Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 4, No. 1, April 2019

Clustering Big Data Streams Using A Modified Hybrid Density-Based Algorithm

P. SHABANA

Associate Professor
Shabana.mtech@gmail.com

P. BHARATHKUMAR

Assistant professor
Bharathkumar1218@gmail.com

E. ANITHA

Assistant Professor
Anitha.alts@gmail.com

Abstract: When mining large data streams, density-based clustering is a vital tool to have at your disposal. This method allows for the discovery of clusters of an arbitrary form while also handling noisy data. This article introduces StreamSW, a novel density-based method for clustering large data streams using sliding windows. A sliding window paradigm is used for the purposes of recording and mining data flows. StreamSW, the suggested approach, utilises a clustering structure that is composed of two phases. Maintaining the summary statistics about the data streams requires making use of the prospective micro-clusters as well as the grid layout in the online phase. During the offline phase, the traditional method known as DBSCAN is used to produce final macro-clusters from prospective micro-clusters that were generated during the online phase. The suggested technique is capable of finding clusters of various shapes in a constrained amount of time and memory. According to the findings of the experiments, the execution speed and clustering quality of StreamSW are superior to those of the other techniques currently in use.
Keywords: Big data stream, sliding windows, density-based clustering, micro-clustering, grid-based clustering.

1. Introduction

In this day and age of big data, data is continually created, often at extraordinarily large quantities and velocities, and in a variety of formats. The term "large data stream" refers to information of this sort that is produced at a rapid rate. The monitoring of energy usage, the monitoring of urban traffic, financial transactions, the monitoring of industrial sensor data, and live updates of stock trading are some examples of applications that use real-time data streaming [1]. The majority of the time, these apps are dealing with a tremendous volume of data throughout the course of some period of time. Because of the high availability and velocity of the data, the data points can often only be read once, and it is not practical to keep all of the data in main memory for the purpose of doing an analysis at a later time. It is necessary to do methodical analysis and processing on the huge amounts of data that are generated as data streams by some real-time applications.

Clustering is a method of unsupervised learning that is widely used in a variety of real-time applications. It is able to extract relevant information and intriguing patterns from large

amounts of data and is often employed in these applications. Clustering is the process of arranging multidimensional data items into groups in such a way that the level of similarity within each cluster is increased while the level of similarity across clusters is decreased. The primary objective of clustering is to organise a given data set into meaningful clusters by combining together aspects of the data set that are similar to one another. Data points are processed in an orderly sequence in the technique known as clustering data streams, which has garnered a lot of attention in the recent years. The field of data stream clustering presents numerous new obstacles to conventional algorithms as a result of the restricted amount of time and the limited amount of memory that is available. When it comes to clustering data streams, one of the most significant challenges is figuring out how to extract useful information from real-world streaming data in a single pass. Clustering data streams calls for the use of methods that operate progressively due to the impossibility of keeping all of the data in main memory [2][3].
Recent years have seen the development of a number of stream clustering algorithms [1][10-18], all with the goal of extracting actionable intelligence from data streams in real time. A distance function is used by a few of these algorithms [1][12][13][14] in order to ascertain the degree of similarity between the individual data pieces that make up a cluster. These approaches are unable to detect clusters of arbitrary shapes, and they are unable to manage noise either.

Density-based techniques stand out among all of the different clustering approaches since they are predicated on the idea of density [4, 5].

Density-based strategies attempt to construct a density profile of the data by segmenting the data into dense zones in order to frame the profile. The use of a density threshold is what differentiates dense regions from low-density areas; these dense regions are then separated from one another. Density-based algorithms, as opposed to distance-based algorithms, have the ability to identify clusters of any form. In addition, density-based algorithms are able to deal with noise in the data. The conventional density-based clustering approach, such as DBSCAN [6,] cannot be used to data streams since these streams need numerous iterations to be performed on the data.
Grid-based approaches are another category of techniques that may be used for clustering data streams. Grid-based approaches include dividing the data space into an unbounded number of grid cells before doing the final clustering on the grids themselves. A grid-based approach of clustering maps an unlimited number of data items included in data streams to a limited number of grids. The grid cells are where the statistics that provide an overall overview of the data streams are stored.
A number of different hybrid clustering algorithms for data streams that are based on density and grid-based clustering approaches have been developed by researchers. In these, each data item is mapped to a grid, and the grids are grouped according to the density of the data contained inside them. The vast majority of the currently available density-based methods take into account the issue of data stream clustering across a damped window model. The topic of grouping data streams across a sliding window is the subject of this study, which we have written.
The following is a summary of the suggested content for the paper: Work that is relevant to this topic is discussed in section II. In this part, we will describe certain fundamental ideas and terms that are associated with the algorithm that we have suggested. The StreamSW

