Anomaly Detection in Data streams using MOA

G Sandhya Madhuri[a] , Dr. M. Usha Rani[b]

[a] Research Scholar, [b] Research Supervisor
[a, b] Dept. of Computer Science, SPMVV, Tirupati, India
[a] sandhyamadhuri@gmail.com, [b] musha_rohan@yahoo.com

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: Anomaly means anything which deviates from normal. It can be a credit card fraud or sensor alarm or a signal from a condition monitoring device. A problem like anomaly arises when we try to monitor the unusual behaviour of a machine. More number of outliers means the machine needs to be inspected. Anomaly detection in static data can be entirely different from that of streaming data.

We have some issues in anomaly detection in streaming data when compared to static data. If any off – line algorithms attempt to find anomalies in streams, it has to store the entire stream for analysis. So, there is a high probability that it will run out of memory space.

Also, streams can be infinite and evolving over a period of time because of which maintenance of high detection accuracy becomes almost impossible. In this paper we will discuss about anomaly detection in data streams and using MOA (Massive Online Analysis) tool we will analyse which algorithm derives best results.

Keywords: Massive Online Analysis, data streams, outliers, outlier detection algorithms

1. Introduction

Data streams are universal, ranging from sensor data to web data and clickstream data. Subsequently, there is a rich and increasing body of work onstream data mining.

Anomaly detection using simulation helps us examine the anomaly examples from big data source. In order to increase the speed of the processing time to handle massive datasets, in this paper we have demonstrated the experiments conducted on advanced distant-based outlier detection algorithms to inspect the most effective algorithms using MOA. The algorithms used in this study are Continuous Outlier Detection (COD), Micro-Cluster based COD or MCOD, Exact STream OutlierR Miner (Exact STORM), Approximate Stream Outlier Miner (Approx STORM), AbstractC algorithms. The results demonstrate MCOD algorithm can surpass other algorithms in terms of processing time and precise anomaly detection.

We have MOA (Massive Online Analysis) an open source frame work for anomaly detection in data streams. The key advantage of MOA is that it provides many recently developed data stream algorithms, with learners for multi-label classification, graph mining and outlier detection.

2. Moa Framework

The aim of MOA is a standard framework for running experiments in the data stream mining context by proving

- storable settings for data streams (real and synthetic) for repeatable experiments
- a set of existing algorithms and measures form the literature for comparison and
- an easily extendable framework for new streams, algorithms and evaluation methods.

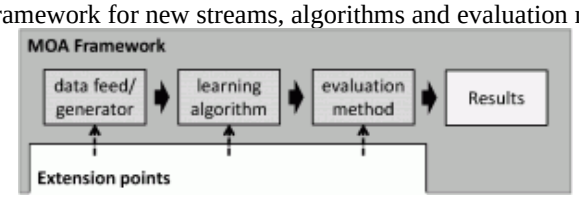

Fig 1: MOA workflow

The workflow in MOA follows the simple outline as shown below:

- **First** a data stream (feed, generator) is chosen and configured.
- **Second** an algorithm (e.g. a classifier) is selected and its parameters are set.
- **Third** the evaluation method or measure is chosen.
- **Finally** the results are acquired after executing the task.

MOA supports stream classification, stream clustering, and outlier detection[1].

3. Outlier Detection In Moa

The Outliers graphical interface can be seen below. The image shows the environment where we setup any two algorithms to analyse on a data stream that is generated by MOA using RandoRBFGeneratorEvents data stream generator.

