OutlierDenStream Documentation

Release 0.0.1

Andrian Putina

Oct 13, 2020

Contents:

CHAPTER 1

OutlierDenStream

1.1 DEPRECATED!

Please use https://github.com/anrputina/oadds

CHAPTER 2

Installation

2.1 Stable release

Release on pip index coming soon. Please use the sources

2.2 From sources

The sources for OutlierDenStream can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/anrputina/outlierdenstream
```

Or download the tarball:

```
$ curl  -OL https://github.com/anrputina/outlierdenstream/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use OutlierDenStream in a project:

```
from outlierdenstream import Sample, OutlierDenStream
```

Initialize OutlierDenStream object:

```
ods = OutlierDenStream(lamb=0.03, epsilon='auto', beta=0.03, mu='auto',␣
↪startingBuffer=bufferDf, tp=12)
ods.runInitialization()
```

Fit each sample of the dataset with:

```
for row in dataset:
    sample = Sample(row, timestamp)
    result = ods.runOnNewSample(sample)
```

The algorithm returns `True` (outlier) if it is not able to merge the new sample to an existing core-micro-cluster or merges the sample to an existing outlier-micro-cluster. Returns `False` (normal) otherwise.

CHAPTER 4

Documentation

4.1 Sample

class outlierdenstream.**Sample**(*value*, *timestamp: int*)
Each record of the stream has to be declared as a *Sample* class.

Parameters

- **value** – the *values* of the current sample.
- **timestamp** – the *timestamp* of current sample.

getValue()

Returns value

setMicroClusterNumber(*microClusterNumber: int*)
Assign to each sample the microClusterNumber in which was merged.

Set microClusterNumber

setTimestamp(*timestamp: int*)

Set timestamp

4.2 Micro-Cluster

class outlierdenstream.**MicroCluster**(*currenttimestamp*, *lamb*, *clusterNumber*)
Micro-Cluster class

Parameters

- **currenttimestamp** – the *timestamp* in which the cluster is created.
- **lamb** – the *lamb* parameter used as decay factor.
- **clusterNumber** – the *number* of the micro-cluster.

getCenter()

Returns the *center* of the micro-cluster.

getRadius()

Returns the *radius* of the micro-cluster.

insertSample(*sample*, *timestamp=0*)
Adds a sample to a micro-cluster. Updates the variables of the micro-cluster with *updateRealTimeWeight()* and *updateRealTimeLSandSS()*

Parameters

- **sample** – the *sample* object
- **timestamp** – deprecated, not needed anymore. Will be removed in the next versions.

noNewSamples()
Updates the *Weighted Linear Sum* (WLS), the *Weighted Squared Sum* (WSS) and the weight of the micro-cluster when no new samples are merged.

updateRealTimeLSandSS(*sample*)
Updates the *Weighted Linear Sum* (WLS), the *Weighted Squared Sum* (WSS), the *center* and the *radius* of the micro-cluster when a new sample is merged.

Parameters **sample** – the *sample* to merge into the micro-cluster.

updateRealTimeWeight()
Updates the Weight of the micro-cluster by the fading factor and increases it by 1.

4.3 Cluster

class outlierdenstream.**Cluster**
Cluster class. Contains the list of the micro-cluster and the number of micro-clusters.

insert(*mc*)
Inserts a micro-cluster into the cluster

Parameters **mc** – the *micro-cluster* to be added to the list of the micro-clusters that make up the cluster.

Increases the counter of micro-clusters into the cluster by 1.

4.4 OutlierDenStream

class outlierdenstream.**OutlierDenStream**(*lamb*, *epsilon=1*, *minPts=1*, *beta=1*, *mu=1*, *numberInitialSamples=None*, *startingBuffer=None*, *tp=60*, *radiusFactor=1*)
OutlierDenStream class.

Parameters

- **lamb** – the *lambda* parameter - fading factor
- **epsilon** – the *epsilon* parameter
- **beta** – the *beta* parameter
- **mu** – the *mu* parameter

- **numberInitialSamples** – samples to use as initial buffer
- **startgingBuffer** – initial *buffer* on which apply DBScan or use it as unique class.
- **tp** – frequency at which to apply the pruning strategy and remove old micro-clusters.

initDBScan()
Init with DBSCAN

initWithoutDBScan()
Produces a micro-cluster merging all the samples passed into the initial buffer

If *epsilon* is auto computes *epsilon* as the maxium radius obtained from these initial samples.

resetLearningImpl()
Initializes two empty *Cluster* as a p-micro-cluter list and o-micro-cluster list.

If *mu* is *auto* computes the value

runDBSCanInitialization()
Initializes the variables of the main algorithm with the methods *resetLearningImpl()* and *initDBScan()*

runInitialization()
Initializes the variables of the main algorithm with the methods *resetLearningImpl()* and *initWithoutDBScan()*

runOnNewSample(*sample*)
Performs the basic DenStream procedure for merging new samples.

- Try to merge the sample to the closest core-micro-cluster (or)
- Try to merge the sample to the closest outlier-micro-cluster (or)
- Generate new outlier-micro-cluster by the sample

Parameters **sample** – the new available *sample* in the stream

Returns `False` if the sample is merged to an existing core-micro-cluster otherwise `True` meaning "anomalous" sample.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/anrputina/outlierdenstream/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

OutlierDenStream could always use more documentation, whether as part of the official OutlierDenStream docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/anrputina/outlierdenstream/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *outlierdenstream* for local development.

1. Fork the *outlierdenstream* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/outlierdenstream.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv outlierdenstream
$ cd outlierdenstream/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 outlierdenstream tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/anrputina/outlierdenstream/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_outlierdenstream
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Andrian Putina <anr.putina@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.0.1 (2019-04-11)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index