clustering method was suggested in section IV of this StreamSW document. In Section V, the findings of StreamSW in the real world are shown. The last part of the paper is section VI, which wraps everything up.

2. Related Work

Classical techniques of clustering such as DBSCAN [6, K-means [7, K-medians [8, and CLIQUE [9] are used in the analysis of limited static data sets. These approaches imply that the data set has an established size and that it can be saved in the primary memory of the computer. This enables several scans to be performed while going through the stored data. These approaches are laborious and need for a significant number of scans to be performed on the data set in order to identify the patterns. On the other hand, algorithms for clustering data streams need just one pass over the data items, as DenStream [10] and BIRCH [11].
Different methods have been developed throughout the course of the last few decades for the purpose of grouping data streams. The modified K-means method serves as the foundation for a number of algorithms, including STREAM [12], C1uStream [1], and HPStream [13], amongst others. These clustering methods have achieved useful results in addressing problems with the clustering of streaming data; nevertheless, the number of clusters must be determined in advance, and they have failed to detect non-spherical clusters.
Aggarwal et al. developed the CluStream [1] method in order to cluster growing amounts of data streams. CluStream functions in two stages: the first stage, known as the online phase, occurs in real time, while the second stage, known as the offline phase, occurs in postponed time. The statistical information about the data stream is stored by the micro-clustering during the online phase, and the macro-clustering employs this statistical information during the offline phase to locate the final clusters. When a new data item is received, the micro-clusters are updated so that they may undo any modifications that have been made. In the offline phase, CluStream uses the K-means method to get the desired results of the final macro-clusters. The CluStream technique is not capable of capturing an accurate representation of the latest record's distribution.

Zhou et al. came up with the SWClustering [14] method, which prefers sliding windows when it comes to monitoring the growth of clusters. SWClustering is responsible for the development of a one-of-a-kind data structure known as the Exponential Histogram of Cluster Features (EHCF), which accurately depicts the distribution of recent records. EHCFs are used in order to preserve the characteristics of the cluster as well as to record the dispersion of more recent items throughout a sliding window. However, as it is based on standard K-means [7], the SWClustering method is unable to find clusters of various shapes and cannot process noisy data. [Citation needed]
DenStream [10] is a method that was developed by Cao and colleagues for grouping developing data streams across a damped window. DenStream is a technique to density-based micro-clustering that was developed in order to reveal clusters of various shapes. The DenStream clustering methodology utilises a two-phase approach, with an online component and an offline component. While the clustering algorithm is in its online phase, it outlines the data objects by applying the concept of core-micro-clusters. At the same time, it introduces p-micro-clusters (potential-micro-clusters) and o-micro-clusters (outlier-micro-clusters) to maintain and differentiate the potential and outlier clusters. A pruning process is included in DenStream so that the true outliers may be found. DenStream employs a variation of the

DBSCAN method whenever a clustering request is made during the offline phase. This algorithm uses the p-micro-clusters as virtual points.

The D-Stream [15] technique was suggested by Chen et al. This approach makes use of the grid structure rather than the micro-clusters in order to collect the summary information of data items. For the purpose of clustering real-time streaming data across a damped window, D-Stream employs a hybrid online-offline clustering methodology. During the online part of the process, the algorithm maps the data items onto the limited amount of gridcells that stand in for the grid's density. When a new data element is added to a grid cell during the online phase, a grid characteristic vector is modified as a result. This takes place automatically. During the offline phase of the process, the algorithm determines the grid density for each grid and then clusters the grids according to the grid density.

