Package ‘streamMOA’

September 3, 2022

Version 1.3-0
Date 2022-09-03
Encoding UTF-8

Title Interface for MOA Stream Clustering Algorithms


Depends stream (>= 2.0-0), rJava (>= 1.0-1)
Imports graphics, stats, methods
Suggests RMOA (>= 1.1.0)

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

RoxygenNote 7.2.1

NeedsCompilation no

Author Michael Hahsler [aut, cre, cph]
(https://orcid.org/0000-0003-2716-1405),
John Forrest [aut, cph],
Matthew Bolanos [ctb],
Matthias Carnein [ctb],
Dalibor Krleža [ctb]

Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>

Repository CRAN

Date/Publication 2022-09-03 21:50:02 UTC
**R topics documented:**

- streamMOA-package ................................................................. 2
- DSClassifier_MOA ................................................................. 2
- DSC_BICO_MOA ........................................................................ 4
- DSC_CluStream ....................................................................... 5
- DSC_ClusTree ......................................................................... 7
- DSC_DenStream ...................................................................... 8
- DSC_DStream_MOA ................................................................. 10
- DSC_MCOD ............................................................................... 11
- DSC_MOA ................................................................................ 13
- DSC_StreamKM ..................................................................... 14
- DSD_MOA ................................................................................ 15
- DSD_RandomRBFGeneratorEvents ......................................... 16

### Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>streamMOA-package</td>
<td>2</td>
</tr>
<tr>
<td>DSClassifier_MOA</td>
<td>2</td>
</tr>
<tr>
<td>DSC_BICO_MOA</td>
<td>4</td>
</tr>
<tr>
<td>DSC_CluStream</td>
<td>5</td>
</tr>
<tr>
<td>DSC_ClusTree</td>
<td>7</td>
</tr>
<tr>
<td>DSC_DenStream</td>
<td>8</td>
</tr>
<tr>
<td>DSC_DStream_MOA</td>
<td>10</td>
</tr>
<tr>
<td>DSC_MCOD</td>
<td>11</td>
</tr>
<tr>
<td>DSC_MOA</td>
<td>13</td>
</tr>
<tr>
<td>DSC_StreamKM</td>
<td>14</td>
</tr>
<tr>
<td>DSD_MOA</td>
<td>15</td>
</tr>
<tr>
<td>DSD_RandomRBFGeneratorEvents</td>
<td>16</td>
</tr>
</tbody>
</table>

---

**streamMOA-package**

*streamMOA: Interface for MOA Stream Clustering Algorithms*

**Description**


**Author(s)**

Michael Hahsler

---

**DSClassifier_MOA**

*DSClassifier_MOA – MOA-based Stream Classifiers*

**Description**

Interface for MOA-based stream classification methods based on package RMOA.

**Usage**

DSClassifier_MOA(formula, RMOA_classifier)

```r
## S3 method for class 'DSClassifier_MOA'
update(object, dsd, n = 1, verbose = FALSE, block = 1000L, ...)

## S3 method for class 'DSClassifier_MOA'
predict(object, newdata, type = "response", ...)```

---
DSClassifier_MOA provides an interface to MOA-based stream classifiers using package RMOA. RMOA provides access to MOAs stream classifiers in the following groups:

- RMOA::MOA_classification_trees
- RMOA::MOA_classification_bayes
- RMOA::MOA_classification_ensemblelearning

Subsequent calls to update() update the current model.

Value
An object of class DSClassifier_MOA

Author(s)
Michael Hahsler

References

Examples
## Not run:
library(streamMOA)
library(RMOA)

# create a data stream for the iris dataset
data <- iris[sample(nrow(iris)), ]
stream <- DSD_Memory(data)
stream

# define the stream classifier
cl <- DSClassifier_MOA(
Species ~ Sepal.Length + Sepal.Width + Petal.Length,
RMOA::HoeffdingTree()
)

cl

# update the classifier with 100 points from the stream
update(cl, stream, 100)

# predict the class for the next 50 points
newdata <- get_points(stream, n = 50)
pr <- predict(cl, newdata)
pr

table(pr, newdata$Species)

## End(Not run)

---

**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 `stream::DSC_BICO`.

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster, k</td>
<td>Number of desired centers</td>
</tr>
<tr>
<td>Dimensions</td>
<td>The number of the dimensions of the input points (stream) need to be specified in advance</td>
</tr>
</tbody>
</table>
MaxClusterFeatures, space
Maximum size of the coreset
Projections, p
Number of random projections used for the nearest neighbor 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

See Also
Other DSC_MOA: DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_MOA(), DSC_StreamKM()

Examples
# data with 3 clusters and 2 dimensions
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

# cluster with BICO
bico <- DSC_BICO_MOA(Cluster = 3, Dimensions = 2)
update(bico, stream, 100)
bico

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

- **m**: Defines the maximum number of micro-clusters used in CluStream.
- **horizon**: Defines the time window to be used in CluStream.
- **t**: Maximal boundary factor (i.e., the 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.

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`).

Author(s)

Michael Hahsler and John Forrest

References


See Also

Other DSC_MOA: `DSC_BICO_MOA()`, `DSC_ClusTree()`, `DSC_DStream_MOA()`, `DSC_DenStream()`, `DSC_MCOD()`, `DSC_MOA()`.

Examples