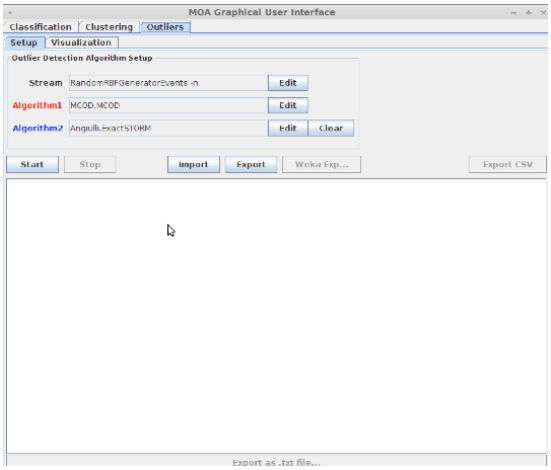

Fig 2: MOA Outliers: Setup

The setup contains of three parameters:

1. Data stream
2. Algorithm 1
3. Algorithm 2

We have to use Edit option to change any of the above mentioned parameters.

Now, let us see the editing options of Algorithms.

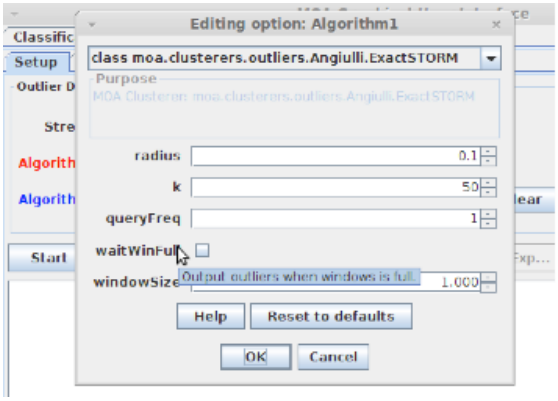

Fig: 3 Setting the parameters for each algorithm

In the above figure we can observe that to analyse the any two algorithms, the user should first select the algorithm and then change the values of each parameter of the chosen algorithm. If we move the cursor over the names of the parameter, a message box appears describing each parameter[2].

Once the algorithms are selected and parameters are set. The user needs to observe the Visualisation tab, where he can visually see both the algorithms result. The below figure shows the Visualisation tab before the user presses start button.

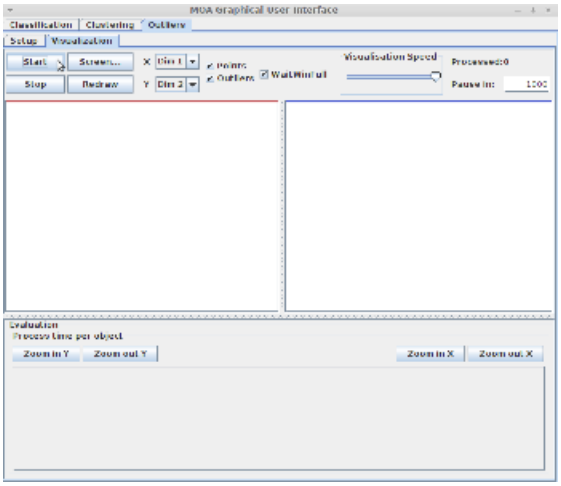

Fig: 4 Visualisation tab

In this tab, the user can change many options like display options i.e. either points or outliers. Also, options like wait till the window is full so as to display outliers. The user can also change the visualisation speed, dimensions of the points that are displayed and the value of the pause interval.

Now, let us observe *Fig*: 5 in which the visualisation of each point in the data stream is displayed in the two areas (screens) allocated for each algorithm. The screen with red colour represents the Algorithm 1 and the screen with blue colour represents Algorithm 2.

The graph at the bottom of the visualisation window shows the time taken by the algorithm to process each data point in the stream.

y-axis represents time per object

x-axis represent eachwindow slide.