The SDStream [16] technique was developed by Ren et al. for the purpose of clustering data streams across a sliding window. The density-based approach known as SDStream is an evolution of the algorithm known as SWClustering [14]. The CluStream [1] algorithm's two-phase clustering structure is used by SDStream in its implementation. SDStream uses a sliding window model to process the most current data items and summarise the older data objects. This is done after processing the most recent data objects. The method takes into account the dispersion of the most recent data, and it gets rid of the data items that cannot be accommodated in a sliding window of a certain length.

The DENGRIS-Stream [17] technique was developed by Amineh and colleagues for the purpose of grouping data streams across a sliding window. DENGRIS-Stream begins by placing each new data item on a grid, then determines the density of each grid, and then clusters the grids according to the density of each grid. It is the first grid-based technique for clustering data streams across a sliding window, and its name is DENGRIS-Stream.

They presented the MCDAStream approach for clustering real-time streaming data in [18], which is based on the common density of the micro-clusters and occurs across a damped frame. This algorithm is used to cluster the data. A two-phase clustering structure is used by MCDAStream. The live phase is responsible for the maintenance of summary information of the data stream in the form of micro-clusters, while the offline phase is responsible for the generation of final macro-clusters. During the online phase, the MCDAStream makes use of an attraction-based approach to record the shared denseness between the micro-clusters. Then, during the offline phase, it makes use of the same information to provide improved clustering results.

In many real-time applications, the users are more interested in discovering patterns in the most current data items as opposed to the whole data collection. This is because the most recent data contains the most relevant information. Sliding window models are often used in this context because [19][20][21] they are able to represent the N most recent data items that are present in the stream. Utilizing a sliding window with a predetermined size over a data stream is the primary concept behind the sliding window paradigm. Because the vast majority of the processed data would no longer be reliable, the algorithms that deal with the whole data set with the same level of relevance would return results that are aberrant. In this article, we introduce an unique density-based technique that we have dubbed StreamSW for the purpose of grouping data streams across sliding windows. StreamSW was developed by us.

3. Basic Concepts and Definitions

In this section, we provide some definitions of terms related to our proposed algorithm. We introduce the format of p-micro-clusters that are similar to those in DenStream [10] and grids that are similar to those in DStream-II [22].

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 4, No. 1, April 2019

Definition 1.Data Stream: An ordered multidimensional data objects p_1, p_2 ... p_i arriving at timestamps t_1, t_2 ... t_i, that can be read only once using limited memory and processing capabilities.

Definition 2. Core object: An object whose ε-neighbourhoodhas at least μ (MinPts) number of points.

Definition 3. Core-micro-cluster: A *c-micro-cluster by* time *t* is outlined just as (*w, c, r*) for a set of data items $p_{i_1}, p_{i_2} \dots p_{i_n}$with timestamps $T_{i_1}, T_{i_2} \dots T_{i_n}$ as follows:

1. *w is the weight,* w ≥μ,and it is equivalent to the total number of points in the *c-micro-cluster*,
2. *Center, c*=$\frac{\sum_{j=1}^{n} p_{i_j}}{w}$
3. *Radius, r*=$\frac{\sum_{j=1}^{n} dist(p_{i_j},c)}{w}$, with r≤ε, where $dist(p_{i_j}, c)$represents the Euclidean distance among the pointp_{i_j} and center *c.*

Definition 4. Potential micro-cluster: A *p-micro-cluster by* time *t* is outlined just as (*w,* $\overline{CF^1}$,$\overline{CF^2}$, *c, r, t*) for a set of close data items $p_{i_1}, p_{i_2} \dots p_{i_n}$with timestamps $T_{i_1}, T_{i_2} \dots T_{i_n}$ as follows:

1. *w is the weight,* w ≥βμ, and it is equivalent to the total number of points in the *p-micro-cluster*, where β, 0<β<1, is the governing threshold.
2. *Linear sum of the data objects,*$\overline{CF^1}$= $\sum_{j=1}^{n} p_{i_j}$
3. *Squared sum of the data objects,*$\overline{CF^2}$ = $\sum_{j=1}^{n} p^2{}_{i_j}$
4. *Center,c*=$\frac{\overline{CF^1}}{w}$
5. *Radius, r*=$\sqrt{\frac{|\overline{CF^2}|}{w} - (\frac{|\overline{CF^1}|}{w})^2}$ with r≤ε,
6. Timestamp *t*, is the arrival time of the most recent data object.

