Package ‘sparcl’

October 14, 2022

Type Package

Title Perform Sparse Hierarchical Clustering and Sparse K-Means Clustering

Version 1.0.4

Date 2018-10-22

Author Daniela M. Witten and Robert Tibshirani

Maintainer Daniela Witten <dwitten@u.washington.edu>

Description Implements the sparse clustering methods of Witten and Tibshirani (2010): ``A framework for feature selection in clustering''; published in Journal of the American Statistical Association 105(490): 713-726.

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2018-10-24 13:50:03 UTC

NeedsCompilation yes

R topics documented:

`sparcl-package` *Performs sparse hierarchical and sparse K-means clustering*

Description

Implements the sparse clustering methods of Witten and Tibshirani (2010) "A framework for feature selection in clustering", Journal Amer. Stat. Assocn. 105(490): 713-726.

Details

Package:	sparcl
Type:	Package
Version:	1.0.3
Date:	2013-1-02
License:	GPL-2
LazyLoad:	yes

The main functions are KMeansSparseCluster and HierarchicalSparseCluster. Tuning parameters for these methods are chosen by KMeansSparseCluster.permute and HierarchicalSparseCluster.permute.

Author(s)

Daniela M. Witten and Robert Tibshirani

Maintainer: Daniela Witten <dwitten@u.washington.edu>

References

Witten and Tibshirani (2010) A framework for feature selection in clustering. Journal Amer. Stat. Assocn. 105(490): 713-726.

`ColorDendrogram` *Color the leaves in a hierarchical clustering dendrogram*

Description

Pass in the output of "hclust" and a class label for each observation. A colored dendrogram will result, with the leaf colors indicating the classes.

Usage

```
ColorDendrogram(hc, y, main = "", branchlength = 0.7, labels = NULL, xlab = NULL,
sub="NULL",ylab = "", cex.main = NULL)
```

Arguments

hc	The output of running "hclust" on a nxn dissimilarity matrix
y	A vector of n class labels for the observations that were clustered using "hclust". If labels are numeric from 1 to K, then colors will be determine automatically. Otherwise the labels can take the form of colors (e.g. c("red", "red", "orange", "orange")).
main	The main title for the dendrogram.
branchlength	How long to make the colored part of the branches. Adjustment will be needed for each dissimilarity matrix
labels	The labels for the n observations.
xlab	X-axis label.
sub	Sub-x-axis label.
ylab	Y-axis label.
cex.main	The amount by which to enlarge the main title for the figure.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten and Tibshirani (2009) A framework for feature selection in clustering.

See Also

HierarchicalSparseCluster, HierarchicalSparseCluster.permute

Examples

```
# Generate 2-class data
set.seed(1)
x <- matrix(rnorm(100*20),ncol=20)
y <- c(rep(1,50),rep(2,50))
x[y==1,] <- x[y==1,]+2
# Perform hierarchical clustering
hc <- hclust(dist(x),method="complete")
# Plot
ColorDendrogram(hc,y=y,main="My Simulated Data",branchlength=3)
```

HierarchicalSparseCluster
Hierarchical sparse clustering

Description

Performs sparse hierarchical clustering. If $d_ii'j$ is the dissimilarity between observations i and i' for feature j, seek a sparse weight vector w and then use $(sum_j (d_ii'j w_j))_ii'$ as a nxn dissimilarity matrix for hierarchical clustering.

Usage

```
HierarchicalSparseCluster(x=NULL, dists=NULL,
method=c("average","complete", "single","centroid"),
wbound=NULL,niter=15,dissimilarity=c("squared.distance","absolute.value"),
 uorth=NULL,
silent=FALSE,cluster.features=FALSE,method.features=c("average", "complete",
"single","centroid"),output.cluster.files=FALSE,
outputfile.prefix="output",genenames=NULL,genedesc=NULL,standardize.arrays=FALSE)
## S3 method for class 'HierarchicalSparseCluster'
print(x,...)
## S3 method for class 'HierarchicalSparseCluster'
plot(x,...)
```

Arguments

`x`	A nxp data matrix; n is the number of observations and p the number of features. If NULL, then specify dists instead.
`dists`	For advanced users, can be entered instead of x. If HierarchicalSparseCluster has already been run on this data, then the dists value of the previous output can be entered here. Under normal circumstances, leave this argument NULL and pass in x instead.
`method`	The type of linkage to use in the hierarchical clustering - "single", "complete", "centroid", or "average".
`wbound`	The L1 bound on w to use; this is the tuning parameter for sparse hierarchical clustering. Should be greater than 1.
`niter`	The number of iterations to perform in the sparse hierarchical clustering algorithm.
`dissimilarity`	The type of dissimilarity measure to use. One of "squared.distance" or "absolute.value". Only use this if x was passed in (rather than dists).
`uorth`	If complementary sparse clustering is desired, then this is the nxn dissimilarity matrix obtained in the original sparse clustering.
`standardize.arrays`	Should the arrays be standardized? Default is FALSE.
`silent`	Print out progress?