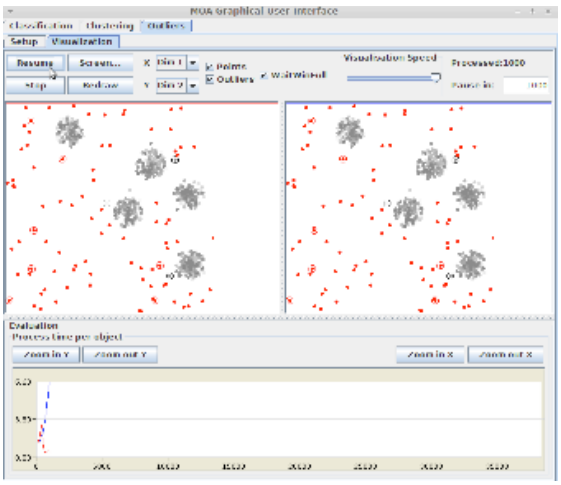

Fig: 5 Visualisation tab displaying the data points and outliers

After 1000 window slide, that means the pause interval specified by the user, the visualisation pauses for the user to notice the outcome from each algorithm. We can observe that red coloured points are the outliers and the grey coloured points are inliers or normal points.

The user can also observe a red circle around some outlier points. This indicates that this particular point has been changed from an inlier to an outlier. In the same way we can also observe a black circle around some grey points which indicate that this particular point was an outlier and now has been changed to an inlier. Observe *Fig*: 6

If we click on any outlier point a popup window is displayed showing information about that point. Such information is specific to algorithm[3].

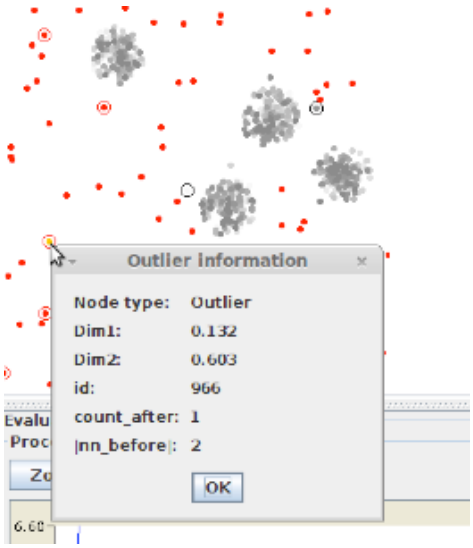

Fig: 6 Outlier information

We can continue the analysis by pressing resume button, and observe the visualisation of outliers. To stop the visualisation we have to press stop button.

