Package ‘PPCI’

May 7, 2026

Type Package

Title Projection Pursuit for Cluster Identification

Version 0.1.5

Author David Hofmeyr <dhofmeyr@sun.ac.za> [aut, cre]
Nicos Pavlidis <n.pavlidis@lancaster.ac.uk> [aut]

Maintainer David Hofmeyr <dhofmeyr@sun.ac.za>

Description Implements recently developed projection pursuit algorithms for finding optimal linear cluster separators. The clustering algorithms use optimal hyperplane separators based on minimum density, Pavlidis et. al (2016) <http://jmlr.org/papers/volume17/15-307/15-307.pdf>; minimum normalised cut, Hofmeyr (2017) <doi:10.1109/TPAMI.2016.2609929>; and maximum variance ratio clusterability, Hofmeyr and Pavlidis (2015) <doi:10.1109/SSCI.2015.116>.

Depends R (>= 2.10.0), rARPACK

License GPL-3

Encoding UTF-8

LazyData yes

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-06 10:10:01 UTC

Contents

PPCI-package 2
breastcancer 3
cluster_performance 4
dermatology 5
mcdc 5
mcdr 7

mch 9
mddc 10
mddr 12
mdh 14
ncutdc 15
ncutdr 17
ncuth 18
optidigits 20
optidigits_mean_images 21
pendigits 21
success_ratio 22
tree_prune 23
tree_split 24
yale 25

Index **26**

`PPCI-package` *Projection Pursuit for Cluster Identification*

Description

This package provides implementations of three recently developed projection pursuit methods for clustering. These methods optimise measures of clusterability of the univariate (projected) dataset that are motivated by three well established approaches to clustering; namely density clustering, centroid based clustering and clustering by graph cuts. The resulting partitions are formed by hyperplanes orthogonal to the optimal projection vectors, and multiple such partitioning hyperplanes are combined in a hierarchical model to generate complete clustering solutions. Model visualisations through low dimensional projections of the data/clusters are provided through multiple plotting functions, which facilitate model validation. Simple model modification functions then allow for pseudo-interactive clustering.

The three main clustering algorithms are implemented in the functions mddc, mcdc and ncutdc. Each takes two mandatory arguments, the data matrix (X) and the number of clusters (K). Numerous optional arguments allow the user to modify the specifics of optimisation, etc. If the correct number of clusters is not known an approximate number can be used, and the resulting solutions visualised using the functions tree_plot (provides a visualisation of the entire model) and node_plot (provides a more detailed visualisation of a single node in the hierarchical model). Both of these cfunctions can be accessed using the base plot function applied to the output of one of mddc, mcdc or ncutdc. Nodes can then be removed using the function tree_prune, or added using the function tree_split, depending on the apparent validity of the existing clustering model.

Details

Package: PPCI

Type: Package

Title: Projection Pursuit for Cluster Identification

Version: 0.1.4

Depends: R (>= 2.10.0), rARPACK

License: GPL-3

LazyData: yes

Author(s)

David Hofmeyr[aut, cre] and Nicos Pavlidis[aut]

Maintainer: David Hofmeyr <dhofmeyr@sun.ac.za>

References

Pavlidis N.G., Hofmeyr D.P., Tasoulis S.K. (2016) Minimum Density Hyperplanes. *Journal of Machine Learning Research*, **17**(156), 1–33.

Hofmeyr, D., Pavlidis, N. (2015) Maximum Clusterability Divisive Clustering. *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 780–786.

Hofmeyr, D. (2017) Clustering by Minimum Cut Hyperplanes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(8), 1547 – 1560.

See Also

A **MATLAB** toolbox with similar functionality to this package is available at https://github.com/nicospavlidis/opc. Outputs may differ slightly due to differences between **R** and **MATLAB**'s base optimisation software.

`breastcancer` *Discrimination of Cancerous and Non-Cancerous Breast Masses*

Description

This data set contains case information from 699 patients from Wisconsin General Hospital who received examination for mammographic masses, which were classified as either benign or malignant.

Usage

```
breastcancer
```

Format

A list with entries $x (a 699x9 matrix with each row corresponding to an individual case) and $c (a vector of labels indicating whether the mass was diagnosed as benign (2) or malignant (4)).

Source

UCI Machine Learning Repository.

