Package: MapperAlgo (via r-universe)

October 21, 2024

Title Topological Data Analysis: Mapper Algorithm

Version 1.0.1

Date 2024-09-17

Maintainer ChiChien Wang <kennywang2003@gmail.com>

Description The Mapper algorithm from Topological Data Analysis, the steps are as follows 1. Define a filter (lens) function on the data. 2. Perform clustering within each level set. 3. Generate a complex from the clustering results.

Depends R (>= 3.1.2)

Suggests fastcluster, networkD3, igraph, testthat (>= 3.0.0)

License MIT + file LICENSE

URL https://github.com/kennywang112/MapperAlgo/

BugReports https://github.com/kennywang112/MapperAlgo/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

RoxygenNote 7.3.2

Repository https://kennywang112.r-universe.dev

RemoteUrl https://github.com/kennywang112/mapperalgo

RemoteRef HEAD

RemoteSha ca640976d52f42bd74abf1ae960e0dfcd7529736

Contents

`cluster_cutoff_at_first_empty_bin` *Cut the hierarchical clustering tree to define clusters*

Description

Cut the hierarchical clustering tree to define clusters

Usage

```
cluster_cutoff_at_first_empty_bin(heights, diam, num_bins_when_clustering)
```

Arguments

`heights`	Heights of the clusters.
`diam`	Diameter of the clusters.
`num_bins_when_clustering`	Number of bins when clustering.

Value

The cutoff height for the clusters.

`cover_points` *Cover points based on intervals and overlap*

Description

Cover points based on intervals and overlap

Usage

```
cover_points(
  lsfi,
  filter_min,
  interval_width,
  percent_overlap,
  filter_values,
  num_intervals
)
```

Arguments

`lsfi`	Level set flat index.
`filter_min`	Minimum filter value.
`interval_width`	Width of the interval.
`percent_overlap`	Percentage overlap between intervals.
`filter_values`	The filter values to be analyzed.
`num_intervals`	Number of intervals.

Value

Indices of points in the range.

`find_best_k_for_kmeans` *Find the optimal number of clusters for k-means*

Description

This function calculates the total within-cluster sum of squares (WSS) for a range of cluster numbers and identifies the best number of clusters (k) based on the elbow method.

Usage

```
find_best_k_for_kmeans(dist_object, max_clusters = 10)
```

Arguments

`dist_object`	A distance matrix or data frame containing the data to be clustered.
`max_clusters`	The maximum number of clusters to test for k-means. Default is 10.

Value

The optimal number of clusters (k) based on the elbow method.

MapperAlgo *Topological data analysis: Mapper algorithm*

Description

The Mapper algorithm is a method for topological data analysis that provides a way to visualize the structure of high-dimensional data. The Mapper algorithm is a generalization of the Reeb graph construction, which is a method for visualizing the topology of scalar fields.

Usage

```
MapperAlgo(filter_values, intervals, percent_overlap, num_bins_when_clustering, methods)
```

Arguments

`filter_values`	A data frame or matrix of the data to be analyzed.
`intervals`	An integer specifying the number of intervals to divide the filter values into.
`percent_overlap`	An integer specifying the percentage of overlap between consecutive intervals.
`num_bins_when_clustering`	An integer specifying the number of bins to use when clustering the data.
`methods`	A character string specifying the clustering method to use. The default is "hierarchical".

Value

An adjacency matrix and other components of the Mapper graph, including:

`adjacency`	An adjacency matrix of the Mapper graph.
`num_vertices`	The number of vertices in the Mapper graph.
`level_of_vertex`	A vector specifying the level of each vertex.
`points_in_vertex`	A list of the indices of the points in each vertex.
`points_in_level_set`	A list of the indices of the points in each level set.
`vertices_in_level_set`	A list of the indices of the vertices in each level set.

Author(s)

ChiChien Wang

References

The original paper on the Mapper algorithm is: G. Singh, F. Memoli, G. Carlsson (2007). Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition, Point Based Graphics 2007, Prague, September 2007. This code is based on Paul Pearson's implementation of the Mapper algorithm in R, optimized for speed and memory usage. You can install using the following command: devtools::install_github("paultpearson/TDAmapper")

