International Journal of Engineering &Technology, 7 (2) (2018) 270-275

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i2.9051
Research paper

MCDAStream: a real-time data stream clustering based on micro-cluster density and attraction

K. Shyam Sunder Reddy [1]*, C. ShobaBindu [2]

[1]Research Scholar, JNTUA, Ananthapuramu
[2]Department of CSE, JNTU College of Engineering, Ananthapuramu
**Corresponding author E-mail: shyamd4@staff.vce.ac.in*

Abstract

Real-time data stream clustering has been widely used in many fields, and it can extract useful information from massive sets of data. Most of the existing density-based algorithms cluster the data streams based on the density within the micro-clusters. These algorithms completely omit the data density in the area between the micro-clusters and recluster the micro-clusters based on erroneous assumptions about the distribution of the data within and between the micro-clusters that lead to poor clustering results. This paper describes a novel density-based clustering algorithm for evolving data streams called MCDAStream, which clusters the data stream based on micro-cluster density and attraction between the micro-clusters. The attraction of micro-clusters characterizes the positional information of the data points in each micro-cluster. We generate better clustering results by considering both micro-cluster density and attraction of micro-clusters. The quality of the proposed algorithm is evaluated on various synthetic and real-time datasets with distinct characteristics and quality metrics.

Keywords: *Data Stream; Data Mining; Density-Based Clustering; Grid-Based Clustering; Micro-Clusters.*

1. Introduction

The amount of data being generated as data streams from real-life applications has significantly increased over the past few years. A data stream (DS) is a potentially unbounded sequence of ordered data points p_1, p_2 ... p_n arriving at time steps t_1, t_2 ... t_n. Data streams are evolving, and are too complicated and voluminous to be processed and analysed with conventional processing methods. Due to the massive amounts of streaming data, it is not possible to store the whole data stream in main memory. One of the key challenging issues in clustering data streams is single pass clustering. i.e., the raw data can only be examined in one scan. Real-time data stream clustering is a significant and challenging problem in data analysis with ample applications such as banking, agriculture, healthcare, weather monitoring, education, stock trading, network monitoring, telecommunication, and website analysis [1].

Data stream clustering can extract valuable information from a huge volume of data. There are many families of data clustering algorithms: Hierarchical, Grid-based, Partitioning, Model-based, and Density-based clustering [2]. Among all the clustering algorithms, Density-based clustering algorithms are a widely-used class of data mining techniques that can find irregularly shaped clusters, can handle noises, and cluster data without prior knowledge of the number of clusters it contains as the k-means algorithm does.

Most density-based algorithms for clustering data streams use a two-stage online-offline framework [3]. The online component reads the data stream in real-time and keeps the summary statistics about the data streams in the form of micro-clusters (grids). A micro-cluster is a tuple (w, c, t), where w is the weight, c is the center, and t is the last updated time. The offline component re-clusters the micro-clusters (grids) into final clusters (sometimes called as macro-clusters). On-demand, since the reclustering process is not time critical, a modified conventional clustering algorithm (e.g., a variant of DBSCAN [4]) is used in the offline phase for reclustering. The offline component performs reclustering based on the density within the micro-clusters, and it considers centers of micro-clusters as pseudo points.

The rest of this paper is organized as follows: We briefly review the related work in section 2. In section 3 we discussed our proposed algorithm MCDAStream, and then we presented the experimental results in Section 4. Then we conclude the paper with Section 5.

2. Related work

Over the past few years, several density-based algorithms have been proposed for clustering potentially unbounded streams of data. These algorithms are categorized into two broad groups. (1) Density-based micro-clustering algorithms group, and (2) Density grid-based clustering algorithms group.

Some of the existing density-based micro-clustering algorithms are: DenStream [5], OpticsStream [6], C-DenStream [7], rDenStream [8], SDStream [9], HDenStream [10], SOStream[11], HDDStream [12], PreDeConStream [13], FlockStream [14], LeaDen-Stream [15], DBStream[16]. All these algorithms use two stage online-offline framework. In the online phase, these algorithms adopt the concept of micro-clusters to compress the data streams efficiently. The micro-cluster (μC) is an extension of Cluster Feature Vector (CFV), and it was first proposed by Zhang et al. in[17]. Micro-clusters keeps the summary statistics about the data stream, and reclustering is performed on these μCs. In the reclustering step (offline phase) these algorithms merge/group adjacent micro-clusters into macro-clusters. For reclustering, only

the distances between the micro-clusters and their weights are considered. In this regard, micro-clusters which are closer to each other are grouped to the same macro-cluster. Most existing clustering algorithms adopt fading window model for clustering data streams.