References

Lichman, M. (2013) UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. [https://archive.ics.uci.edu/ml]

`cluster_performance` *External Cluster Validity Metrics*

Description

Computes four popular external cluster validity metrics (adjusted Rand index, purity, V-measure and Normalised Mutual Information) through comparison of cluster assignments and true class labels.

Usage

```
cluster_performance(assigned, labels, beta)
```

Arguments

`assigned`	a vector of cluster assignments made by a clustering algorithm.
`labels`	a vector of true class labels to be compared with assigned.
`beta`	(optional) positive numeric, used in the computation of V-measure. larger values apply higher weight to homogeneity over completeness measures. if omitted then beta = 1 (equal weight applied to both measures).

Value

a vector containing the four evaluation metrics listed in the description.

References

Zhao Y., Karypis G. (2004) Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. *Machine Learning*, **55**(3), 311–331.

Strehl A., Ghosh J. (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, **3**, 583–617.

Rosenberg A., Hirschberg J. (2007) V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. *EMNLP-CoNLL*, **7**, 410–420. Citeseer.

Hubert, L., Arabie, P. (1985) Comparing Partitions. *Journal of Classification*, **2**(1), 193–218.

Examples

```
## load dermatology dataset
data(dermatology)

## obtain clustering solution using MCDC
sol <- mcdc(dermatology$x, 6)

## evaluate solution using external cluster validity measures
cluster_performance(sol$cluster, dermatology$c)
```

dermatology — *Eryhemato-Squamous Disease Identification*

Description

This data set contains clinical and histopathological information from 366 cases of 6 different skin disorders/diseases for the purpous of differential diagnosis of eryhemato-squamous disease.

Usage

```
dermatology
```

Format

A list with entries $x (a 366x34 matrix with each row corresponding to an individual case) and $c (a vector of labels indicating the skin condition).

Source

UCI Machine Learning Repository.

References

Lichman, M. (2013) UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. https://archive.ics.uci.edu/ml

mcdc — *Divisive Clustering Using Maximum Clusterability*

Description

Generates a binary partitioning tree by recursively partitioning a dataset using a hierarchical collection of hyperplanes with high variance ratio custerability across them.

Usage

```
mcdc(X, K, v0, split.index, minsize, verb, labels, maxit, ftol)
```

Arguments

`X`	a numeric matrix (num_data x num_dimensions); the dataset to be clustered.
`K`	the number of clusters to extract.
`split.index`	(optional) determines the order in which clusters are split (in decreasing order of split indices). can be a numeric valued function(v, X, P) of projection vector v, data matrix X and list of parameters P. can also be one of "size" (split the largest cluster), "fval" (split the cluster with the maximum variance ratio) or "Fdist" (split indices determined by the non-central F-distribution. See SSCI paper for details. slight difference from the paper is that when the data size is above 2000 cluster size is used instead. This is because the naive estimation of the model degrees of freedom has been found to be unreliable when the number of data is large). if omitted then "Fdist" is used.
`v0`	(optional) initial projection direction(s). a function(X) of the data being split, which returns a matrix with ncol(X) rows. each column of the output of v0(X) is used as an initialisation for projection pursuit. the solution with the maximum variance ratio is used within the final model. initialisations are determined separately for each cluster being split. if omitted then a single initialisation is used; the vector joining the cluster means of a 2-means solution.
`minsize`	(optional) the minimum cluster size allowable. if omitted then minsize = 1.
`verb`	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
`labels`	(optional) vector of class labels. not used in the actual clustering procedure. only used for illustrative purposes for values of verb>0.
`maxit`	(optional) maximum number of iterations in optimisation. if omitted then maxit=50.
`ftol`	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-8.

Value

a named list with class ppci_cluster_solution containing

`$cluster`	cluster assignment vector.
`$model`	matrix containing the would-be location of each node (depth and position at depth) within a complete tree of appropriate depth.
`$nodes`	unnamed list each element of which is a named list containing details of the binary partitions at each node in the model.
`$data`	the data matrix being clustered.
`$method`	=="MCDC". used in plotting and model modification functions.
`$args`	named list of arguments passed to mcdc.

References

Hofmeyr, D., Pavlidis, N. (2015) Maximum Clusterability Divisive Clustering. *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 780–786.