Definition 5. Grid density(w_g): Let N be the length of the sliding window and M be the total number of data objects mapped to the grid g until time t. Grid density is calculated based on the number of data objects that are mapped to grid *g* until time t.

$w_g(t)= |M|;$ $t \in (t_c - N+1, t_c)$

Definition 6. Maximum density($w_{g\text{-}max}$):Maximum density is the sum of all density of each grid over each sliding window.

$w_{g\text{-}max}(t)=\sum_{i=1}^{n} |M(g_i, t_g)|, t_g \in (t_c - N+1, t_c)$

t_g: the timestamp of the most recent record of the grid cell.

Definition 7. Average density($w_{g\text{-}avg}$): The average density of each grid is defined as follows:

$w_{g\text{-}avg} = \frac{w_{g-max}}{n}$, where n is the number of cells in the grids.

Definition 8. Active grid(g_{active}):At current timestamp t_c, a grid g with timestamp t_g, is said to be active if: $t_g \in (t_c - N+1, t_c)$.

Definition 9. Expired grid($g_{expired}$): At current timestamp t_c, a grid g with timestamp t_g, is said to be expired if: $t_g \notin (t_c - N+1, t_c)$. Expired grids are the grids which are no more active in the N-length sliding window.

Definition 10. Densegrid: For an active grid *g*, at current time stamp t_c, if $w_g(t_c) \geq \frac{\beta * w_{g-max}}{n}$, whereβ is a controlling threshold, then we call it as a dense grid.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 4, No. 1, April 2019

Definition 11. Sparse grid: For an active grid *g*, at times tampt_c, we call it a sparse grid if $w_g(t_c) < \frac{\beta * w_{g-max}}{n}$.

Definition 12. Grid Characteristic Vector (CV): Grid characteristic vector is used to hold summary statistics about the data objects in each grid. The characteristic vector is a tuple *CV* (w_g, t_g, l) where w_g is the grid density, t_g is the timestamp of the grid, and l is the label, where label = {active, expired, dense, sparse}.

4. StreamSW Algorithm

StreamSW algorithm adopts the two-phase online-offline framework for clustering data streams over a sliding window. The online phase of StreamSW continuously reads an object from a data stream over a fixed size sliding window, and adds it to an existing p-micro-cluster or maps it to the grid cell. In the offline phase, the algorithm uses modified DBSCAN[6] algorithm on requirement by user to generate the final clusters..

Algorithm 1: StreamSW
Input: N, data stream (*DS*), μ, ε, and β
Output: Clusters of arbitrary shape in an N-length sliding window.
Method:

1) $t_c \leftarrow 1$;
2) *grid_list* ← empty;
3) **while***DS* is active **do**
4) get an object*p* from *DS* at timestamp t_c;
/* Merging and Mapping phase*/
5) Get c_p, the nearest *p*-micro-cluster from an object*p*
6) Merge *p* tentatively into c_p;
7) let r_p be the new radius of c_p;
8) **if** ($r_p \leq \varepsilon$) **then**
9) Merge*p* into c_p;
10) **else**
11) Mapthe data object*p* into the grid;
12) Update $CV(w_g, t_g, l)$;
13) $w_g \leftarrow w_g+1$;
14) $t_g \leftarrow t_c$;
15) //creating new *pμC*
16) **if** $w_g(t_c) \geq \frac{\beta * w_{g-max}}{n}$ **then**
17) create new *p-micro-cluster*(*w*, $\overline{CF^1}$,$\overline{CF^2}$, *c*, *r*, *t*);
18) w ← w_g;
19) $\overline{CF^1} = \sum_{j=1}^{n} p_{i_j}$;
20) $\overline{CF^2} = \sum_{j=1}^{n} p^2_{i_j}$;
21) $c = \frac{\overline{CF^1}}{w}$;
22) $r = \sqrt{\frac{|\overline{CF^2}|}{w} - (\frac{|\overline{CF^1}|}{w})^2}$;
23) t ← t_c;

```
24)   delete grid g from the grid_list;
25)   end if
26)   if (tc mod N ==0) then
27)   pruning( );
28)   end if
29)   end if
30)   tc← tc + 1;
31)   end while
/*Offline-phase*/
32)   if clustering request arrived then
33)   Apply DBSCAN algorithm on the set of p-micro-clusters that are
      maintained in the online phase to generate the final macro-clusters;
34)  end if
```

In DBSCAN[6],a set of initial *p-micro-clusters* are generated,before the execution of the StreamSW algorithm. The online component of StreamSW has two parts: Merging and Mapping phase, and the Pruning phase.