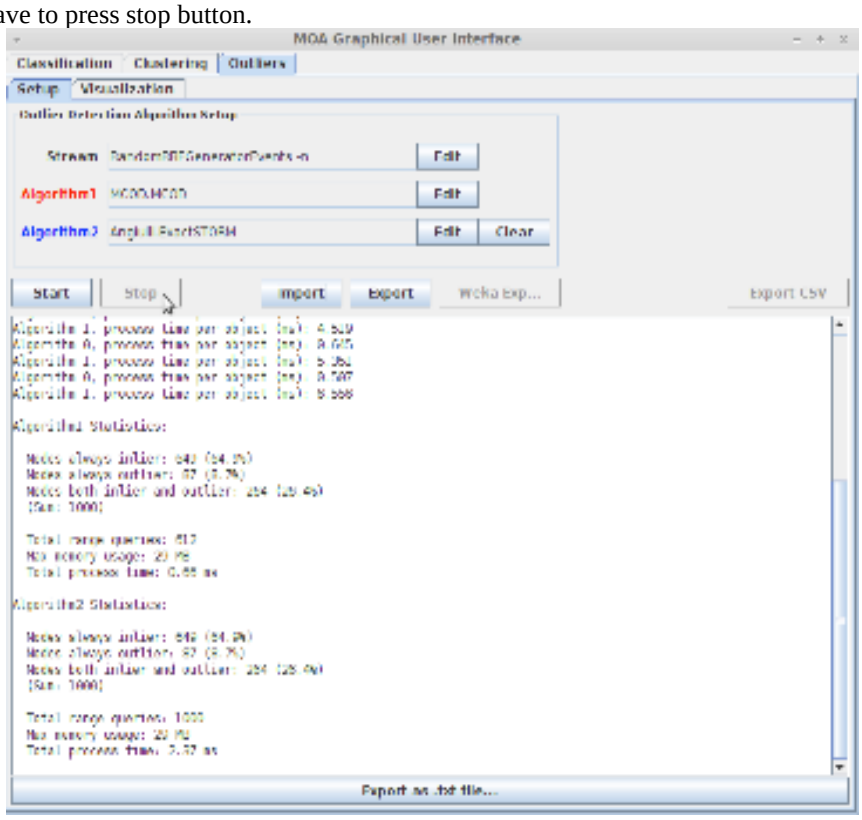

Fig: 7 Setup tab with statistical information

After this, we can observe some statistical information about each algorithm in the setup tab as in the *Fig*:7.

The information includes processing time, Total range queries, Maximum memory usage, information about nodes that are inlier and outlier.

4. Analysis Of Data Stream In Moa

We have taken the stream generator by RandomRBFGeneratorEvents in MOA tool and tested MCOD algorithm with all the other algorithms[iii]. Given below is the screenshot of the parameters set for the data stream generator.

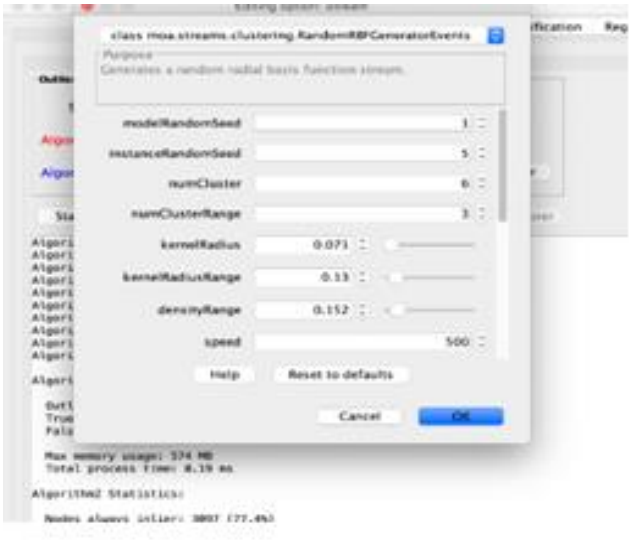

Fig: 8 Parameters in Stream Generator

In the above figure Fig: 8 we have set the parameters such as number of clusters that can be formed are 6, the Kernel radius as 0.071, Range of the radius as 0.13 etc. Using these parameters, a data stream is generated with sliding window size as 1000[4].

Now let us examine the results of each algorithm we have tested with MCOD. Observe the figures below.