`cluster.features`
: Not for use.

`method.features`
: Not for use.

`output.cluster.files`
: Not for use.

`outputfile.prefix`
: Not for use.

`genenames` Not for use.

`genedesc` Not for use.

`...` not used.

Details

We seek a p-vector of weights w (one per feature) and a nxn matrix U that optimize

$maximize_U,w sum_j w_j sum_ii' d_ii'j U_ii'$ subject to $||w||_2 <= 1, ||w||_1 <= wbound, w_j >= 0, sum_ii' U_ii'^2 <= 1$.

Here, $d_ii'j$ is the dissimilarity between observations i and i' with along feature j. The resulting matrix U is used as a dissimilarity matrix for hierarchical clustering. "wbound" is a tuning parameter for this method, which controls the L1 bound on w, and as a result the number of features with non-zero w_j weights. The non-zero elements of w indicate features that are used in the sparse clustering.

We optimize the above criterion with an iterative approach: hold U fixed and optimize with respect to w. Then, hold w fixed and optimize with respect to U.

Note that the arguments described as "Not for use" are included for the sparcl package to function with GenePattern but should be ignored by the R user.

Value

`hc`	The output of a call to "hclust", giving the results of hierarchical sparse clustering.
`ws`	The p-vector of feature weights.
`u`	The nxn dissimilarity matrix passed into hclust, of the form $(sum_j w_j d_ii'j)_ii'$.
`dists`	The (n*n)xp dissimilarity matrix for the data matrix x. This is useful if additional calls to HierarchicalSparseCluster will be made.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten and Tibshirani (2009) A framework for feature selection in clustering.

See Also

HierarchicalSparseCluster.permute,KMeansSparseCluster,KMeansSparseCluster.permute

Examples

```
  # Generate 2-class data
  set.seed(1)
  x <- matrix(rnorm(100*50),ncol=50)
  y <- c(rep(1,50),rep(2,50))
  x[y==1,1:25] <- x[y==1,1:25]+2
  # Do tuning parameter selection for sparse hierarchical clustering
  perm.out <- HierarchicalSparseCluster.permute(x, wbounds=c(1.5,2:6),
nperms=5)
  print(perm.out)
  plot(perm.out)
  # Perform sparse hierarchical clustering
  sparsehc <- HierarchicalSparseCluster(dists=perm.out$dists,
wbound=perm.out$bestw, method="complete")
  # faster than   sparsehc <- HierarchicalSparseCluster(x=x,wbound=perm.out$bestw,
#  method="complete")
  par(mfrow=c(1,2))
  plot(sparsehc)
  plot(sparsehc$hc, labels=rep("", length(y)))
  print(sparsehc)
  # Plot using knowledge of class labels in order to compare true class
  #   labels to clustering obtained
  par(mfrow=c(1,1))
  ColorDendrogram(sparsehc$hc,y=y,main="My Simulated Data",branchlength=.007)
  # Now, what if we want to see if out data contains a *secondary*
  #   clustering after accounting for the first one obtained. We
  #   look for a complementary sparse clustering:
  sparsehc.comp <- HierarchicalSparseCluster(x,wbound=perm.out$bestw,
     method="complete",uorth=sparsehc$u)
  # Redo the analysis, but this time use "absolute value" dissimilarity:
  perm.out <- HierarchicalSparseCluster.permute(x, wbounds=c(1.5,2:6),
    nperms=5, dissimilarity="absolute.value")
  print(perm.out)
  plot(perm.out)
  # Perform sparse hierarchical clustering
  sparsehc <- HierarchicalSparseCluster(dists=perm.out$dists, wbound=perm.out$bestw,
method="complete",
 dissimilarity="absolute.value")
  par(mfrow=c(1,2))
  plot(sparsehc)
```

`HierarchicalSparseCluster.permute`
Choose tuning parameter for sparse hierarchical clustering

Description

The tuning parameter controls the L1 bound on w, the feature weights. A permutation approach is used to select the tuning parameter.