There are few other algorithms like DUCStream [18], DStream-I [19], DDStream [20], DStream-II [1], MR-Stream [21], PKS-Stream [22], DCUStream [23], DENGRIS-Stream [24], ExCC [25], which are belongs to density grid-based clustering algorithms group. These algorithms adopt density-based and grid-based methods for clustering the data streams. In these algorithms group, the data space is divided into grids (small segments), data points are mapped to these grids, and the clusters are created based on the density of the grids. For each data point in the grid, density coefficient is considered to capture the dynamic changes of the clusters. The density of the grids can be determined based on the aggregation of the weights of all the data points in the grid. In some grid-based algorithms, the density of the grid is determined based on the number of data points in the grid. Algorithms such as DStream-I [19] and DStream-II [1] adopt Characteristic Vector (CV) to maintain summary statistics about the data points in the grid. For some grid-based clustering algorithms, the concept of sporadic grids was introduced to handle outlier data.

Most of the existing density-based clustering algorithms completely omit the data density in the area between the adjacent micro-clusters (grids). Reclustering methods only consider the closeness of the micro-clusters into account and recluster the micro-clusters based on erroneous assumptions about the distribution of the data within and between micro-clusters. The concept of density estimation in grid cells was used in the existing algorithms like DStream-I [19] and MR-Stream [26]. This makes it possible that two micro-clusters which are separated by low-density region will be merged into a macro-cluster as long as their distance is low.

To address this problem, CHAMELEON [27] introduced inter-connectivity concept that works only for clustering static datasets. This interconnectivity concept was extended in DStream-II [1] for clustering data streams based on grid density and attraction. The information about the grid attraction is collected in the online component, and it is used for reclustering in the offline component. The adjacent grid cells are grouped if the density attraction between the cells is high enough. This method is not directly applicable for micro-clusters. To address the same problem, LeaDen-Stream [15] introduced a concept in which micro-clusters are represented by mini-micro leaders based on the distribution of data points in the micro-clusters, and it uses this representation for reclustering. DBStream [16] is the first density-based clustering method that clusters the micro-clusters based on shared density between the micro-clusters. DBStream captures the shared density information in the online component via a shared density graph, and it uses the same information in the offline phase for reclustering the micro-clusters. DBStream uses adjacency list for implementing sparse shared density graph. This approach generates large quantities of outliers which increased cluster quality, but it requires high computation time for clustering process. To reduce the time complexity and to improve the efficiency of current density-based clustering methods, we propose a novel density-based algorithm for clustering data streams, called MCDAStream.

3. Proposed work

The MCDAStream algorithm clusters the data stream based on micro-cluster density and attraction. The attraction of micro-clusters characterizes the positional information of the data points in each micro-cluster. Like many existing algorithms, MCDAStream has two phases: online phase for maintaining micro-clusters and offline phase for generating final clusters. Our algorithm adopts an attraction-based mechanism to capture the density information between the micro-clusters in the online phase, and it uses the same information in the offline phase to generate better clustering results accurately.

3.1. Basic concepts and definitions

To record the dynamic changes of a data stream, MCDAStream adopts a density decaying function (aging function) to the density of each data point. The density decaying function is an exponential function which is defined as $f(t)= 2^{-\lambda t}$ where f is a decaying function and $\lambda > 0$ is the decay parameter. The clusters are formed automatically and dynamically by placing more weights on the most recent data without totally discarding the historical information. According to this decaying function, the weight of the micro-clusters (grids) decreases exponentially with time t.

Definition 1:*(Weight of a data point (w_p)): The weight of a data object p at current time t_c is defined as follows:*

$$w_p(t_c) = w_p(t_p) * f(t_c\text{-}t_p) = 2^{-\lambda(t_c - t_p)}$$

Where $w_p(t_p)$ is the weight of the data point p at time t_p and $t_c > t_p$. The initial weight of a data point is assumed to be 1.

Definition 2:*(Grid density (w_g)): Density of a grid g at current time t_c is defined as follows:*

$$w_g(t_c) = \sum_{p \in g} 2^{-\lambda(t_c - t_p)}$$

The weight of a grid is updated at current time t_c with the last updated time t_g as follows:

$$w_g(t_c) = w_g(t_g) * f(t_c\text{-}t_g) + 1$$

The total weight of the grid has an upper bound of $\frac{1}{(1-2^{-\lambda})}$, and the average density is $\frac{1}{N_g(1-2^{-\lambda})}$ where N_g is the number of grids. A grid g is said to be a dense grid if its weight at time t is greater than $\frac{\beta}{N_g(1-2^{-\lambda})}$, where $\beta > 0$ is a controlling threshold. If the weight of a gird at time t is less than $\frac{\beta}{N_g(1-2^{-\lambda})}$, then it is called as a sparse grid.