Examples

```
## load the dermatology dataset
data(dermatology)

## obtain a clustering solution using MCDC
sol <- mcdc(dermatology$x, 6)

## evaluate the performance of the solution using external cluster validity metrics
cluster_performance(sol$cluster, dermatology$c)
```

`mcdr` *Maximum Clusterability Dimension Reduction*

Description

Finds a linear projection of a data set using projection pursuit to maximise the variance ratio clusterability measured in each dimension separately.

Usage

```
mcdr(X, p, minsize, v0, verb, labels, maxit, ftol)
```

Arguments

`X`	a numeric matrix (num_data x num_dimensions); the dataset.
`p`	an integer; the number of dimensions in the projection.
`v0`	(optional) initial projection direction(s). a function(X) of the data, which returns a matrix with ncol(X) rows. each column of the output of v0(X) is used as an initialisation for projection pursuit. the solution with the minimum normalised cut is used within the final model. if omitted then a single initialisation is used for each column of the projection matrix; the first principal component within the null space of the other columns.
`verb`	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
`labels`	(optional) vector of class labels. not used in the actual projection pursuit. only used for illustrative purposes for values of verb>0.

`maxit`	(optional) maximum number of iterations in optimisation for each value of alpha. if omitted then maxit=15.
`ftol`	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-5.
`minsize`	(optional) the minimum number of data on each side of a hyperplane. if omitted then minsize = 1.

Value

a named list with class ppci_projection_solution with the following components

`$projection`	the num_dimensions x p projection matrix.
`$fitted`	the num_data x p projected data set.
`$data`	the input data matrix.
`$method`	=="MCDC".
`$params`	list of parameters used to find $projection.

References

Hofmeyr, D., Pavlidis, N. (2015) Maximum Clusterability Divisive Clustering. *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 780–786.

Examples

```
### not run
run = FALSE
if(run){
  ## load optidigits dataset
  data(optidigits)

  ## find nine dimensional projection (one fewer than
  ## the number of clusters, as is common in clustering)
  sol <- mcdr(optidigits$x, 9)

  ## visualise the solution via the first 3 pairs of dimensions
  plot(sol, pairs = 3, labels = optidigits$c)

  ## compare with PCA projection
  pairs(optidigits$x%*%eigen(cov(optidigits$x))$vectors[,1:3], col = optidigits$c)
  }
```

mch *Maximum Clusteriability Hyperplane*

Description

Finds maximum clusterability hyperplane(s) for clustering.

Usage

```
mch(X, v0, minsize, verb, labels, maxit, ftol)
```

Arguments

X	a numeric matrix (num_data x num_dimensions); the dataset to be clustered.
v0	(optional) initial projection direction(s). a matrix with ncol(X) rows. each column of v0 is used as an initialisation for projection pursuit. if omitted then a single initialisation is used; the vector joining the cluster means from a 2-means solution.
minsize	(optional) the minimum cluster size allowable. if omitted then minsize = 1.
verb	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
labels	(optional) vector of class labels. not used in the actual clustering procedure. only used for illustrative purposes for values of verb>0.
maxit	(optional) maximum number of iterations in optimisation. if omitted then maxit=50.
ftol	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-8.

Value

a named list with class ppci_hyperplane_solution with the following components

$cluster	cluster assignment vector.
$v	the optimal projection vector.
$b	the value of b making H(v, b) the minimum normalised cut hyperplane.
$fitted	data projected into two dimensional subspace defined by $v and the principal component in the null space of $v.
$data	the input data matrix.
$fval	the variance ratio clusterability across H(v, b).
$method	=="MCDC".
$params	list of parameters used to find H(v, b).
$alternatives	an unnamed list. If more than one initilisation is considered, the alternatives to the best are stored in this field.

References

Hofmeyr, D., Pavlidis, N. (2015) Maximum Clusterability Divisive Clustering. *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 780–786.

Examples

```
## generate dataset with elongated clusters for which variance ratio in
## both dimensions is misleading for clustering
set.seed(1)
S <- matrix(c(1, .7, .7, 1), 2, 2)
X <- matrix(rnorm(2000), ncol = 2)%*%S
X[,1] <- X[,1] + rep(c(.8, -.8), each = 500)
X[,2] <- X[,2] + rep(c(-.8, .8), each = 500)

## find the optimal variance ratio hyperplane solution
sol <- mch(X)

## visualise the solution
plot(X, col = sol$cluster)
```

`mddc` *Divisive Clustering Using Minimum Density Hyperplanes*

Description

Generates a binary partitioning tree by recursively partitioning a dataset using a hierarchical collection of hyperplanes with low empirical density integral.