Examples

```
library(igraph)
library(networkD3)

data("iris")

mapper <- MapperAlgo(
  filter_values = iris[,1:4],
  intervals = 4,
  percent_overlap = 50,
  num_bins_when_clustering = 30,
  methods = "hierarchical")

graph <- graph.adjacency(mapper$adjacency, mode="undirected")
l = length(V(graph))
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
# Distribution of specific variable in each vertex - Majority vote
var.maj.vertex <- c()
filter.vertex <- c()

for (i in 1:l){
  points.in.vertex <- mapper$points_in_vertex[[i]]
  Mode.in.vertex <- Mode(iris$Species[points.in.vertex])
  var.maj.vertex <- c(var.maj.vertex, as.character(Mode.in.vertex))
}

# Size
vertex.size <- rep(0, l)
for (i in 1:l){
  points.in.vertex <- mapper$points_in_vertex[[i]]
  vertex.size[i] <- length(mapper$points_in_vertex[[i]])
}

MapperNodes <- mapperVertices(mapper, 1:nrow(iris))
MapperNodes$var.maj.vertex <- as.factor(var.maj.vertex)
MapperNodes$Nodesize <- vertex.size
MapperLinks <- mapperEdges(mapper)
forceNetwork(Nodes = MapperNodes, Links = MapperLinks, Target = "Linktarget",
             Value = "Linkvalue", NodeID = "Nodename", Nodesize = "Nodesize",
             Group = "var.maj.vertex", opacity = 1, zoom = TRUE,
             linkDistance = 10, charge = -10, legend = TRUE)
```

mapperEdges *Create Mapper Edges*

Description

This function generates the edges of the Mapper graph by analyzing the adjacency matrix. It returns a data frame with source and target vertices that are connected by edges.

Usage

```
mapperEdges(m)
```

Arguments

m	The Mapper output object that contains the adjacency matrix and other graph components.

Value

A data frame containing the source (`Linksource`), target (`Linktarget`), and edge values (`Linkvalue`) for the graph's edges.

mapperVertices *Create Mapper Vertices*

Description

This function generates the vertices of the Mapper graph, including their labels and groupings. It returns a data frame with the vertex names, the group each vertex belongs to, and the size of each vertex.

Usage

```
mapperVertices(m, pt_labels)
```

Arguments

m	The Mapper output object that contains information about the vertices and level sets.
pt_labels	A vector of point labels to be assigned to the points in each vertex.

Value

A data frame containing the vertex names (`Nodename`), group information (`Nodegroup`), and vertex sizes (`Nodesize`).

`perform_clustering` *Perform clustering within a level set*

Description

Perform clustering within a level set

Usage

```
perform_clustering(
  points_in_this_level,
  filter_values,
  num_bins_when_clustering,
  methods,
  max_kmeans_clusters = 10,
  eps = 0.5,
  minPts = 5,
  num_clusters = 5
)
```

Arguments

`points_in_this_level`	Points in the current level set.
`filter_values`	The filter values.
`num_bins_when_clustering`	Number of bins when clustering.
`methods`	Specify the clustering method to be used, e.g., "hclust" or "kmeans".
`max_kmeans_clusters`	Maximum number of clusters when using k-means clustering.
`eps`	The maximum distance between two samples for one to be considered as in the neighborhood of the other.
`minPts`	The number of samples in a neighborhood for a point to be considered as a core point.
`num_clusters`	Number of clusters when using PAM clustering.

Value

A list containing the number of vertices, external indices, and internal indices.

simplcial_complex *Construct adjacency matrix of the simplicial complex*

Description

Construct adjacency matrix of the simplicial complex

Usage

```
simplcial_complex(
  filter_values,
  vertex_index,
  num_levelsets,
  num_intervals,
  vertices_in_level_set,
  points_in_vertex
)
```

Arguments

`filter_values`	A matrix of filter values.
`vertex_index`	The number of vertices.
`num_levelsets`	The total number of level sets.
`num_intervals`	A vector representing the number of intervals for each filter.
`vertices_in_level_set`	A list where each element contains the vertices corresponding to each level set.
`points_in_vertex`	A list where each element contains the points corresponding to each vertex.

Value

An adjacency matrix representing the simplicial complex.

to_lsfi *Convert level set multi-index (lsmi) to flat index (lsfi)*

Description

Convert level set multi-index (lsmi) to flat index (lsfi)

Usage

```
to_lsfi(lsmi, num_intervals)
```

Arguments

`lsmi`	Level set multi-index.
`num_intervals`	Number of intervals.

Value

A flat index corresponding to the multi-index.

`to_lsmi` *Convert level set flat index (lsfi) to multi-index (lsmi)*

Description

Convert level set flat index (lsfi) to multi-index (lsmi)

Usage

```
to_lsmi(lsfi, num_intervals)
```

Arguments

`lsfi`	Level set flat index.
`num_intervals`	Number of intervals.

Value

A multi-index corresponding to the flat index.

Index