1. MCOD and Abstract C algorithms

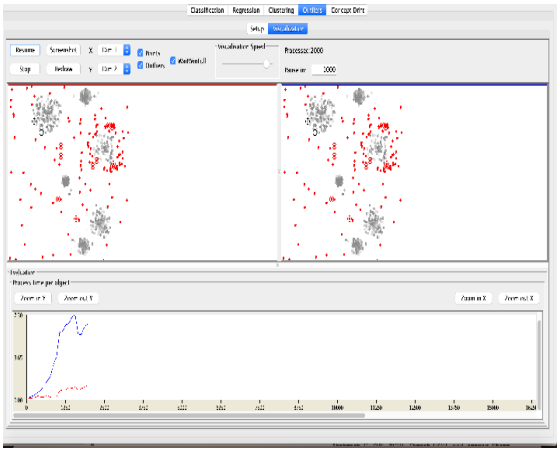

Fig: 9 Visualisation of MCOD and Abstract C

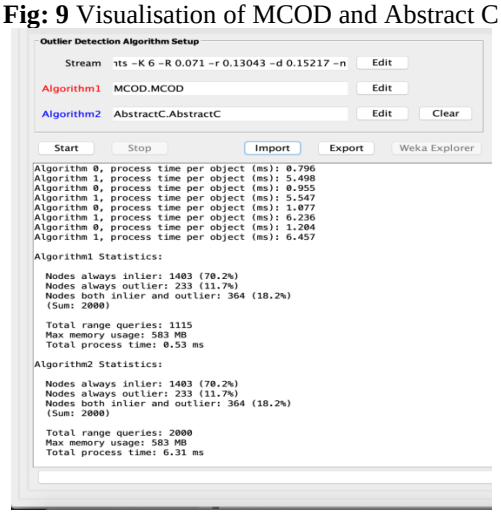

Fig: 10 - Comparison of MCOD and Abstract C

We can clearly observe that in the above two algorithms, almost all the parameters of the two algorithms are same except the processing time per object. Clearly MCOD has less processing time compared to AbstractC.

2. MCOD and Approx STORM algorithms

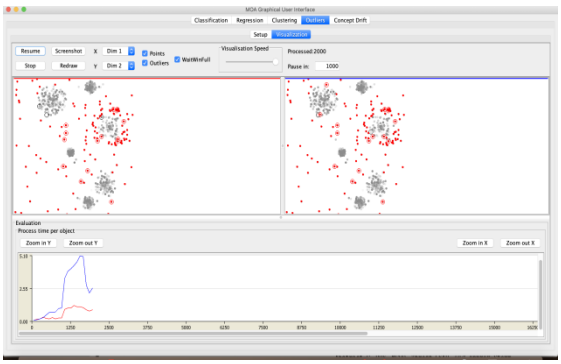

Fig: 11 Visualisation of MCOD and Approx STORM

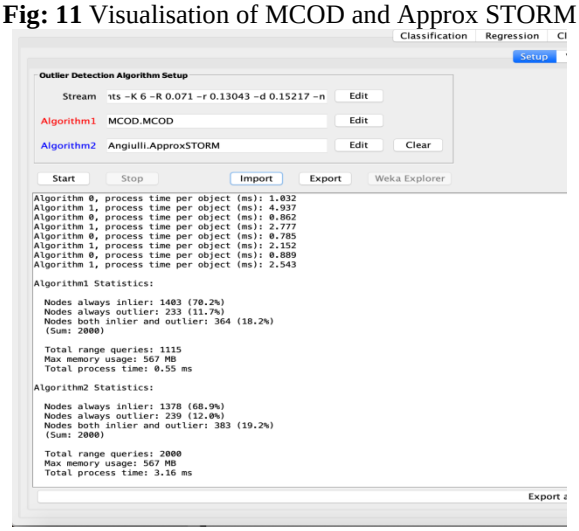

Fig : 12 Comparison of MCOD and Approx STORM

If we observe the above two figures we can clearly understand that the Memory usage of both the algorithms it is same. Also, we can see that the number of Outliers found is also approximately same for both the algorithms. But, if we observe the processing time per object MCOD has outdone Approx STORM algorithm[5].