Usage

```
HierarchicalSparseCluster.permute(x, nperms = 10, wbounds = NULL,
dissimilarity=c("squared.distance",
"absolute.value"),standardize.arrays=FALSE)
## S3 method for class 'HierarchicalSparseCluster.permute'
plot(x,...)
## S3 method for class 'HierarchicalSparseCluster.permute'
print(x,...)
```

Arguments

`x`	A nxp data matrix, with n observations and p feaures.
`nperms`	The number of permutations to perform.
`wbounds`	The sequence of tuning parameters to consider. The tuning parameters are the L1 bound on w, the feature weights. If NULL, then a default sequence will be used. If non-null, should be greater than 1.
`dissimilarity`	How should dissimilarity be computed? Default is squared.distance.
`standardize.arrays`	Should the arrays first be standardized? Default is FALSE.
`...`	not used.

Details

Let $d_ii'j$ denote the dissimilarity between observations i and i' along feature j.

Sparse hierarchical clustering seeks a p-vector of weights w (one per feature) and a nxn matrix U that optimize $maximize_U,w sum_j w_j sum_ii' d_ii'j U_ii'$ subject to $||w||_2 <= 1, ||w||_1 <= s, w_j >= 0, sum_ii' U_ii'^2 <= 1$, where s is a value for the L1 bound on w. Let O(s) denote the objective function with tuning parameter s: i.e. $O(s)=sum_j w_j sum_ii' d_ii'j U_ii'$.

We permute the data as follows: within each feature, we permute the observations. Using the permuted data, we can run sparse hierarchical clustering with tuning parameter s, yielding the objective function O*(s). If we do this repeatedly we can get a number of O*(s) values.

Then, the Gap statistic is given by $Gap(s)=log(O(s))-mean(log(O*(s)))$. The optimal s is that which results in the highest Gap statistic. Or, we can choose the smallest s such that its Gap statistic is within $sd(log(O*(s)))$ of the largest Gap statistic.

Value

`gaps`	The gap statistics obtained (one for each of the tuning parameters tried). If O(s) is the objective function evaluated at the tuning parameter s, and O*(s) is the same quantity but for the permuted data, then Gap(s)=log(O(s))-mean(log(O*(s))).
`sdgaps`	The standard deviation of log(O*(s)), for each value of the tuning parameter s.
`nnonzerows`	The number of features with non-zero weights, for each value of the tuning parameter.
`wbounds`	The tuning parameters considered.
`bestw`	The value of the tuning parameter corresponding to the highest gap statistic.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten and Tibshirani (2009) A framework for feature selection in clustering.

See Also

HierarchicalSparseCluster, KMeansSparseCluster, KMeansSparseCluster.permute

Examples

```
  # Generate 2-class data
  set.seed(1)
  x <- matrix(rnorm(100*50),ncol=50)
  y <- c(rep(1,50),rep(2,50))
  x[y==1,1:25] <- x[y==1,1:25]+2
  # Do tuning parameter selection for sparse hierarchical clustering
  perm.out <- HierarchicalSparseCluster.permute(x, wbounds=c(1.5,2:6),
nperms=5)
  print(perm.out)
  plot(perm.out)
  # Perform sparse hierarchical clustering
  sparsehc <- HierarchicalSparseCluster(dists=perm.out$dists, wbound=perm.out$bestw,
method="complete")
  par(mfrow=c(1,2))
  plot(sparsehc)
  plot(sparsehc$hc, labels=rep("", length(y)))
  print(sparsehc)
  # Plot using knowledge of class labels in order to compare true class
  #   labels to clustering obtained
  par(mfrow=c(1,1))
  ColorDendrogram(sparsehc$hc,y=y,main="My Simulated
Data",branchlength=.007)
```

HierarchicalSparseCluster.wrapper

A wrapper for the hierarchical sparse clustering algorithm

Description

A wrapper for HierarchicalSparseCluster which reads in the data in GCT file format, and then automatically chooses the optimal tuning parameter value using HierarchicalSparseCluster.permute if not specified.

