Package ‘subspaceMOA’

April 25, 2017

Type Package

Title Interface to 'subspaceMOA'

Version 0.6.0

Date 2017-04-25

Author c(person("Marwan", "Hassani", role=c("aut", "cre"), email="rsubspace@cs.rwth-aachen.de"),
person("Matthias", "Hansen", role="aut"),
person("Yunsu", "Kim", role="ctb"),
person("Thomas", "Seidl", role="ctb"),
person("University of", "Waikato", role=c("ctb", "cph")))

Maintainer Marwan Hassani <rsubspace@cs.rwth-aachen.de>

Description An interface to 'subspaceMOA', a Framework for the Evaluation of subspace stream clustering algorithms. (see <http://dme.rwth-aachen.de/de/subspacemoa> for more information.)

License GPL-2

Imports rJava(>= 0.9), ggplot2 (>= 1.0.1), gridExtra (>= 2.0.0), grid
(>= 3.2.2), fields (>= 8.3), magrittr (>= 1.5), shiny (>= 1.0.2), methods, stats (>= 3.2.2)

Depends stream(>= 1.2), streamMOA(>= 1.1)

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-04-25 21:30:49 UTC

R topics documented:

animate_stream_interactive .................................................. 2
DSC_clique ................................................................. 3
DSC_HDDStream ............................................................. 3
DSC_p3c ........................................................................ 4
DSC_PreDeConStream ....................................................... 5
DSC_proclus ................................................................. 6
DSC_subclu ................................................................. 6
DSC_subspaceCluStream .................................................... 7
animate_stream_interactive

Animate Stream Clustering.

Description

A function to plot data streams and clusterings. The visualisation is based on shiny and ggplot. Data is plotted as a scatterplot matrix and individual scatterplots can be selected for a more detailed view that includes tooltips. Please note that this function was developed for the Streaming algorithms in the subspaceMOA package and may or may not work for streams and clustering algorithms.

Usage

animate_stream_interactive(dsc, dsd, step = 1500, delay = 10000,
launch.browser = getOption("shiny.launch.browser", interactive()))

Arguments

dsc  a DSC object representing the clustering of a data stream.
dsd  a DSD object representing a data stream.
step the step size used in animate_stream_interactive. This regulates how many points will be taken out of the stream, clustered and the plotted along with their clusters every time a step is performed.
delay time between two clustering steps
launch.browser will be passed on to runApp, so that the visualisation can be shown in e.g. RStudio’s Viewer pane, if this is desired.

Examples

clusterer <- DSC_ThreeStage(DSC_p3c(), DSC_subspaceCluStream())
stream <- DSD_RandomRBFSubspaceGeneratorEvents()

## Not run:
animate_stream_interactive(clusterer, stream)

## End(Not run)
**DSC_clique**

*CLIQUETM algorithm for use with DSC_ThreeStage*

**Description**

An implementation of the CLIQUE algorithm that can be used with DSC_ThreeStage. For more details on this algorithm, consult CLIQUE.

**Usage**

```r
dsc_clique(xi = 10, tau = 0.2)
```

**Arguments**

- **xi**
  - the grid size used. E.g. a value of 10 means that the dataspace is divided into 10 regions along each dimension.
- **tau**
  - the density threshold used to determine whether a hypercube is dense.

**Examples**

```r
dsc <- DSC_ThreeStage(macro=dsc_clique(), micro=DSC_subspaceCluStream())
dsd <- DSD_RandomRBFSubspaceGeneratorEvents()
update(dsc, dsd, 1000)
```

---

**DSC_HDDStream**

*Density-based Projected Clustering over High-Dimensional Data*

**Description**

This function creates a DSC object that represents an instance of the HDDStream algorithm and can be used for stream clustering.

**Usage**

```r
DSC_HDDStream(epsilonN = 0.1, beta = 0.5, mu = 10, lambda = 0.5,
               initPoints = 2000, pi = 30, kappa = 10, delta = 0.001, offline = 2,
               speed = 100)
```

**Arguments**

- **epsilonN**
  - radius of each neighborhood
- **beta**
  - control the effect of mu
- **mu**
  - minimum number of points desired to be in a microcluster
- **lambda**
  - decaying parameter
initPoints  number of points to use for initialization
pi          number of maximal subspace dimensionality
kappa       parameter to define preference weighted vector
delta       defines the threshold for the variance
offline     offline multiplier for epsilon
speed       number of incoming points per time unit

Details

HDDStream is an algorithm for the density-based projected clustering of high-dimensional data streams.

