

y = soybean[, ncol(soybean)]
Index

*Topic datasets
  dietary_survey_IBS, 6
  mushroom, 15
  soybean, 27

center_scale, 2
Clara_Medoids, 3
Cluster_Medoids, 5

dietary_survey_IBS, 6
distance_matrix, 7

external_validation, 8

GMM, 9

KMeans arma, 10
KMeans_rcpp, 12

MiniBatchKmeans, 13
mushroom, 15

Optimal_Clusters_GMM, 17
Optimal_Clusters_KMeans, 18
Optimal_Clusters_Medoids, 20

plot_2d, 21
predict_GMM, 22
predict_KMeans, 23
predict_MBatchKMeans, 24
predict_Medoids, 25

Silhouette_Dissimilarity_Plot, 26
soybean, 27