Merging and Mapping phase: Algorithm 1 shows the merging and mapping phase of StreamSW. Where as new data object p in the data flow enter at timestamp t_c, the procedure for merging and mapping the data point is described as follows.

1. At first, StreamSW finds the closestp-micro-clusterc_p from *p*
2. Then we try to merge p into c_p.
3. If r_p, the new radius of c_p, is less than or equal to maximum radius ε, then *p* is merged into c_p.
4. Else, *p* is mapped into the grid *g*, and then we update the grid characteristic vector.
5. If g is a dense grid, create a new *p-micro-cluster*, out of data objects in the grid*g*.
6. From the grid list, associated grid *g* of the new *p-micro-cluster* is deleted.

Offline phase: A variant of DBSCAN algorithm is adopted to create final macro-clusters in the offline phase. The online phase of StreamSW maintains summary information of data streams in the form of *p-micro-clusters*. However, suppose to create final arbitrary shape clusters, a modified DBSCAN algorithm is executed on the online maintained p-micro-clusters.Each p-micro-cluster is considered as a virtual point placed at center *c* with weight *w*.

```
Algorithm 2: Pruning ( )
1)    RemoveExpiredGrids( )
2)    for each grid in grid_list do
3)    if g is a sparse grid then
4)    remove the grid g from the grid_list;
5)    endif
6)    end for
7)    //detecting and removing outlier micro-clusters
8)    if w < β*μ then
9)    remove  p-micro-cluster against p-micro-cluster_list;
10)   end if
11)   if (all data objects of cp are outdated)
12)   Delete cp;
13)   end if
```

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 4, No. 1, April 2019

Pruning Phase: Algorithm 2 shows the pruning phase of StreamSW. In this phase, expired grids, sparse grids, and outlier micro-clusters are removed. We detect and remove expired grids based on the timestamp of the grids. If the timestamp t_g of the grid *g* is not in the N-length sliding window, it is treated as an expired grid, and it will be removed from the grid list (see Algorithm 3). Sparse grids and outlier micro-clusters are deleted from gird list and p-micro-cluster list, respectively, by checking their weights periodically.

Algorithm 3:*RemoveExpiredGrids*( )
1) **for each** grid *g* in *grid_list* **do**
2) **if** the timestamp t_g of the grid g is not in the sliding window size **then**
3) label(g) ← expired;
4) remove the grid g against the *grid_list*;
5) **end if**
6) **done**

5. Experimental Results

This section presents the experimental results of the StreamSW algorithm. We have implemented our StreamSW in an R-extension, named stream [23], which is an extended framework for implementing and experimenting with various data mining algorithms. We compare StreamSW with existing methods MCDAStream[18], SDStream[16] and CluStream[1].Similar to SDStream [16], the parameters for StreamSW algorithm were set as follows: outlier threshold β=0.2, μ=10, and ε=16.For MCDAStream we choose λ=0.5, and for CluStream we choose k=5 clusters.

5.1 Data sets:

To evaluate the performance ofStreamSW and other existing algorithms we use both synthetic and real data sets.

Synthetic data set: We have experimented with a synthetic data set called NMG (Noisy Mixture of Gaussians) generated from stream [23].Example points for NMG data set are shown in Figure 1. It contains 2-d data objects with 5% noisy data.

