Package ‘MapperAlgo’

September 18, 2024

Title Topological Data Analysis: Mapper Algorithm

Version 1.0

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, 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

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Author ChiChien Wang [aut, cre, trl],
Paul Pearson [ctb],
Daniel Muellner [ctb],
Gurjeet Singh [ctb]

Repository CRAN

Date/Publication 2024-09-18 14:30:07 UTC

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.

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

Description

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

Usage

```
lsfi_from_lsmi(lsmi, num_intervals)
```

Arguments

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

Value

A flat index corresponding to the multi-index.

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

Description

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

Usage

```
lsmi_from_lsfi(lsfi, num_intervals)
```

Arguments

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

Value

A multi-index corresponding to the flat index.

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)
```

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.

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)

data("iris")

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

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)
```

`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
)
```

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.

Value

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

Index