Package 'FuzzyDBScan'

January 20, 2025

Title Run and Predict a Fuzzy DBScan

Version 0.0.3

Description An interface for training Fuzzy DBScan with both Fuzzy Core and Fuzzy Border. Therefore, the package provides a method to initialize and run the algorithm and a function to predict new data w.t.h. of 'R6'.
The package is build upon the paper ``Fuzzy Extensions of the DBScan algorithm" from Ienco and Bordogna (2018) <doi:10.1007/s00500-016-2435-0>.
A predict function assigns new data according to the same criteria as the algorithm itself. However, the prediction function freezes the algorithm to preserve the trained cluster structure and treats each new prediction object individually.

License LGPL-3

Depends R (>= 4.0.0)

Imports ggplot2, R6, data.table, dbscan, checkmate

Suggests testthat (>= 3.0.0), rmarkdown, factoextra, spelling

Encoding UTF-8

RoxygenNote 7.2.2

Language en-US

NeedsCompilation no

Author Henri Funk [aut, cre]

Maintainer Henri Funk <Henri.Funk@stat.uni-muenchen.de>

Repository CRAN

Date/Publication 2023-01-10 13:53:18 UTC

Contents

Fuzzy_DBScan 2

Index **5**

Fuzzy_DBScan *Fuzzy DBScan*

Description

This object implements fuzzy DBScan with both, fuzzy cores and fuzzy borders. Additionally, it provides a predict function.

Details

A method to initialize and run the algorithm and a function to predict new data. The package is build upon the paper "Fuzzy Extensions of the DBScan algorithm" from Ienco and Bordogna. The predict function assigns new data based on the same criteria as the algorithm itself. However, the prediction function freezes the algorithm to preserve the trained cluster structure and treats each new prediction object individually. Note, that border points are included to the cluster.

Public fields

`dta` data.frame | matrix
: The data to be clustered by the algorithm. Allowed are only numeric columns.

`eps` numeric
: The size (radius) of the epsilon neighborhood. If the radius contains 2 numbers, the fuzzy cores are calculated between the minimum and the maximum radius. If epsilon is a single number, the algorithm looses the fuzzy core property. If the length of `pts` is also 1L, the algorithm equals to non-fuzzy DBScan.

`pts` numeric
: number of maximum and minimum points required in the `eps` neighborhood for core points (excluding the point itself). If the length of the argument is 1, the algorithm looses its fuzzy border property. If the length of `eps` is also 1L, the algorithm equals to non-fuzzy DBScan.

`clusters` factor
: Contains the assigned clusters per observation in the same order as in `dta`.

`dense` numeric
: Contains the assigned density estimates per observation in the same order as in `dta`.

`point_def` character
: Contains the assigned definition estimates per observation in the same order as in `dta`. Possible are "Core Point", "Border Point" and "Noise".

`results` data.table
: A table where each column indicates for the probability of the new data to belong to a respective cluster.

Methods

Public methods:

- `FuzzyDBScan$new()`
- `FuzzyDBScan$predict()`

- `FuzzyDBScan$plot()`
- `FuzzyDBScan$clone()`

Method `new()`**:** Create a FuzzyDBScan object. Apply the fuzzy DBScan algorithm given the data `dta`, the range of the radius `eps` and the range of the Points `pts`.

Usage:

```
FuzzyDBScan$new(dta, eps, pts)
```

Arguments:

`dta` data.frame | matrix
: The data to be clustered by the algorithm. Allowed are only numeric columns.

`eps` numeric
: The size (radius) of the epsilon neighborhood. If the radius contains 2 numbers, the fuzzy cores are calculated between the minimum and the maximum radius. If epsilon is a single number, the algorithm looses the fuzzy core property. If the length of `pts` is also 1L, the algorithm equals to non-fuzzy DBScan.

`pts` numeric
: number of maximum and minimum points required in the `eps` neighborhood for core points (excluding the point itself). If the length of the argument is 1, the algorithm looses its fuzzy border property. If the length of `eps` is also 1L, the algorithm equals to non-fuzzy DBScan.

Method `predict()`**:** Predict new data with the initialized algorithm.

Usage:

```
FuzzyDBScan$predict(new_data, cmatrix = TRUE)
```

Arguments:

`new_data` data.frame | matrix
: The data to be predicted by the algorithm. Allowed are only numeric columns which should match to `self$dta`.

`cmatrix` logical
: Indicating whether the assigned cluster should be returned in form of a matrix where each column indicates for the probability of the new data to belong to a respective cluster. The object will have the same shape as the `results` field. If set to `FALSE` the shape of the returned assigned clusters is a two-column data.table with one column indicating the assigned cluster and the second column indicating the respective probability of the new data.

Method `plot()`**:** Plot clusters and soft labels on two features.

Usage:

```
FuzzyDBScan$plot(x, y)
```

Arguments:

`x` character
: Feature to plot on the x-axis.

`y` character
: Feature to plot on the y-axis.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FuzzyDBScan$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Ienco, Dino, and Gloria Bordogna. Fuzzy extensions of the DBScan clustering algorithm. Soft Computing 22.5 (2018): 1719-1730.

Examples

```
# load factoextra for data and ggplot for plotting
library(factoextra)
dta = multishapes[, 1:2]
eps = c(0, 0.2)
pts = c(3, 15)
# train DBScan based on data, ep and pts
cl = FuzzyDBScan$new(dta, eps, pts)
# Plot DBScan for x and y
library(ggplot2)
cl$plot("x", "y")
# produce test data
x <- seq(min(dta$x), max(dta$x), length.out = 50)
y <- seq(min(dta$y), max(dta$y), length.out = 50)
p_dta = expand.grid(x = x, y = y)
# predict on test data and plot results
p = cl$predict(p_dta, FALSE)
ggplot(p, aes(x = p_dta[, 1], y = p_dta[, 2], colour = as.factor(cluster))) +
  geom_point(alpha = p$dense)
```

Index

character, *2*, *3*

data.frame, *2*, *3*
data.table, *2*, *3*

factor, *2*
Fuzzy_DBScan, 2
FuzzyDBScan *(*Fuzzy_DBScan*)*, 2

logical, *3*

matrix, *2*, *3*

numeric, *2*, *3*