The algorithm is initialized by buffering the first initPoints points that arrive and then applying the PreDeCon algorithm over these points.

Then, Microclusters are maintained online by adding each new point to its closest core Microcluster iff doing so does not increase the projected radius of this microcluster beyond epsilonN. If a point can not be added to a core microcluster, an attempt will be made to add it to an outlier microcluster, with the same criterion as for core microclusters. If these attempts both fail, the point will start its own microcluster. Microclusters are aged according to the decaying parameter lambda.

Macroclustering is performed on-demand, using the PreDeCon algorithm.

Examples

dsc <- DSC_HDDStream()
dsd <- DSD_RandomRBFSubspaceGeneratorEvents()
update(dsc, dsd, 1000)

Description

An implementation of the P3C algorithm that can be used with DSC_ThreeStage. For more details on this algorithm, consult P3C

Usage

DSC_p3c(poissonThreshold = 10, chiSquareAlpha = 0.001)

Arguments

poissonThreshold
threshold value to determine whether two bins will be merged. Note that the value provided will be used as a negative power of 10. E.g. if a value of 20 is provided here, then the algorithm will use a threshold of 1.0\times10^{-20}. dsc <- DSC_ThreeStage(macro=DSC_p3c(), micro=DSC_subspaceCluStream()) dsd <- DSD_RandomRBFSubspaceGeneratorEvents() update(dsc, dsd, 1000)
DSC_PreDeConStream

chiSquareAlpha  threshold value for the chi-square distribution that is used to determine whether an area is dense.

DSC_PreDeConStream  Density-Based Projected Clustering of Data Streams

Description

This function creates a DSC object that represents an instance of the PreDeConStream algorithm and can be used for stream clustering.

Usage

DSC_PreDeConStream(epsilonN = 0.7, beta = 0.3, muN = 10, muF = 3, lambda = 0.1, initPoints = 1000, tau = 2, kappa = 10, delta = 0.01, offline = 2, speed = 100)

Arguments

epsilonN  radius of each neighborhood
beta       control the effect of mu
muN        minimum number of points in microclusters
muF        minimum number of points in macroclusters
lambda     decaying parameter
initPoints  number of points to use for initialization
tau        number of maximal subspace dimensionality
kappa      parameter to define preference weighted vector
delta      defines the threshold for the variance
offline     offline multiplier for epsilon
speed       processing number of incoming points per time unit

Details

The PreDeConStream algorithm is a Density-Based algorithm for the projected clustering of data streams. To initially obtain a set of microclusters initPoints points are buffered and clustered using the PreDeCon algorithm. Then, microclusters are maintained by checking for each new point whether it falls within the radius of an existing microcluster, similar to DSC_DenStream. Microclusters are aged according to a decay parameter lambda. Macroclusters are also maintained throughout the run of the algorithm by updating the affected macroclusters, whenever a change in the microcluster structure has occurred, using a component of the PreDeCon algorithm to do so.

Examples

dsc <- DSC_PreDeConStream()
 dsd <- DSD_RandomRBFSubspaceGeneratorEvents()
 update(dsc, dsd, 1000)
DSC_proclus  
*ProClus algorithm for use with DSC_ThreeStage*

**Description**

An implementation of the ProClus algorithm that can be used with DSC_ThreeStage. For more details on this algorithm, consult ProClus.

**Usage**

DSC_proclus(numOfClusters = 5, avgDimensions = 3)

**Arguments**

- **numOfClusters**: Number of Clusters to be found.
- **avgDimensions**: Average number of dimensions in which each cluster resides.

```
dsc <- DSC_ThreeStage(macro=DSC_proclus(), micro=DSC_subspaceCluStream())
dsd <- DSD_RandomRBFSubspaceGeneratorEvents() update(dsc, dsd, 1000)
```

DSC_subclu  
*SubClu algorithm for use with DSC_ThreeStage*

**Description**

An implementation of the SubClu algorithm that can be used with DSC_ThreeStage. For more details on this algorithm, consult SubClu.