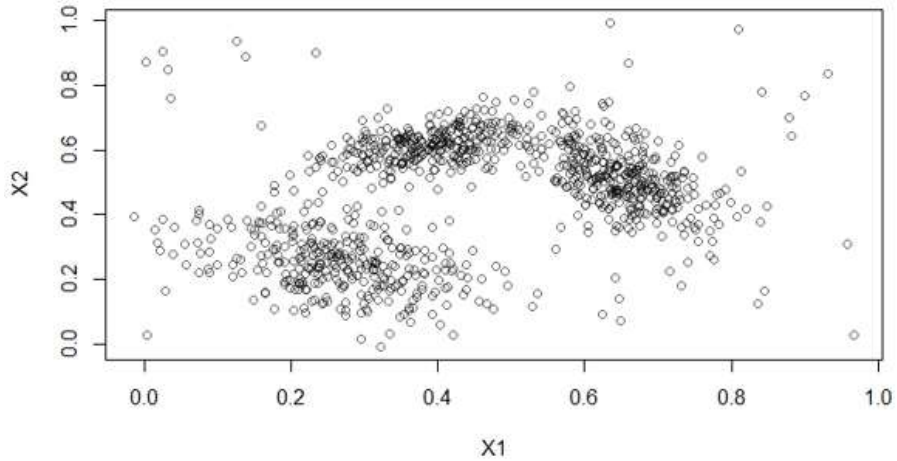


Figure 1: NMG data set

*Real data set:*To assess the performance of the StreamSW algorithm with real-world data, we choose a data set called Network Intrusion Detection (NID) data set.NID is an extensively used data set which is obtainable from UCI machine learning repository [24]. It consists of simulated LAN network traffic with a collection of intrusions. Each connection record has 42

attributes in which 34 are continuous and remaining are categorical. Total it has 48,98,431 data points. We consideredall 34 continuous attributes for evaluation of our algorithm.

5.2 Clustering Quality Evaluation

In order to determine the quality of StreamSW, we have used a simple evaluation criterion, termedpurity [10][15]. At first,determine the StreamSW quality on NMG data set. Figure 2 display the clustering pureness of results of StreamSW compared to MCDAStream, SDStream, and CluStream. We can see that the clustering purity of StreamSW is greater than 98% in all time intervals and it is better than the traditional clustering algorithms. In figure 2, N (the sliding window size) is set to 4.

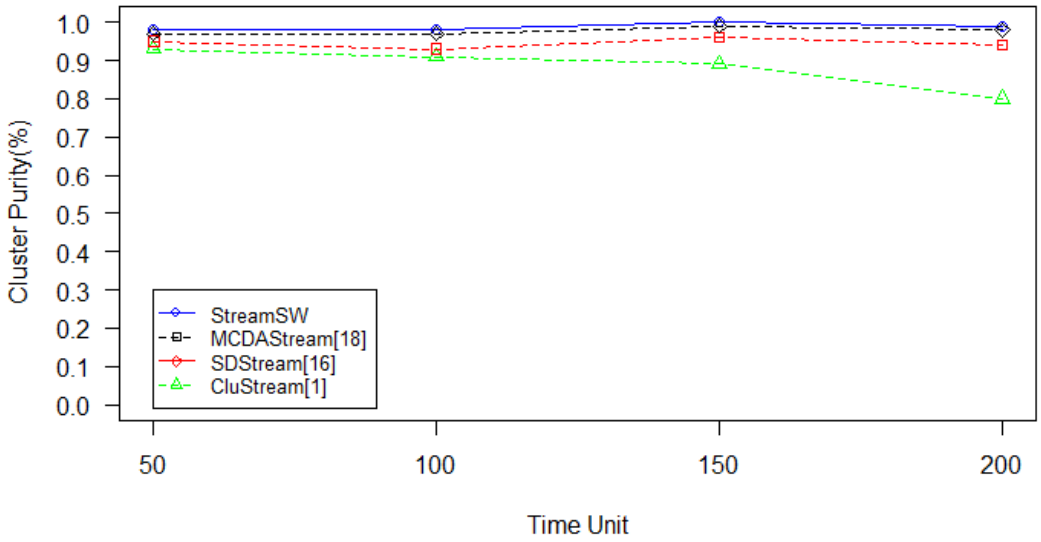
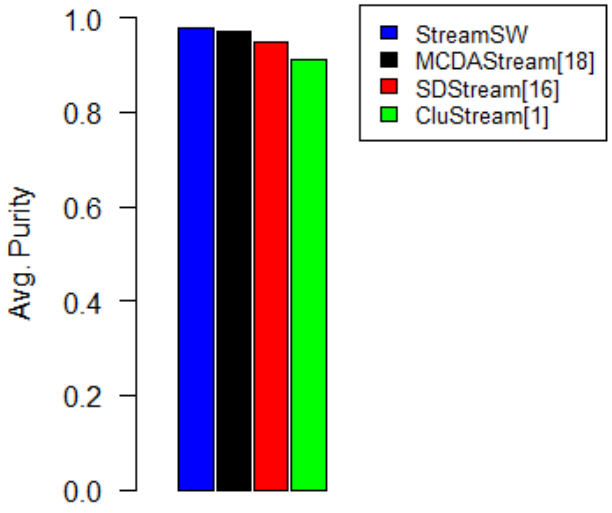