Usage

```
HierarchicalSparseCluster.wrapper(file, method=c("average", "complete", "single",
 "centroid"),
wbound=NULL, silent=FALSE, cluster.features=FALSE,
method.features=c("average", "complete",
"single","centroid"),output.cluster.files=TRUE,outputfile.prefix=NULL,maxnumgenes=5000,
standardize.arrays=TRUE)
```

Arguments

`file`	A GCT filename in the working directory containing the data to be clustered.
`method`	The type of linkage to use in the hierarchical clustering - "single", "complete", "average", or "centroid".
`wbound`	The L1 bound on w to use; this is the tuning parameter for sparse hierarchical clustering. If NULL, then it will be chosen via HierarchicalSparseCluster.permute.
`silent`	Print out progress?
`cluster.features`	Is a clustering for the features with non-zero weights also desired? Default is FALSE.
`method.features`	If cluster.features is TRUE, then the type of linkage used to cluster the features with non-zero weights: one of "single", "complete", "average", or "centroid".
`output.cluster.files`	Should files containing the clustering be output? Default is TRUE.
`outputfile.prefix`	The prefix for the output files. If NULL, then the prefix of the input file is used.
`maxnumgenes`	Limit the analysis to some number of genes with highest marginal variance, for computational reasons. This is recommended when the number of genes is very large. If NULL, then all genes are used.
`standardize.arrays`	Should the arrays first be standardized? Default is TRUE.

Value

`hc`	The output of a call to "hclust", giving the results of hierarchical sparse clustering.
`ws`	The p-vector of feature weights.
`u`	The nxn dissimilarity matrix passed into hclust, of the form $(sum_j w_j d_ii'j)_ii'$.
`dists`	The (n*n)xp dissimilarity matrix for the data matrix x. This is useful if additional calls to HierarchicalSparseCluster will be made.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten and Tibshirani (2009) A framework for feature selection in clustering.

See Also

HierarchicalSparseCluster.permute,KMeansSparseCluster,KMeansSparseCluster.permute

---

`KMeansSparseCluster` *Performs sparse k-means clustering*

Description

This function performs sparse k-means clustering. You must specify a number of clusters K and an L1 bound on w, the feature weights.

Usage

```
KMeansSparseCluster(x, K=NULL, wbounds = NULL, nstart = 20, silent =
FALSE, maxiter=6, centers=NULL)
## S3 method for class 'KMeansSparseCluster'
plot(x,...)
## S3 method for class 'KMeansSparseCluster'
print(x,...)
```

Arguments

`x`	An nxp data matrix. There are n observations and p features.
`K`	The number of clusters desired ("K" in K-means clustering). Must provide either K or centers.
`wbounds`	A single L1 bound on w (the feature weights), or a vector of L1 bounds on w. If wbound is small, then few features will have non-zero weights. If wbound is large then all features will have non-zero weights. Should be greater than 1.
`nstart`	The number of random starts for the k-means algorithm.
`silent`	Print out progress?
`maxiter`	The maximum number of iterations.
`centers`	Optional argument. If you want to run the k-means algorithm starting from a particular set of clusters, then you can enter the Kxp matrix of cluster centers here. Default use case involves taking centers=NULL and instead specifying K.
`...`	not used.

Details

We seek a p-vector of weights w (one per feature) and a set of clusters C1,...,CK that optimize

$maximize_C1,...,CK,w sum_j w_j BCSS_j$ subject to $||w||_2 <= 1, ||w||_1 <= wbound, w_j >= 0$

where $BCSS_j$ is the between cluster sum of squares for feature j. An iterative approach is taken: with w fixed, optimize with respect to C1,...,CK, and with C1,...,CK fixed, optimize with respect to w. Here, wbound is a tuning parameter which determines the L1 bound on w.

The non-zero elements of w indicate features that are used in the sparse clustering.

Value

If wbounds is a vector, then a list with elements as follows (one per element of wbounds). If wbounds is just a single value, then elements as follows:

`ws`	The p-vector of feature weights.
`Cs`	The clustering obtained.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten and Tibshirani (2009) A framework for feature selection in clustering.

See Also

KMeansSparseCluster.permute,HierarchicalSparseCluster

Examples

```
# generate data
set.seed(11)
x <- matrix(rnorm(50*70),ncol=70)
x[1:25,1:20] <- x[1:25,1:20]+1
x <- scale(x, TRUE, TRUE)
# choose tuning parameter
km.perm <- KMeansSparseCluster.permute(x,K=2,wbounds=seq(3,7,len=15),nperms=5)
print(km.perm)
plot(km.perm)
# run sparse k-means
km.out <- KMeansSparseCluster(x,K=2,wbounds=km.perm$bestw)
print(km.out)
plot(km.out)
# run sparse k-means for a range of tuning parameter values
km.out <- KMeansSparseCluster(x,K=2,wbounds=seq(1.3,4,len=8))
print(km.out)
plot(km.out)
# Run sparse k-means starting from a particular set of cluster centers
#in the k-means algorithm.
```

```
km.out <- KMeansSparseCluster(x,wbounds=2:7,centers=x[c(1,3,5),])
```

KMeansSparseCluster.permute
Choose tuning parameter for sparse k-means clustering

Description

The tuning parameter controls the L1 bound on w, the feature weights. A permutation approach is used to select the tuning parameter.