3. MCOD and Exact STORM algorithms

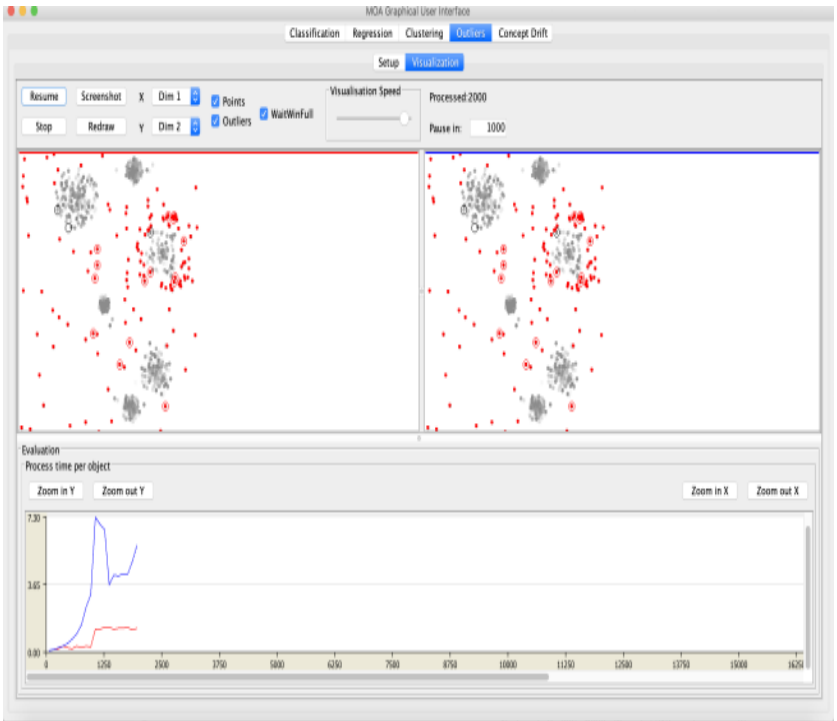

Fig: 13 – Visualisation of MCOD and Exact STORM

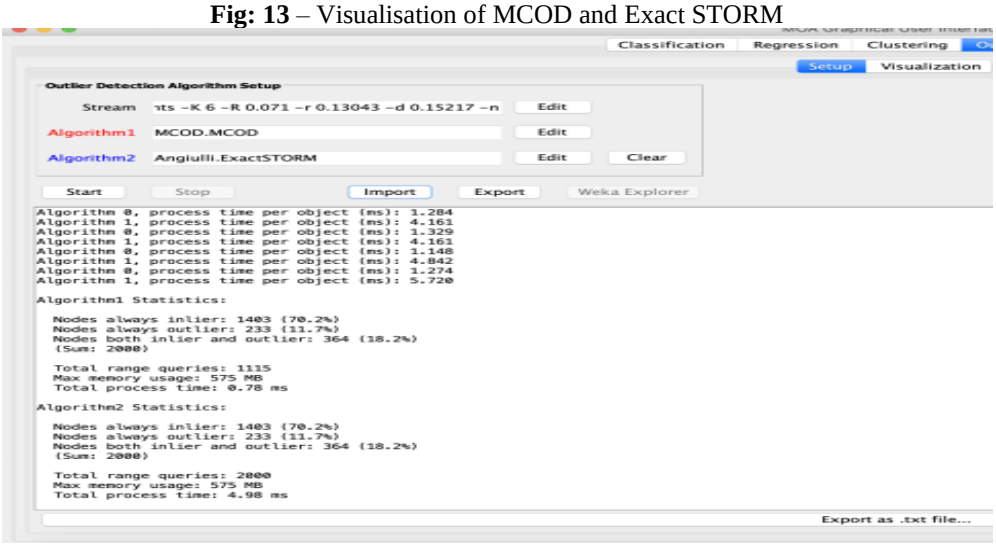

Fig: 14 Comparison of MCOD and Exact STORM

By comparison of these two algorithms we have observed that the outlier detection, memory usage etc are all similar except processing time. MCOD is the faster in processing when compared to Exact STORM[6].