Usage

```
mddc(X, K, minsize, split.index, v0, bandwidth,
      alphamin, alphamax, verb, labels, maxit, ftol)
```

Arguments

`X`	a numeric matrix (num_data x num_dimensions); the dataset to be clustered.
`K`	the number of clusters to extract.
`minsize`	(optional) minimum cluster size. if omitted then minsize = 1.
`split.index`	(optional) determines the order in which clusters are split (in decreasing order of split indices). can be a numeric valued function(v, X, P) of projection vector v, data matrix X and list of parameters P. can also be one of "size" (split the largest cluster), "fval" (split the cluster with the minimum density hyperplane) or "rdepth" (split the cluster with the maximum relative depth). if omitted then "size" is used.

`v0` (optional) initial projection direction(s). a function(X) of the data being split, which returns a matrix with ncol(X) rows. each column of the output of v0(X) is used as an initialisation for projection pursuit. the solution with the greatest relative depthis used within the final model. initialisations are determined separately for each cluster being split. if omitted then a single initialisation is used; the first principal component.

`bandwidth` (optional) used to compute the bandwidth parameter (h) for the kernel density estimator. a numeric valued function(X) of the cluster being split. if omitted then bandwidth(X) = 0.9*eigen(cov(X))$values[1]^.5*nrow(X)^(-0.2).

`alphamin` (optional) initial (scaled) bound on the distance of the optimal hyperplane from the mean of the data (or subset being split). if omitted then alphamin = 0.

`alphamax` (optional) maximum (scaled) distance of the optimal hyperplane from the mean of the data (or subset being split). if omitted then alphamax = 1.

`verb` (optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.

`labels` (optional) vector of class labels. not used in the actual clustering procedure. only used for illustrative purposes for values of verb>0.

`maxit` (optional) maximum number of iterations in optimisation for each value of alpha. if omitted then maxit=15.

`ftol` (optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-5.

Value

a named list with class ppci_cluster_solution containing

`$cluster` cluster assignment vector.

`$model` matrix containing the would-be location of each node (depth and position at depth) within a complete tree of appropriate depth.

`$nodes` unnamed list each element of which is a named list containing details of the binary partitions at each node in the model.

`$data` the data matrix being clustered.

`$method` =="MDH". used in plotting and model modification functions.

`$args` named list of arguments passed to mddc.

References

Pavlidis N.G., Hofmeyr D.P., Tasoulis S.K. (2016) Minimum Density Hyperplanes. *Journal of Machine Learning Research*, **17**(156), 1–33.

Examples

```
## load dermatology dataset
data(dermatology)

## obtain a clustering solution using minimum density hyperplanes
sol <- mddc(dermatology$x, 6)

## evaluate the solution using external cluster validity metrics
cluster_performance(sol$cluster, dermatology$c)
```

---

mddr *Minimum Density Dimension Reduction*

Description

Finds a linear projection of a data set using projection pursuit to find the vector(s) orthogonal to minimum density hyperplanes.

Usage

```
mddr(X, p, minsize, v0, bandwidth,
      alphamin, alphamax, verb, labels, maxit, ftol)
```

Arguments

X	a numeric matrix (num_data x num_dimensions); the dataset.
p	an integer; the number of dimensions in the projection.
v0	(optional) initial projection direction(s). a function(X) of the data, which returns a matrix with ncol(X) rows. each column of the output of v0(X) is used as an initialisation for projection pursuit. the solution with the minimum normalised cut is used within the final model. if omitted then a single initialisation is used for each column of the projection matrix; the first principal component within the null space of the other columns.
bandwidth	(optional) used to compute the bandwidth parameter (h) for the kernel density estimator. a numeric valued function(X) of the cluster being split. if omitted then bandwidth(X) = 0.9*eigen(cov(X))$values[1]^.5*nrow(X)^(-0.2).
alphamin	(optional) initial (scaled) bound on the distance of the optimal hyperplane from the mean of the data. if omitted then alphamin = 0.
alphamax	(optional) maximum (scaled) distance of the optimal hyperplane from the mean of the data. if omitted then alphamax = 1.
verb	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.