Figure 2: Quality comparison of the Noisy Mixture of Gaussians data set, N=4.

We have also assessed clustering quality of StreamSW on NID data set. Figure 3 shows the clustering purity results for the NID data set. It can be observed that StreamSW has a very good clustering quality than MCDAStream, SDStream, and CluStream. The average cluster purity of StreamSW is greater than 98% which is higher than the existing algorithms. This is because CluStream uses a distance function to measure the similarity between two objects. It may merge different attacks into a single attack that leads to poor clustering results. In MCDAStream outdated data points will affect the clustering results. StreamSW can find arbitrary shape clusters and separate different attacks into different clusters.



Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 4, No. 1, April 2019

Figure 3: Average clustering purity of StreamSW on NID real-world dataset.

5.3 Execution Time

The run time performance of the StreamSW is evaluated by the execution time. We have used NID data set to measure the efficiency of StreamSW against MCDAStream, SDStream, and CluStream. The execution time of StreamSW is affected by the stream speed. Figure 4 display the execution time in seconds for NID dataset.In this ,observation is that execution time of StreamSW and existing clustering methods grow linearly as the stream gains over sliding windows. Also observe that StreamSW, when compared to SDStream and CluStream,it has lower execution time. The online phase of StreamSW uses the grid structure to map the outlier data points. So the execution time of StreamSW is lower than SDStream and CluStream. The time complexity of StreamSW is reduced by adopting grid-based clustering.

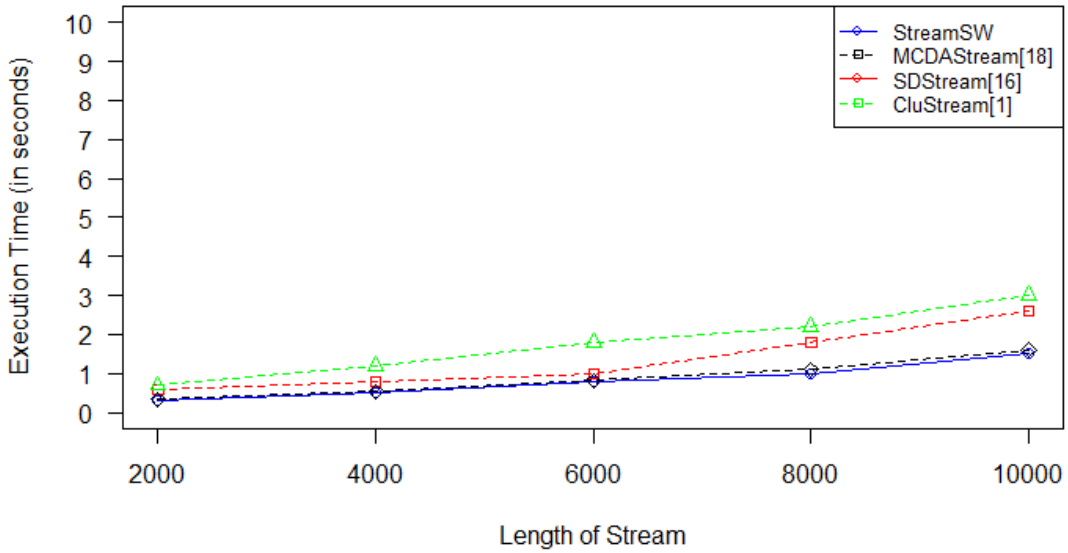


Figure 4: Execution time vs. Length of the stream.

