Package ‘streamMOA’

May 15, 2019

Version 1.2-2
Date 2019-05-15
Title Interface for MOA Stream Clustering Algorithms
Depends stream (>= 1.1-2), rJava (>= 0.9-0)
Imports graphics, stats, methods
SystemRequirements Java (>= 8)
BugReports https://github.com/mhahsler/streamMOA
License GPL-3
Copyright MOA code in inst/java/moa.jar is Copyright (C) The University of Waikato and distributed under the Apache License, version 2.0. All other code is Copyright (C) Matthew Bolanos, John Forrest and Michael Hahsler
NeedsCompilation no
Author Michael Hahsler [aut, cre, cph], John Forrest [aut, cph], Matthew Bolanos [ctb], Matthias Carnein [ctb]
Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>
Repository CRAN
Date/Publication 2019-05-15 20:10:20 UTC

R topics documented:

DSC_BICO_MOA .................................................. 2
DSC_CluStream_MOA ........................................... 3
DSC_ClusTree_MOA ........................................... 4
DSC_DenStream_MOA ......................................... 6
**DSC_BICO_MOA**

**DSC_DStream_MOA** .................................................. 8
**DSC_MOA** ................................................................ 9
**DSC_StreamKM_MOA** .................................................. 10
**DSD_RandomRBFGeneratorEvents** ................................. 11

Index 13

---

**DSC_BICO_MOA**

*BICO - Fast computation of k-means coresets in a data stream*

---

**Description**

This is an interface to the MOA implementation of BICO. The original BICO implementation by Fichtenberger et al is also available as **DSC_BICO**.

**Usage**

DSC_BICO_MOA(Cluster = 5, Dimensions, MaxClusterFeatures = 1000,
Projections = 10, k = NULL, space = NULL, p = NULL)

**Arguments**

Cluster, k Number of desired centers
Dimensions The number of the dimensions of the input points (stream) need to be specified in advance
MaxClusterFeatures, space Maximum size of the coreset
Projections, p Number of random projections used for the nearest neighbour search

**Details**

BICO maintains a tree which is inspired by the clustering tree of BIRCH, a SIGMOD Test of Time award-winning clustering algorithm. Each node in the tree represents a subset of these points. Instead of storing all points as individual objects, only the number of points, the sum and the squared sum of the subset’s points are stored as key features of each subset. Points are inserted into exactly one node.

**Author(s)**

Matthias Carnein

**References**

Examples

# data with 3 clusters and 2 dimensions
stream <- DSD_Gaussians(k=3, d=2)

# cluster with BICO
bico <- DSC_BICO_MOA(Cluster=3, Dimensions=2)
update(bico, stream, 10000)
bico

# plot micro and macro-clusters
plot(bico, stream, type="both")

---

DSC_CluStream_MOA  

CluStream Data Stream Clusterer

Description

Class implements the CluStream cluster algorithm for data streams.

Usage

DSC_CluStream(m = 100, horizon = 1000, t = 2, k = NULL)
DSC_CluStream_MOA(m = 100, horizon = 1000, t = 2, k = NULL)

Arguments

- \( m \)  
  Defines the maximum number of micro-clusters used in CluStream

- \( \text{horizon} \)  
  Defines the time window to be used in CluStream

- \( t \)  
  Maximal boundary factor (=Kernel radius factor). When deciding to add a new data point to a micro-cluster, the maximum boundary is defined as a factor of \( t \) of the RMS deviation of the data points in the micro-cluster from the centroid.

- \( k \)  
  Number of macro-clusters to produce using weighted k-means. NULL disables automatic reclustering.

Details

This is an interface to the MOA implementation of CluStream.
If \( k \) is specified, then CluStream applies a weighted k-means algorithm for reclustering (see Examples section below).