**Usage**

DSC_subclu(epsilon = 0.05, minSupport = 50, minOutputDim = 3)

**Arguments**

- **epsilon**: this parameter determines the size of the epsilon environment for the DBSCAN that is run as a part of this algorithm.
- **minSupport**: minimum number of points in the epsilon environment.
- **minOutputDim**: minimum dimensionality that a cluster must have to be output.

```
dsc <- DSC_ThreeStage(macro=DSC_subclu(), micro=DSC_subspaceCluStream())
dsd <- DSD_RandomRBFSubspaceGeneratorEvents() update(dsc, dsd, 1000)
```
**DSC_subspaceCluStream**  
*CluStream for use with DSC_ThreeStage*

**Description**
A version of the DSC_CluStream algorithm that is optimized for use with DSC_ThreeStage. Do not attempt to use this as a standalone stream clustering algorithm.

**Usage**
```python
DSC_subspaceCluStream(timeWindow = 1000, maxNumKernels = 200, 
                       kernelRadifactor = 2, streamSpeed = 1)
```

**Arguments**
- `timeWindow` window for aging the points
- `maxNumKernels` maximum number of microclusters to be produced
- `kernelRadifactor` multiplier for the kernel radius
- `streamSpeed` number of points processed per time unit

---

**DSC_subspaceDenStream**  
*DenStream for use with DSC_ThreeStage*

**Description**
A version of the DSC_DenStream algorithm that is optimized for use with DSC_ThreeStage. Do not attempt to use this as a standalone stream clustering algorithm.

**Usage**
```python
DSC_subspaceDenStream(horizon = 1000, epsilon = 0.04, minPoints = 4, 
                       beta = 0.2, mu = 1, initPoints = 1000, speed = 100)
```

**Arguments**
- `horizon` range of the window
- `epsilon` defines the epsilon neighborhood
- `minPoints` minimal number of points a cluster has to contain
- `beta` multiplier for mu to detect outlier micro-clusters
- `mu` minimal number of points to form a micro-cluster
- `initPoints` number of points used to initialize the algorithm
- `speed` number of data points processed in one time unit
**DSC_ThreeStage**  
*Construct Subspace Stream Clusterers*

**Description**

This function allows you to combine a micro clustering algorithm and a macro clustering algorithm into a single object that can then be used as a normal DSC object. This object can then be used with e.g. `update` to produce a micro clustering of a stream.

**Usage**

```r
DSC_ThreeStage(macro, micro)
```

**Arguments**

- `macro`: a dsc object representing a macroclustering
- `micro`: a dsc object representing a micro clustering, which can be obtained by calling e.g. `DSC_subspaceCluStream`.

**Details**

The microclustering component is implemented as usual, i.e. a `DSC_ThreeStage` object with a `DSC_subspaceCluStream` microclustering component will produce the same microclusters that a normal `CluStream` would have produced. This is the first of the three stages.

Whenever a macro clustering is requested, Multivariate Gaussian Distributions around the positions of the microclusters are used to simulate the original stream. This is the second stage.

Then the macro clustering is performed on the points generated by these distributions using the selected macro clustering algorithm. This is the third stage.

Possible choices for the micro clusterer are `DSC_subspaceDenStream` and `DSC_subspaceCluStream`. Possible macro clusterers are `DSC_clique`, `DSC_p3c`, `DSC_proclus` and `DSC_subclu`. Other clusterers are currently not supported.

**DSD_RandomRBFSubspaceGeneratorEvents**  
*Synthetic Subspace Data Stream*

**Description**

A Random data stream that generated data points that are clustered in several subspaces.
Usage

DSD_RandomRBFSubspaceGeneratorEvents(numAtts = 5, numCluster = 5,
numClusterRange = 0, avgSubspaceSize = 4, avgSubspaceSizeRange = 0,
kernelRadii = 0.07, kernelRadiiRange = 0, numOverlappedCluster = 0,
overlappingDegree = 0, densityRange = 0, noiseLevel = 0.1,
noiseInCluster = F, speed = 200, speedRange = 0,
eventFrequency = 30000, eventMergeSplit = F, eventDeleteCreate = F,
subspaceEventFrequency = 0, decayHorizon = 1000, decayThreshold = 0.1,
modelRandomSeed = sample.int(n = (2^31) - 1, 1),
instanceRandomSeed = sample.int(n = (2^31) - 1, 1))