```r
# data with 3 clusters and 5% noise
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = .05)

# cluster with CluStream
clustream <- DSC_CluStream(m = 50, horizon = 100, k = 3)
update(clustream, stream, 500)
clustream
plot(clustream, stream, type = "both")
```
**Description**

Interface for the MOA implementation of the ClusTree data stream clustering algorithm (Kranen et al, 2009).

**Usage**

```r
DSC_ClusTree(horizon = 1000, maxHeight = 8, lambda = NULL, k = NULL)
```

**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**


**Value**

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

**Author(s)**

Michael Hahsler and John Forrest

**References**


**See Also**

Other DSC_MOA: `DSC_BICO_MOA()`, `DSC_CluStream()`, `DSC_DStream_MOA()`, `DSC_DenStream()`, `DSC_MCOD()`, `DSC_MOA()`, `DSC_StreamKM()`
Examples

# data with 3 clusters
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

clustree <- DSC_ClusTree(maxHeight = 3)
update(clustree, stream, 500)
clustree

plot(clustree, stream)

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

plot(clustree, stream, type = “both”)

---

DSC_DenStream  
DenStream Data Stream Clusterer

Description

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

Usage

DSC_DenStream(
  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<=epsilon). Range: 0 to 1.

mu  
minpoints as the weight w a core-micro-clusters needs to be created (w>=mu). Range: 0 to max(int).

beta  
multiplier for mu to detect outlier micro-clusters given their weight w (w<beta x mu). Range: 0 to 1
DSC_DenStream

lambda: decay constant.
initPoints: number of points to use for initialization via DBSCAN.
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 reachbility (from DBSCAN) between micro-clusters for reclustering using epsilon \times \text{offline} \ (\text{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

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_MCOD(), DSC_MOA(), DSC_StreamKM()

Examples

# data with 3 clusters and 5% noise
set.seed(1000)
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, type = "both")

# 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 4 clusters
denstream2 <- DSC_DenStream(epsilon = .05, k = 4)
update(denstream2, stream, 500)
plot(denstream2, stream, type = "both")

---

**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 `stream::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.

**Notes:**

- This implementation seems to use a 1 x 1 grid and therefore the range is increased in the example.  
- The MOA implementation of D-Stream currently does not return micro clusters.
**Micro-cluster Continuous Outlier Detector (MCOD)**

**Description**

Class interfaces the MOA implementation of the MCOD algorithm for distance-based data stream outlier detection.

**Usage**

```r
DSC_MCOD(r = 0.1, t = 50, w = 1000, recheck_outliers = FALSE)
DSOutlier_MCOD(r = 0.1, t = 50, w = 1000, recheck_outliers = TRUE)
get_outlier_positions(x, ...)
recheck_outlier(x, outlier_correlated_id, ...)
clean_outliers(x, ...)
```
**Arguments**

- **r**: Defines the micro-cluster radius.
- **t**: Defines the number of neighbors (k in the article).
- **w**: Defines the window width in data points.
- **recheck_outliers**: Defines that the MCOD algorithm allows re-checking of detected outliers.
- **x**: a DSC_MCOD object.
- **...**: further arguments are currently ignored.
- **outlier_correlated_id**: ids of outliers.

**Details**

The algorithm detects density-based outliers. An object \( x \) is defined to be an outlier if there are less than \( t \) objects lying at distance at most \( r \) from \( x \).

Outliers are stored and can be retrieved using `get_outlier_position()` and `recheck_outlier()`.

**Note**: The implementation updates the clustering when `predict()` is called.

**Value**

An object of class `DSC_MCOD` (subclass of `DSOutlier`, `DSC_Micro`, `DSC_MOA` and `DSC`).

**Functions**

- `get_outlier_positions()`: Returns spatial positions of all current outliers.
- `recheck_outlier()`: DSC_MCOD Re-checks the outlier having `outlier_correlated_id`. If this object is still an outlier, the method returns `TRUE`.
- `clean_outliers()`: forget detected outliers from the outlier detector (currently not implemented).

**Author(s)**

Dalibor Krleža

**References**


**See Also**

Other DSC_MOA: `DSC_BICO_MOA()`, `DSC_CluStream()`, `DSC_ClusTree()`, `DSC_DStream_MOA()`, `DSC_DenStream()`, `DSC_MOA()`, `DSC_StreamKM()`
Examples

# Example 1: Clustering with MCOD
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)
mcod <- DSC_MCOD(r = .1, t = 3, w = 100)
update(mcod, stream, 100)
mcod

plot(mcod, stream, n = 100)

# Example 2: Predict outliers (have a class label of NA)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)
mcod <- DSOutlier_MCOD(r = .1, t = 3, w = 100)
update(mcod, stream, 100)

plot(mcod, stream, n = 100)

# MCOD can retrieve the outliers
get_outlier_positions(mcod)

# Example 3: evaluate on a stream
evaluate_static(mcod, stream, n = 100, type = "micro",
measure = c("crand", "noisePrecision", "outlierjaccard"))

---

**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.

**Usage**

DSC_MOA(...)

**Arguments**

... further arguments.

**Details**

DSC_MOA is a subclass of DSC for MOA-based clusterers. 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.

**Note:** The formula interface is currently not implemented for MOA-based clusterers. Use DSF to select features instead.
Author(s)

Michael Hahsler and John Forrest

References


See Also

DSC

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_StreamKM()

DSC_StreamKM streamKM++

Description

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

Usage

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

Arguments

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

Further arguments are passed on to DSC_Kmeans for reclustering.

Details

streamKM++ uses a tree-based sampling strategy to obtain a small weighted sample of the stream called coreset. The MOA implementation applies the k-means++ algorithm to find a given number of centers in the coreset.

Notes:

- The cluster can only cluster the number of points specified in length and then produces an ArrayIndexOutOfBoundsException error.
- The coreset (micro-clusters are not accessible), only the macro-clusters can be requested.

Author(s)

Matthias Carnein
References


See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_MOA()

Examples