Value

An object of class DSC_CluStream (subclass of DSC_Micro, DSC_MOA and DSC), or, if \( k \) is not NULL then an object of DSC_TwoStage.
Author(s)

Michael Hahsler and John Forrest

References


See Also

DSC, DSC_Micro, DSC_MOA

Examples

# data with 3 clusters and 5% noise
stream <- DSD_Gaussians(k=3, d=2, noise=.05)

# cluster with CluStream
clustream <- DSC_CluStream(m=50)
update(clustream, stream, 500)
clustream

# plot micro-clusters
plot(clustream, stream)

# plot assignment area (micro-cluster radius)
plot(clustream, stream, assignment=TRUE, weights=FALSE)

# reclustering. Use weighted k-means for CluStream
kmeans <- DSC_Kmeans(k=3, weighted=TRUE)
recluster(kmeans, clustream)
plot(kmeans, stream, type="both")

# use k-means reclustering automatically by specifying k
clustream <- DSC_CluStream(m=50, k=3)
update(clustream, stream, 500)
clustream

plot(clustream, stream, type="both")

DSC_CluTree_MOA  ClusTree Data Stream Clusterer

Description

Interface for the MOA implementation of the ClusTree data stream clustering algorithm.
**Usage**

```r
dsc_clustreeHhorizon = 1PPPL maxheight = 8L lambda = NULLL k = NULLI
dsc_clustree_moaHhorizon = 1PPPL maxheight = 8L lambda = NULLL k = NULLI
```

**Arguments**

- `horizon`: Range of the (time) window.
- `maxHeight`: The maximum height of the tree.
- `lambda`: Number used to override computed lambda (decay).
- `k`: If specified, k-means with k clusters is used for reclustering.

**Details**

ClusTree uses a compact and self-adaptive index structure for maintaining stream summaries.

**Value**

An object of class `DSC_ClusTree` (subclass of `DSC`, `DSC_MOA`, `DSC_Micro`).

**Author(s)**

Michael Hahsler and John Forrest

**References**


http://dx.doi.org/10.1109/ICDM.2009.47


**See Also**

`DSC`, `DSC_Micro`, `DSC_MOA`

**Examples**

```r
# data with 3 clusters and 5% noise
stream <- DSD_Gaussians(k=3, d=2, noise=0.05)

# Use automatically the k-means reclusterer with k=3 to create macro clusters
clustree <- DSC_ClusTree(maxHeight=3, k = 3)
update(clustree, stream, 500)

# plot micro-clusters
plot(clustree, stream, , type = "both")
```
# create a two stage clustering using ClusTree and reachability reclustering
CTxReach <- DSC_TwoStage(
    micro=DSC_Clustree(maxHeight=3),
    macro=DSC_Reachability(epsilon = .15)
)
CTxReach
update(CTxReach, stream, 1000)
plot(CTxReach, stream, type = "both")

---

**DSC_DenStream_MOA**  
*DenStream Data Stream Clusterer*

**Description**

Interface for the DenStream cluster algorithm for data streams implemented in MOA.

**Usage**

```r
DSC_DenStream(epsilon, mu = 1, beta = 0.2, lambda = 0.001, 
    initPoints = 100, offline = 2, processingSpeed=1, recluster = TRUE, k=NULL)
DSC_DenStream_MOA(epsilon, mu = 1, beta = 0.2, lambda = 0.001, 
    initPoints = 100, offline = 2, processingSpeed=1, recluster = TRUE, k=NULL)
```

**Arguments**

- **epsilon**: defines the epsilon neighbourhood which is the maximal radius of micro-clusters ($r \leq \text{epsilon}$). Range: 0 to 1.
- **mu**: minpoints as the weight $w$ a core-micro-clusters needs to be created ($w \geq \text{mu}$). Range: 0 to max(int).
- **beta**: multiplier for mu to detect outlier micro-clusters given their weight $w$ ($w < \text{beta} \times \text{mu}$). Range: 0 to 1
- **lambda**: decay constant.
- **initPoints**: number of points to use for initialization via DBSCAN.
- **offline**: offline multiplier for epsilon. Range: between 2 and 20). Used for reachability reclustering
- **processingSpeed**: Number of incoming points per time unit (important for decay). Range: between 1 and 1000.
- **recluster**: logical; should the offline DBSCAN-based (i.e., reachability at a distance of epsilon) be performed?
- **k**: integer; tries to automatically chooses offline to find k macro-clusters.
Details

DenStream applies reachability (from DBSCAN) between micro-clusters for reclustering using epsilon x offline (defaults to 2) as the reachability threshold.