Usage

```
KMeansSparseCluster.permute(x, K=NULL, nperms = 25, wbounds = NULL,
silent = FALSE, nvals = 10, centers=NULL)
## S3 method for class 'KMeansSparseCluster.permute'
print(x,...)
## S3 method for class 'KMeansSparseCluster.permute'
plot(x,...)
```

Arguments

`x`	The nxp data matrix, n is the number of observations and p the number of features.
`K`	The number of clusters desired - that is, the "K" in K-means clustering. Must specify K or centers.
`nperms`	Number of permutations.
`wbounds`	The range of tuning parameters to consider. This is the L1 bound on w, the feature weights. If NULL, then a range of values will be chosen automatically. Should be greater than 1 if non-null.
`silent`	Print out progress?
`nvals`	If wbounds is NULL, then the number of candidate tuning parameter values to consider.
`centers`	Optional argument. If you want to run the k-means algorithm starting from a particular set of clusters, then you can enter the Kxp matrix of cluster centers here. Default use case involves taking centers=NULL and instead specifying K.
`...`	not used.

Details

Sparse k-means clustering seeks a p-vector of weights w (one per feature) and a set of clusters C1,...,CK that optimize $maximize_C1,...,CK,w sum_j w_j BCSS_j$ subject to $||w||_2 <= 1, ||w||_1 <= s, w_j >= 0$, where $BCSS_j$ is the between cluster sum of squares for feature j, and s is a value for the L1 bound on w. Let O(s) denote the objective function with tuning parameter s: i.e. $O(s)=sum_j w_j BCSS_j$.

We permute the data as follows: within each feature, we permute the observations. Using the permuted data, we can run sparse K-means with tuning parameter s, yielding the objective function O*(s). If we do this repeatedly we can get a number of O*(s) values.

Then, the Gap statistic is given by $Gap(s)=log(O(s))-mean(log(O*(s)))$. The optimal s is that which results in the highest Gap statistic. Or, we can choose the smallest s such that its Gap statistic is within $sd(log(O*(s)))$ of the largest Gap statistic.

Value

`gaps`	The gap statistics obtained (one for each of the tuning parameters tried). If O(s) is the objective function evaluated at the tuning parameter s, and O*(s) is the same quantity but for the permuted data, then Gap(s)=log(O(s))-mean(log(O*(s))).
`sdgaps`	The standard deviation of log(O*(s)), for each value of the tuning parameter s.
`nnonzerows`	The number of features with non-zero weights, for each value of the tuning parameter.
`wbounds`	The tuning parameters considered.
`bestw`	The value of the tuning parameter corresponding to the highest gap statistic.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten and Tibshirani (2009) A framework for feature selection in clustering.

See Also

KMeansSparseCluster, HierarchicalSparseCluster, HierarchicalSparseCluster.permute

Examples

```
# generate data
set.seed(11)
x <- matrix(rnorm(50*70),ncol=70)
x[1:25,1:10] <- x[1:25,1:10]+1.5
x <- scale(x, TRUE, TRUE)
# choose tuning parameter
km.perm <-
KMeansSparseCluster.permute(x,K=2,wbounds=seq(2,5,len=8),nperms=3)
print(km.perm)
plot(km.perm)
# run sparse k-means
km.out <- KMeansSparseCluster(x,K=2,wbounds=km.perm$bestw)
print(km.out)
plot(km.out)
# run sparse k-means for a range of tuning parameter values
km.out <- KMeansSparseCluster(x,K=2,wbounds=2:7)
print(km.out)
plot(km.out)
```

```
# Repeat, but this time start with a particular choice of cluster
# centers.
# This will do 4-means clustering starting with this particular choice
# of cluster centers.
km.perm.out <- KMeansSparseCluster.permute(x,wbounds=2:6, centers=x[1:4,],nperms=3)
print(km.out)
plot(km.out)
```

Index