4. MCOD and Simple COD algorithms

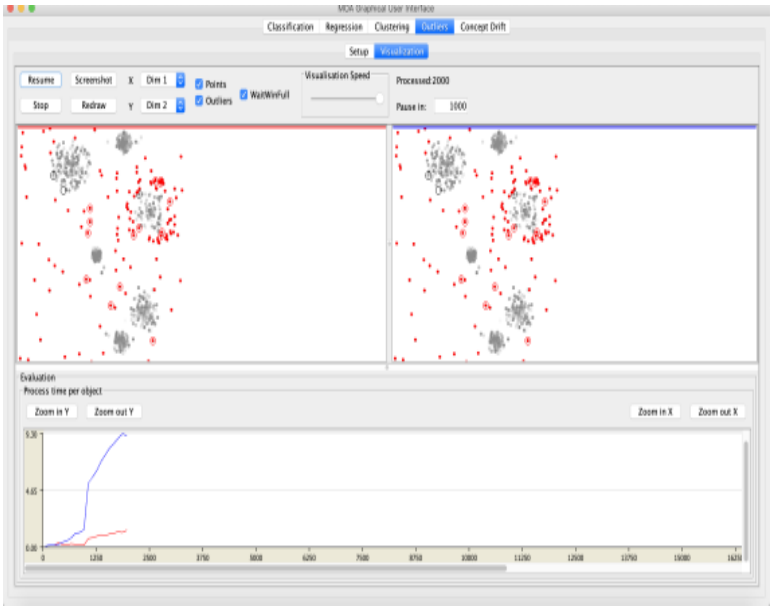

Fig: 15 Visualisation of MCOD and Simple COD

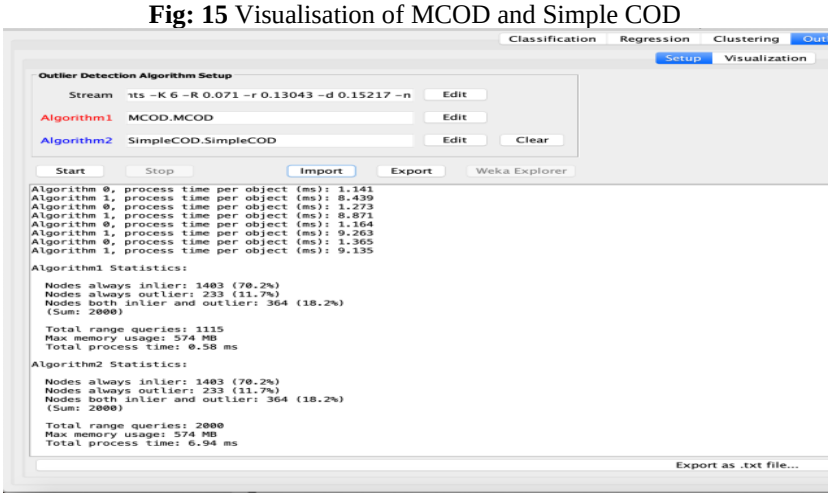

Fig: 16 Comparison of MCOD and Simple COD

If we observe the comparison of these two algorithms we see that all other aspects like memory usage and outlier nodes are similar except the processing speed. MCOD is faster than Simple COD. Thus, we conclude that MCOD is faster in its processing speed when compared to all other algorithms, even though all other parameters are similar.

5. Conclusion

During our analysis in finding the best algorithm in MOA in case of processing time, outliers, inliers, memory usage etc., we have concluded that MCOD is the best possible algorithm.

References

1. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," J. Mach. Learn. Res., 2010.
2. S. V. V. Reddy, T. Harshita, S. A. Koneru, and K. Ashesh, "Outlier detection in data streams using MCOD algorithm," 2018, doi: 10.1109/ICATCCT.2017.8389156.
3. P. T. Darshana Parikh, "Data Mining & Data Stream Mining – Open SOURCE Tools.pdf," Int. J. Innov. Res. Sci. Eng. Technol., 2013.
4. Bifet, G. Holmes, and B. Pfahringer, "MOA-TweetReader: Real-time analysis in twitter streaming data," 2011, doi: 10.1007/978-3-642-24477-3_7.
5. P. M. Arunkumar and S. Kannimuthu, "Mining big data streams using business analytics tools: A bird's eye view on MOA and SAMOA," Int. J. Bus. Intell. Data Min., 2020, doi: 10.1504/IJBIDM.2020.108761.
6. M. J. Bah, H. Wang, M. Hammad, F. Zeshan, and H. Aljuaid, "An effective minimal probing approach with micro-cluster for distance-based outlier detection in data streams," IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2946966