If k is specified it automatically chooses the reachability threshold to find k clusters. This is achieved using single-link hierarchical clustering.

Value

An object of class DSC_DenStream (subclass of DSC, DSC_MOA, DSC_Micro) or, for recluster=TRUE, an object of class DSC_TwoStage.

Author(s)

Michael Hahsler and John Forrest

References


See Also

DSC, DSC_Micro, DSC_MOA

Examples

# data with 3 clusters and 5% noise
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

# use Den-Stream with reachability reclustering
denstream <- DSC_DenStream(epsilon = .05)
update(denstream, stream, 500)
denstream

# plot macro-clusters
plot(denstream, stream)

# plot micro-cluster
plot(denstream, stream, type = "micro")

# show micro and macro-clusters
plot(denstream, stream, type = "both")

# reclustering. Choose reclustering reachability threshold automatically to find 3 clusters
denstream2 <- DSC_DenStream(epsilon = .05, k = 3)
update(denstream2, stream, 500)
DSC_DStream_MOA

D-Stream Data Stream Clustering Algorithm

Description

This is an interface to the MOA implementation of D-Stream. A C++ implementation (including reclustering with attraction) is available as DSC_DStream.

Usage

DSC_DStream_MOA(decayFactor = 0.998, Cm = 3, Cl = 0.8, Beta = 0.3)

Arguments

- **decayFactor**: The decay factor
- **Cm**: Controls the threshold for dense grids
- **Cl**: Controls the threshold for sparse grids
- **Beta**: Adjusts the window of protection for renaming previously deleted grids as sporadic

Details

D-Stream creates an equally spaced grid and estimates the density in each grid cell using the count of points falling in the cells. Grid cells are classified based on density into dense, transitional and sporadic cells. The density is faded after every new point by a decay factor.

**Note**: The MOA implementation of D-Stream currently does not return micro clusters.

Author(s)

Matthias Carnein

References

- Li Tu and Yixin Chen. 2009. Stream data clustering based on grid density and attraction. ACM Transactions on Knowledge Discovery from Data, 3(3), Article 12 (July 2009), 27 pages.
Examples

# data with 2 clusters in 2 dimensions
stream = dsd_gaussians(2,2, mu = rbind(c(-10,-10), c(10,10)))

# cluster with D-Stream
dstream <- DSC_DStream_MOA(decayFactor=0.998)
update(dstream, stream, 10000)
dstream

# plot macro-clusters
plot(dstream, stream, type= "macro")

---

**DSC_MOA**

**DSC_MOA Class**

**Description**

An abstract class that inherits from the base class DSC and provides the common functions needed to interface MOA clusterers.

**Details**

DSC_MOA classes operate in a different way in that the centers of the micro-clusters have to be extracted from the underlying Java object. This is done by using rJava to perform method calls directly in the JRI and converting the multi-dimensional Java array into a local R data type.

**Author(s)**

Michael Hahsler and John Forrest

**References**


**See Also**

DSC
**Description**

This is an interface to the MOA implementation of streamKM++.

**Usage**

DSC_StreamKM(sizeCoreset = 10000, numClusters = 5, length = 100000L)
DSC_StreamKM_MOA(sizeCoreset = 10000, numClusters = 5, length = 100000L)

**Arguments**

- **sizeCoreset**  
  Size of the coreset
- **numClusters**  
  Number of clusters to compute
- **length**  
  Length of the data stream

**Details**

streamKM++ uses a tree-based sampling strategy to obtain a small weighted sample of the stream called coreset. Upon reclustering, the algorithm applies the k-means++ algorithm to find a given number of centres in the coreset.