`labels`	(optional) vector of class labels. not used in the actual projection pursuit. only used for illustrative purposes for values of verb>0.
`maxit`	(optional) maximum number of iterations in optimisation for each value of alpha. if omitted then maxit=15.
`ftol`	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-5.
`minsize`	(optional) the minimum number of data on each side of a hyperplane. if omitted then minsize = 1.

Value

a named list with class ppci_projection_solution with the following components

`$projection`	the num_dimensions x p projection matrix.
`$fitted`	the num_data x p projected data set.
`$data`	the input data matrix.
`$method`	=="MDH".
`$params`	list of parameters used to find $projection.

References

Pavlidis N.G., Hofmeyr D.P., Tasoulis S.K. (2016) Minimum Density Hyperplanes. *Journal of Machine Learning Research*, **17**(156), 1–33.

Examples

```
### not run
run = FALSE
if(run){
  ## load optidigits dataset
  data(optidigits)

  ## find nine dimensional projection (one fewer than
  ## the number of clusters, as is common in clustering)
  sol <- mddr(optidigits$x, 9)

  ## visualise the solution via the first 3 pairs of dimensions
  plot(sol, pairs = 3, labels = optidigits$c)

  ## compare with PCA projection
  pairs(optidigits$x%*%eigen(cov(optidigits$x))$vectors[,1:3], col = optidigits$c)
  }
```

mdh *Minimum Density Hyperplane*

Description

Finds minimum density hyperplane(s) for clustering.

Usage

```
mdh(X, v0, minsize, bandwidth, alphamin, alphamax, verb, labels, maxit, ftol)
```

Arguments

`X`	a numeric matrix (num_data x num_dimensions); the dataset to be clustered.
`v0`	(optional) initial projection direction(s). a matrix with ncol(X) rows. each column of v0 is used as an initialisation for projection pursuit. if omitted then a single initialisation is used; the first principal component.
`minsize`	(optional) minimum cluster size. if omitted then minsize = 1.
`bandwidth`	(optional) positive numeric bandwidth parameter (h) for the kernel density estimator. if omitted then bandwidth = 0.9*eigen(cov(X))$values[1]^.5*nrow(X)^(-0.2).
`alphamin`	(optional) initial (scaled) bound on the distance of the optimal hyperplane from the mean of the data. if omitted then alphamin = 0.
`alphamax`	(optional) maximum/final (scaled) distance of the optimal hyperplane from the mean of the data. if omitted then alphamax = 1.
`verb`	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
`labels`	(optional) vector of class labels. not used in the actual clustering procedure. only used for illustrative purposes for values of verb>0.
`maxit`	(optional) maximum number of iterations in optimisation for each value of alpha. if omitted then maxit=15.
`ftol`	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-5.

Value

a named list with class ppci_hyperplane_solution with the following components

`$cluster`	cluster assignment vector.
`$v`	the optimal projection vector.
`$b`	the value of b making H(v, b) the minimum density hyperplane.

`$fitted`	data projected into two dimensional subspace defined by $v and the principal component in the null space of $v.
`$data`	the input data matrix.
`$rel.dep`	the relative depth of H(v, b).
`$fval`	the integrated dentsity on H(v, b).
`$method`	=="MDH".
`$params`	list of parameters used to find H(v, b).
`$alternatives`	an unnamed list. If more than one initilisation is considered, the alternatives to the best are stored in this field.

References

Pavlidis N.G., Hofmeyr D.P., Tasoulis S.K. (2016) Minimum Density Hyperplanes. *Journal of Machine Learning Research*, **17**(156), 1–33.

Examples

```
## load breast cancer dataset
data(breastcancer)

## find minimum density hyperplane
sol <- mdh(breastcancer$x)

## visualise the solution
plot(sol)

## evaluate the quality of the partition
success_ratio(sol$cluster, breastcancer$c)
```

`ncutdc` *Divisive Clustering Using Minimum Normalised Cut Hyperplanes*

Description

Generates a binary partitioning tree by recursively partitioning a dataset using a hierarchical collection of hyperplanes with low normalised cut measured across them.