```r
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

# cluster with streamKM++
streamkm <- DSC_StreamKM(sizeCoreset = 100, numClusters = 3, length = 1000)
update(streamkm, stream, 100)
streamkm

# plot macro-clusters (no access to micro-clusters)
plot(streamkm, stream)
```

DSD_MOA

Abstract base class for MOA-based data stream generators directly inherits from DSD.

Usage

DSD_MOA()

Arguments

... further arguments.

Value

The abstract class cannot be instantiated and produces an error.

Author(s)

Michael Hahsler
References

See Also
Other DSD_MOA: `DSD_RandomRBFGeneratorEvents()`

Examples
DSD()

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.
DSD_RandomRBFGeneratorEvents

- 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.
- noiseLevel: Noise level.
- noiseInCluster: 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

Other DSD_MOA: DSD_MOA()
Examples

```r
stream <- DSD_RandomRBFGeneratorEvents()
get_points(stream, 10)

if (interactive()) {
animate_data(stream, n = 5000, horizon = 100, xlim = c(0, 1), ylim = c(0, 1))
}
```
Index

* DSC_MOA
  DSC_BICO_MOA, 4
  DSC_CluStream, 5
  DSC_ClusTree, 7
  DSC_DenStream, 8
  DSC_DStream_MOA, 10
  DSC_MCOD, 11
  DSC_MOA, 13
  DSC_StreamKM, 14
* DSClassifier_MOA
  DSClassifier_MOA, 2
* DSD_MOA
  DSD_MOA, 15
  DSD_RandomRBFGeneratorEvents, 16
* DSOutlier_MOA
  DSOutlier_MOA, 11

DSC, 6, 7, 9, 12–14
DSC_BICO_MOA, 4, 6, 7, 9, 11, 12, 14, 15
DSC_CluStream, 5, 5, 7, 9, 11, 12, 14, 15
DSC_ClusTree, 5, 6, 7, 9, 11, 12, 14, 15
DSC_DenStream, 5–7, 8, 11, 12, 14, 15
DSC_DStream_MOA (DSC_DenStream), 8
DSC_DStream_MOA, 5–7, 9, 10, 12, 14, 15
DSC_Kmeans, 14
DSC_MCOD, 5–7, 9, 11, 11, 14, 15
DSC_MCOD_MOA (DSC_MCOD), 11
DSC_Micro, 6, 7, 9, 12
DSC_MOA, 5–7, 9, 11, 12, 13, 15
DSC_StreamKM, 5–7, 9, 11, 12, 14, 14
DSC_TwoStage, 9

DSClassifier_MOA, 2
DSD, 15, 17
DSD_MOA, 15, 17
DSD_RandomRBFGeneratorEvents, 16, 16
DSF, 13
DSOutlier, 12
DSOutlier_MCOD (DSC_MCOD), 11
DSOutlier_MCOD_MOA (DSC_MCOD), 11
get_outlier_positions (DSC_MCOD), 11
MCOD (DSC_MCOD), 11
predict(), 12
predict.DSClassifier_MOA (DSClassifier_MOA), 2
recheck_outlier (DSC_MCOD), 11
RMOA::MOA_classification_bayes, 3
RMOA::MOA_classification_ensemblelearning, 3
RMOA::MOA_classification_trees, 3
RMOA::predict.MOA_trainedmodel(), 3
stream::DSC_BICO, 4
stream::DSC_DStream, 10
StreamKM (DSC_StreamKM), 14
streamkm (DSC_StreamKM), 14
streamMOA-package, 2
update.DSClassifier_MOA (DSClassifier_MOA), 2

19