**Note:** This implementation currently does not return micro-clusters.

**Author(s)**

Matthias Carnein

**References**


**Examples**

```r
# data with 3 clusters
stream <- DSD_Gaussians(k=3, d=2)

# cluster with streamKM++
streamkm <- DSC_StreamKM(sizeCoreset=10000, numClusters=3, length=100000)
update(streamkm, stream, 10000)
streamkm

# plot macro-clusters
plot(streamkm, stream, type="macro")
```
DSD\_RandomRBFGeneratorEvents

Random RBF Generator Events Data Stream Generator

Description
A class that generates random data based on RandomRBFGeneratorEvents implemented in MOA.

Usage
DSD\_RandomRBFGeneratorEvents(k = 3, d = 2, numClusterRange = 3L,
kernelRadius = 0.07, kernelRadiusRange = 0, densityRange = 0,
speed = 100L, speedRange = 0L, noiseLevel = 0.1,
noiseInCluster = FALSE, eventFrequency = 30000L,
eventMergeSplitOption = FALSE, eventDeleteCreate = FALSE,
modelSeed = NULL, instanceSeed = NULL)

Arguments
k The average number of centroids in the model.
d The dimensionality of the data.
numClusterRange Range for number of clusters.
kernelRadius The average radius of the micro-clusters.
kernelRadiusRange Deviation of the number of centroids in the model.
densityRange Density range.
speed Kernels move a predefined distance of 0.01 every X points.
speedRange Speed/Velocity point offset.
oiseLevel Noise level.
oiseInCluster Allow noise to be placed within a cluster.
eventFrequency Frequency of events.
eventMergeSplitOption Merge and split?
eventDeleteCreate Delete and create?
modelSeed Random seed for the model.
instanceSeed Random seed for the instances.
Details

There are an assortment of parameters available for the underlying MOA data structure, however, we have currently limited the available parameters to the arguments above. Currently the modelSeed and instanceSeed are set to default values every time a DSD_MOA is created, therefore the generated data will be the same. Because of this, it is important to set the seed manually when different data is needed.

The default behavior is to create a data stream with 3 clusters and concept drift. The locations of the clusters will change slightly, and they will merge with one another as time progresses.

Value

An object of class DSD_RandomRBFGeneratorEvent (subclass of DSD_MOA, DSD).

Author(s)

Michael Hahsler and John Forrest

References


See Also

DSD

Examples

stream <- DSD_RandomRBFGeneratorEvents()
get_points(stream, 10, class=TRUE)

## Not run:
animate_data(stream, n=5000, pointInterval=100, xlim=c(0,1), ylim=c(0,1))

## End(Not run)
Index

CluStream (DSC_Clustream_MOA), 3
clustream (DSC_Clustream_MOA), 3
ClusTree (DSC_Clustree_MOA), 4
clustree (DSC_Clustree_MOA), 4

DenStream (DSC_DenStream_MOA), 6
denstream (DSC_DenStream_MOA), 6
DSC, 4, 5, 7, 9
DSC_BICO, 2
DSC_BICO_MOA, 2
DSC_Clustream (DSC_Clustream_MOA), 3
DSC_Clustream_MOA, 3
DSC_Clustree (DSC_Clustree_MOA), 4
DSC_Clustree_MOA, 4
DSC_DenStream (DSC_DenStream_MOA), 6
DSC_DenStream_MOA, 6
DSC_DStream, 8
DSC_DStream_MOA, 8
DSC_Micro, 4, 5, 7
DSC_MOA, 4, 5, 7, 9
DSC_StreamKM (DSC_StreamKM_MOA), 10
DSC_StreamKM_MOA, 10
DSD, 12
DSD_RandomRBFGeneratorEvents, 11