Usage

```
ncutdc(X, K, split.index, v0, s, minsize, verb, labels, maxit, ftol)
```

Arguments

`X`	a numeric matrix (num_data x num_dimensions); the dataset to be clustered.
`K`	the number of clusters to extract.
`split.index`	(optional) determines the order in which clusters are split (in decreasing order of split indices). can be a numeric valued function(v, X, P) of projection vector v, data matrix X and list of parameters P. can also be one of "size" (split the largest cluster) or "fval" (split the cluster with the minimum density hyperplane). if omitted then "fval" is used.
`v0`	(optional) initial projection direction(s). a function(X) of the data being split, which returns a matrix with ncol(X) rows. each column of the output of v0(X) is used as an initialisation for projection pursuit. the solution with the minimum normalised cut is used within the final model. initialisations are determined separately for each cluster being split. if omitted then a single initialisation is used; the first principal component.
`s`	(optional) used to compute the scaling parameter (sigma) for pairwise similarities. a numeric valued function(X) of the cluster being split. if omitted then s(X) = 100*eigen(cov(X))$values[1]^.5*nrow(X)^(-0.2).
`minsize`	(optional) the minimum cluster size allowable. if omitted then minsize = 1.
`verb`	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
`labels`	(optional) vector of class labels. not used in the actual clustering procedure. only used for illustrative purposes for values of verb>0.
`maxit`	(optional) maximum number of iterations in optimisation. if omitted then maxit=50.
`ftol`	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-8.

Value

a named list with class ppci_cluster_solution containing

`$cluster`	cluster assignment vector.
`$model`	matrix containing the would-be location of each node (depth and position at depth) within a complete tree of appropriate depth.
`$nodes`	unnamed list each element of which is a named list containing details of the binary partitions at each node in the model.
`$data`	the data matrix being clustered.
`$method`	=="NCutH". used in plotting and model modification functions.
`$args`	named list of arguments passed to ncutdc.

References

Hofmeyr, D. (2016) Clustering by Minimum Cut Hyperplanes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(8), 1547 – 1560.

Examples

```
## load dermatology dataset
data(dermatology)

## obtain clustering solution
sol <- ncutdc(dermatology$x, 6)

## evaluate solution using external cluster validity metrics
cluster_performance(sol$cluster, dermatology$c)
```

---

ncutdr *Minimum Normalised Cut Dimension Reduction*

Description

Finds a linear projection of a data set using projection pursuit based on minimising the normalised cut measured in each dimension separately.

Usage

```
ncutdr(X, p, v0, s, minsize, verb, labels, maxit, ftol)
```

Arguments

X	a numeric matrix (num_data x num_dimensions); the dataset.
p	an integer; the number of dimensions in the projection.
v0	(optional) initial projection direction(s). a function(X) of the data, which returns a matrix with ncol(X) rows. each column of the output of v0(X) is used as an initialisation for projection pursuit. the solution with the minimum normalised cut is used within the final model. if omitted then a single initialisation is used for each column of the projection matrix; the first principal component within the null space of the other columns.
s	(optional) used to compute the scaling parameter (sigma) for pairwise similarities. a numeric valued function(X) of the data. if omitted then s(X) = 100*eigen(cov(X))$values[1]^.5*nrow(X)^(-0.2).
minsize	(optional) the minimum cluster size allowable when computing the normalised cut. if omitted then minsize = 1.
verb	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
labels	(optional) vector of class labels. not used in the actual projection pursuit. only used for illustrative purposes for values of verb>0.
maxit	(optional) maximum number of iterations in optimisation. if omitted then maxit=50.
ftol	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-8.

Value

a named list with class ppci_projection_solution with the following components

`$projection`	the num_dimensions x p projection matrix.
`$fitted`	the num_data x p projected data set.
`$data`	the input data matrix.
`$method`	=="NCutH".
`$params`	list of parameters used to find $projection.

References

Hofmeyr, D. (2016) Clustering by Minimum Cut Hyperplanes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(8), 1547 – 1560.

Examples

```
### not run
run = FALSE
if(run){
  ## load optidigits dataset
  data(optidigits)

  ## find nine dimensional projection (one fewer than
  ## the number of clusters, as is common in clustering)
  sol <- ncutdr(optidigits$x, 9)

  ## visualise the solution via the first 3 pairs of dimensions
  plot(sol, pairs = 3, labels = optidigits$c)

  ## compare with PCA projection
  pairs(optidigits$x%*%eigen(cov(optidigits$x))$vectors[,1:3], col = optidigits$c)
  }
```

`ncuth` *Minimum Normalised Cut Hyperplane*

Description

Finds minimum normalised cut hyperplane(s) for clustering.