6. Conclusion

An effective density-based clustering technique for data streams was reported in this article under the name StreamSW. This approach uses a sliding window. The purpose of the proposed technique is to extract data clusters of various shapes from huge data sets that have been structured in the form of data streams. The StreamSW method uses a framework that combines online and offline clustering, and it does so in two phases. The data items are read from the data stream by the online component, which also keeps the summary statistics on the data stream in the form of p-micro-clusters. The offline component uses a modified version of the DBSCAN algorithm in order to recluster the p-micro-clusters. Our technique takes use of the density-based micro-clustering strategy as well as the density grid-based clustering approach in order to locate high-quality arbitrary shape clusters in a much shorter amount of execution time and in a minimal amount of memory. Experiments were run on both synthetic and actual data sets in order to assess the performance of StreamSW in a variety of contexts and context-specific contexts. According to the findings of the study, the StreamSW creates clusters of a high quality while also having a shorter execution time than other algorithms now in use.

7. References

[1] Yu, K.; Shi, W.; Santoro, N. Designing a streaming algorithm for outlier detection in data mining—An incrementa approach. Sensors 2020, 20, 1261. [CrossRef] [PubMed]
[2] 2. Degirmenci, A.; Karal, O. Efficient Density and Cluster Based Incremental Outlier Detection in Data Streams. Inf. Sci. 2022, 607, 901–920. [CrossRef]
[3] Al-Amri, R.; Murugesan, R.K.; Alshari, E.M.; Alhadawi, H.S. Toward a Full Exploitation of IoT in Smart Cities: A Review of IoT Anomaly Detection Techniques. In International Conference on Emerging Technologies and Intelligent Systems; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2022; Volume 322, pp. 193–214. [CrossRef]
[4] Märzinger, T.; Kotík, J.; Pfeifer, C. Application of hierarchical agglomerative clustering (Hac) for systemic classification of pop-up housing (puh) environments. Appl. Sci. 2021, 11, 11122. [CrossRef]
[5] Zubaro ˘glu, A.; Atalay, V. Data Stream Clustering: A Review; Springer: Amsterdam, The Netherlands, 2020.
[6] Al-amri, R.; Murugesan, R.K.; Man, M.; Abdulateef, A.F. A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data. Appl. Sci. 2021, 11, 5320. [CrossRef]
[7] Habeeb, R.A.A.; Nasaruddin, F.; Gani, A.; Hashem, I.A.T.; Ahmed, E.; Imran, M. Real-time big data processing for anomaly detection: A Survey. Int. J. Inf. Manag. 2019, 45, 289–307. [CrossRef]
[8] Carnein, M.; Trautmann, H. Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms. Bus. Inf. Syst. Eng. 2019, 61, 277–297. [CrossRef]
[9] Maia, J.; Junior, C.A.S.; Guimarães, F.G.; de Castro, C.L.; Lemos, A.P.; Galindo, J.C.F.; Cohen, M.W. Evolving clustering algorithm based on mixture of typicalities for stream data mining. Future Gener. Comput. Syst. 2020, 106, 672–684. [CrossRef]
[10] Manzoor, E.; Lamba, H.; Akoglu, L. xStream: Outlier Detection in Feature-Evolving Data Streams. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA, 19–23 August 2018. [CrossRef]
[11] Anandharaj, A.; Sivakumar, P.B. Anomaly Detection in Time Series data using Hierarchical Temporal Memory Model. In Proceedings of the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 12–14 June 2019; pp. 1287–1292. [CrossRef]
[12] Gottwalt, F.; Chang, E.; Dillon, T. CorrCorr: A feature selection method for multivariate correlation network anomaly detection techniques. Comput. Secur. 2019, 83, 234–245. [CrossRef]
[13] Munir, M.; Siddiqui, S.A.; Dengel, A.; Ahmed, S. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. IEEE Access 2019, 7, 1991–2005. [CrossRef]
[14] Hyde, R.; Angelov, P.; MacKenzie, A.R. Fully online clustering of evolving data streams into arbitrarily shaped clusters. Inf. Sci. 2017, 382–383, 96–114. [CrossRef]
[15] Islam, M.K.; Ahmed, M.M.; Zamli, K.Z. A buffer-based online clustering for evolving data stream. Inf. Sci. 2019, 489, 113–135. [CrossRef]