Arguments

numAtts    the number of dimensions of the data stream.
numCluster the average number of clusters at any point in time.
umClusterRange amount by which the actual number of clusters can deviate from numCluster.
avgSubspaceSize the average number of dimensions in the subspace of a cluster.
avgSubspaceSizeRange the amount by which the number of dimensions can deviate from avgSubspaceSize.
kernelRadii the average radii of the clusters in the model.
kernelRadiiRange the amount by which the radii can deviate from kernelRadii.
numOverlappedCluster the number of overlapped clusters at the beginning of the stream.
overlappingDegree how close the initially overlapped clusters are
densityRange how strongly the amount of points in each cluster differs from each other. 0 means all clusters have the same size. 1 is the maximum value.
noiseLevel amount of noise
noiseInCluster can noise be placed in a cluster?
speed every speed points, the clusters move by 0.01
speedRange speed/Velocity point offset
eventFrequency events happen every eventFrequency points if at least one event is enabled and numClusterRange is set
eventMergeSplit can clusters merge or split?
eventDeleteCreate can clusters be deleted or created?
subspaceEventFrequency Subspace event frequency by each cluster movement destination.
decayHorizon decay horizon
evaluate_subspace

decayThreshold decay horizon threshold
modelRandomSeed number used to seed the RNG for the model
instanceRandomSeed number used to seed the RNG for the instances

DSD_SubspaceARFFStream
Stream Subspace Instances from Disc

Description
A DSD object to stream Data Points from a .arff file.

Usage
DSD_SubspaceARFFStream(file)

Arguments
file a string that contains the path to the file

evaluate_subspace Evaluate Subspace Clusterings

Description
This function evaluates Subspace Clusterings based on data points from a stream.

Usage
evaluate_subspace(dsc, dsd, n = 1000, measures = c("clustering error", "cmm subspace", "entropy subspace", "f1 subspace", "purity", "rand statistic"), alsoTrainOn = F)

Arguments
dsc The clusterer whose current clustering should be evaluated.
dsd The stream from which the data points for evaluation should be drawn.
n How many points to evaluate over
measures A vector of evaluation measures to use. By default, all supported measures are used.
alsoTrainOn This will train the clusterer on the data points before running the evaluation.
**plot_stream_interactive**

Show Stream Clustering.

---

**Description**

A non-animated version of animate_stream_interactive.

**Usage**

```r
plot_stream_interactive(dsc, points,
   launch.browser = getOption("shiny.launch.browser", interactive()))
```

**Arguments**

- **dsc**: a DSC object representing the clustering of a data stream.
- **points**: a data.frame of points that will be plotted along with the clustering.
- **launch.browser**: will be passed on to runApp, so that the visualisation can be shown in e.g. RStudio’s Viewer pane, if this is desired.

---

**subspaceMOA**

Subspace and Projected stream clustering

---

**Description**

The subspaceMOA package is an extension of the stream package, focusing on stream clustering for high-dimensional data streams.

To this end, two new data streams are provided: DSD_RandomRBFSpaceGeneratorEvents, a synthetic data stream for high dimensional data with clusters that only exist in certain subspaces of the data and DSD_SubspaceARFFStream, a utility that reads ARFF files with subspace information.

New subspace stream clustering algorithms for high-dimensional data can be constructed using DSC_ThreeStage in combination with one of DSC_subspaceDenStream and DSC_subspaceCluStream and one of DSC_clique, DSC_p3c, DSC_proclus, DSC_subclu.

Additionally, implementations of the existing subspace stream clustering algorithms DSC_PreDeConStream and DSC_HDDStream are present.

Lastly, functions for interactive visualization with the shiny package are also included: A specific clustering can be plotted using plot_stream_interactive and the clustering can also be animated with the function animate_stream_interactive.
Index

animate_stream_interactive, 2, 2, 11

CLIQUE, 3

data.frame, 11
DSC_clique, 3, 8, 11
DSC_CluStream, 7
DSC_DenStream, 5, 7
DSC_HDDStream, 3, 11
DSC_p3c, 4, 8, 11
DSC_PreDeConStream, 5, 11
DSC_proclus, 6, 8, 11
DSC_subclu, 6, 8, 11
DSC_subspaceCluStream, 7, 8, 11
DSC_subspaceDenStream, 7, 8, 11
DSC_ThreeStage, 3, 4, 6, 7, 8, 11
DSDRandomRBFSubspaceGeneratorEvents, 8, 11
DSD_SubspaceARFFStream, 10, 11

evaluate_subspace, 10

ggplot, 2

P3C, 4
plot_stream_interactive, 11, 11
ProClus, 6

runApp, 2, 11

shiny, 2, 11
SubClu, 6
subspaceMOA, 11

update, 8