Usage

```
ncuth(X, v0, s, minsize, verb, labels, maxit, ftol)
```

Arguments

`X`	a numeric matrix (num_data x num_dimensions); the dataset to be clustered.
`v0`	(optional) initial projection direction(s). a matrix with ncol(X) rows. each column of v0 is used as an initialisation for projection pursuit. if omitted then a single initialisation is used; the first principal component.
`s`	(optional) positive numeric scaling parameter (sigma). if omitted then s = 100*eigen(cov(X))$values[1]^.5 0.2).
`minsize`	(optional) the minimum cluster size allowable. if omitted then minsize = 1.
`verb`	(optional) verbosity level of optimisation procedure. verb==0 produces no output. verb==1 produces plots illustrating the progress of projection pursuit via plots of the projected data. verb==2 adds to these plots additional information about the progress. verb==3 creates a folder in working directory and stores all plots for verb==2. if omitted then verb==0.
`labels`	(optional) vector of class labels. not used in the actual clustering procedure. only used for illustrative purposes for values of verb>0.
`maxit`	(optional) maximum number of iterations in optimisation. if omitted then maxit=50.
`ftol`	(optional) tolerance level for convergence of optimisation, based on relative function value improvements. if omitted then ftol = 1e-8.

Value

a named list with class ppci_hyperplane_solution with the following components

`$cluster`	cluster assignment vector.
`$v`	the optimal projection vector.
`$b`	the value of b making H(v, b) the minimum normalised cut hyperplane.
`$fitted`	data projected into two dimensional subspace defined by $v and the principal component in the null space of $v.
`$data`	the input data matrix.
`$fval`	the normalised cut across H(v, b).
`$method`	=="NCutH".
`$params`	list of parameters used to find H(v, b).
`$alternatives`	an unnamed list. If more than one initilisation is considered, the alternatives to the best are stored in this field.

References

Hofmeyr, D. (2016) Clustering by Minimum Cut Hyperplanes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(8), 1547 – 1560.

Examples

```
## load breast cancer dataset
data(breastcancer)

## find minimum normalised cut hyperplane
sol <- ncuth(breastcancer$x)

## visualise the solution
plot(sol)

## evaluate the performance of the solution
success_ratio(sol$cluster, breastcancer$c)
```

`optidigits` *Optical Recognition of Handwritten Digits*

Description

This data set contains vectorised images of handwritten digits (0–9), compressed to 8x8 pixels.

Usage

```
optidigits
```

Format

A list with entries $x (a 5620x64 matrix with each row corresponding to an image) and $c (a vector of labels indicating the written digit).

Source

UCI Machine Learning Repository.

References

Lichman, M. (2013) UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. https://archive.ics.uci.edu/ml

optidigits_mean_images

Visualise Cluster Means from optidigits data set

Description

Provides a visualisation of the cluster means computed from the optidigits data set, recast as images. Cluster labels are aligned with the true labels using simulated annealing to maximise the trace of the confusion matrix (or subset if number of clusters != number of classes.)

Usage

```
optidigits_mean_images(clusters)
```

Arguments

clusters	a vector of cluster assignments. Must take values in 1:k, where k is the number of clusters.

References

Lichman, M. (2013) UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. https://archive.ics.uci.edu/ml

Examples

```
### not run
run = FALSE
if(run){
  ## load optidigits dataset
  data(optidigits)

  ## obtain a clustering solution using normalised cut hyperplanes
  sol <- ncutdc(optidigits$x, 10)

  ## visualise the cluster means as images
  optidigits_mean_images(sol$cluster)
  }
```

pendigits

Pen-based Recognition of Handwritten Digits

Description

This data set contains features derived from pen trajectories arising from handwritten digits (0–9) from 44 subjects.

Usage

```
optidigits
```

Format

A list with entries $x (a 10992x16 matrix with each row corresponding to a pen trajectory) and $c (a vector of labels indicating the written digit).

Source

UCI Machine Learning Repository.

References

Lichman, M. (2013) UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. https://archive.ics.uci.edu/ml

---

success_ratio *Evaluate External Valifity os a Binary Partition*

Description

Computes the success ratio of a binary partition by comparing the solution with true class labels.

Usage

```
success_ratio(assigned, labels)
```

Arguments

assigned	a vector of cluster assignments made by a clustering algorithm.
labels	a vector of true class labels to be compared with assigned.

Value

the success ratio of the cluster assignment solution.

References

Hofmeyr, D. (2016) Clustering by Minimum Cut Hyperplanes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Examples

```
## load optidigits dataset
data(optidigits)

## generate a binary partition using minimum normalised cut hyperplane
sol <- ncuth(optidigits$x)

## evaluate using success ratio
success_ratio(sol$cluster, optidigits$c)
```

---

tree_prune *Prune a Hierarchical Clustering Model*

Description

Removes the subtree rooted at the specified node from a hierarchical clustering model generated by one of mcdc, mddc and ncutdc.

Usage

```
tree_prune(sol, node)
```

Arguments

sol	a clustering solution arising from one of the functions mcdc, mddc and ncutdc.
node	the node at which to prune the hierarchy. can be either an integer specifying the node number in sol$nodes or a vector of length two specifying c(depth, position at depth) of the node.

Value

a list with the same components as sol.

Examples

```
## load the optidigits dataset
data(optidigits)

## cluster using minimum normalised cut hyperplanes,
## assuming no domain knowledge begin with 12 clusters
sol <- ncutdc(optidigits$x, 12)

## the node numbered 11 has been split,
## yet it appears there may not be multiple clusters present.
## inspect this node more closely
plot(sol, node = 11)

## remove this node from the model
sol_new <- tree_prune(sol, 11)
```

```
## compare the solutions using external cluster validity metrics

cluster_performance(sol$cluster, optidigits$c)

cluster_performance(sol_new$cluster, optidigits$c)
```

`tree_split` *Split a Leaf in a Hierarchical Clustering Model*

Description

Adds an additional binary partition to an existing hierarchical clustering model produced by one of mcdc, mddc and ncutdc.

Usage

```
tree_split(sol, node, ...)
```

Arguments

`sol`	a clustering solution arising from one of the functions mcdc, mddc and ncutdc.
`node`	the node to be further partitioned. can be either an integer specifying the node number in sol$nodes or a vector of length two specifying c(depth, position at depth) of the node.
`...`	any modifications to parameters used in optimisation. these should have the same names and types as the corresponding arguments for the method used to construct sol.

Value

a list with the same components as sol. the $args field will reflect any changes included in ... above.

Examples

```
## load the optidigits dataset
data(optidigits)

## cluster using minimum normalised cut hyperplanes,
## assuming no domain knowledge begin with 8 clusters
sol <- ncutdc(optidigits$x, 8)

## visualise solution
plot(sol)

## node 13 shows evidence of multiple clusters. Inspect this node more closely
plot(sol, node = 13)

## split node 13
```

```
sol_new <- tree_split(sol, 13)

## compare the solutions using external cluster validity metrics
cluster_performance(sol$cluster, optidigits$c)

cluster_performance(sol_new$cluster, optidigits$c)
```

`yale` *Face Recognition*

Description

This data set contains vectorised images of the faces of 10 different human subjects with different poses and lighting conditions. The images were compressed to 30x20 pixels.

Usage

```
yale
```

Format

A list with entries $x (a 2000x600 matrix with each row corresponding to an image) and $y (a vector of labels indicating the human subject).

Source

Yale Faces Database B. Compressed images (30x40) available from [https://cervisia.org/machine_learning_data.php/]. Further compression was performed by the package developers. In addition only the first 200 images of each subject are included.

References

Georghiades, A.S. and Belhumeur, P.N. and Kriegman, D.J. (2001) From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**(6) pp. 643–660.

Index

∗ **datasets**
- breastcancer, 3
- dermatology, 5
- optidigits, 20
- pendigits, 21
- yale, 25

∗ **file**
- cluster_performance, 4
- mcdc, 5
- mcdr, 7
- mch, 9
- mddc, 10
- mddr, 12
- mdh, 14
- ncutdc, 15
- ncutdr, 17
- ncuth, 18
- optidigits_mean_images, 21
- success_ratio, 22
- tree_prune, 23
- tree_split, 24

∗ **package**
- PPCI-package, 2

breastcancer, 3

cluster_performance, 4

dermatology, 5

mcdc, 5
mcdr, 7
mch, 9
mddc, 10
mddr, 12
mdh, 14

ncutdc, 15
ncutdr, 17
ncuth, 18

optidigits, 20
optidigits_mean_images, 21

pendigits, 21
PPCI *(PPCI-package)*, 2
PPCI-package, 2

success_ratio, 22

tree_prune, 23
tree_split, 24

yale, 25