Definition 3:*(micro-cluster (μC)): A micro-cluster (μC) at time t is determined as μC(w, c, t) for a group of data objects $p_{i_1}, p_{i_2} \ldots p_{i_n}$ with timestamps $T_{i_1}, T_{i_2} \ldots T_{i_n}$ as follows:*

1) $w = \sum_{j=1}^{n} f(t - T_{i_j})$ is the weight, $w \geq \beta\mu$, where β is the parameter to determine the threshold of an outlier relative to μCs and μ represents the minimum number of points.
2) $c = \frac{\overrightarrow{CF^1}}{w}$ is the center of μC, and $\overrightarrow{CF^1} = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j}$ is the weighted linear sum of the data points.t is the last updated time of μC
3) If all the data points are merged into the same μC, then the maximum weight of the micro-cluster is defined as follows:

$$\mu C[w_{max}] = \lim_{t_c \to \infty} \frac{1-2^{-\lambda(t_c+1)}}{(1-2^{-\lambda})} = \frac{1}{(1-2^{-\lambda})}.$$

Definition 4 (Characteristic Vector): The characteristic vector (CV) of a grid is a tuple (n_g, t_g, w_g) where n_g is the number of data points in the grid, t_g is the last updated timestamp and w_g is the weight of the grid.

Definition 5 (Density threshold function (Δ)): Density threshold function [22] is designed for the sporadic grids. The sporadic grids are the grids which do not receive any data objects for a long time. If the grid weight is less than the density threshold function, then we can safely prune the grid from the grid list. The density threshold function Δ is defined as follows:

$$\Delta(t_c, t_g) = \frac{\beta}{N_g} \sum_{i=0}^{t_c - t_g} 2^{-\lambda i} = \frac{\beta\left(1-2^{-\lambda(t_c - t_g + 1)}\right)}{N_g(1-2^{-\lambda})}$$

Where t_c is the current time, and t_g is the last updated time of a grid.

Definition 6 (Pruning time): For each micro-cluster μc_i, if no new data point is added to it, the weight of μc_i will decay gradually. If $\mu c_i[w] < \beta\mu$, then μc_i becomes an outlier micro-cluster, and it should be deleted from the memory. We check the weights of all the micro-clusters as well as the weights of all the grids at a time we call t_r. Pruning time t_r is the minimum time for a μC/gird in time step t_1 to be converted to an outlier in t_2 ($t_2 > t_1$) which is determined as follows[5]:

$$t_r = \frac{1}{\lambda}\log_2\left(\frac{\beta\mu}{\beta\mu-1}\right)$$

3.2. The MCDAStream algorithm

At each time stept_c, the online component of MCDAStream reads a data point p from the data stream, and either merge it into an existing micro-clusters or maps it to the grid. To improve the runtime performance of the proposed algorithm we integrate grid-based mechanism into the online phase. In the reclustering (offline) phase, MCDAStream generates the final clusters, on demand by the user. Algorithm 1 shows the complete outline of MCDAStream clustering method. The user-defined parameters r (radius), λ (decay parameter), β (controlling threshold), α (intersecting parameter) are part of the base algorithm.

Algorithm 1 MCDAStream

Input: A Data Stream (DS), λ, r, μ,α, β
Output: Arbitrary shape clusters
Method:

```
1.  t_c ← 0
2.  t_r ← [1/λ log_2(βμ/(βμ−1))];
3.  while the data stream is active do
4.  read next data point p from DS at time step t_c
5.  //finding nearest micro-clusters within fixed radius r
6.  μC_s ← find nearest micro-clusters to p in μC_list
7.  if | μC_s | ≥ 1 then
8.  merging(p, μC_s);
9.  else
10. mapping(p, g);
11. end if
12. if t_c mod t_r = 0 then
13. pruning( );
14. end if
15. t_c ← t_c + 1
16. end while
17. // offline phase
18. if clustering request arrives then
19. for each i and j in μC_list where j>i do
20. if attr_ij ≥ α. μ then // α is an intersecting threshold
21. consider i and j for final clustering
22. generate final clusters using a variant of DBSCAN;
23. end for
24. end if
25. end if
```

In the online phase of MCDAStream, first, we read a data point p from the data stream, and we find all the micro-clusters (μC_s) which are within a fixed radius r from data point p. If one or more micro-clusters are found, then we update the weights and centers of all the micro-clusters incrementally by considering the fading function λ. The density attraction between the micro-clusters can be measured by counting the data points which are assigned to two or more micro-clusters. In this regard, micro-clusters which are closer to each other and which share an area of high density are grouped to the same macro-cluster. Updating the centers of micro-clusters may lead to collapsing micro-clusters. If the distance between two adjacent micro-clusters is less than r, then collapsing of those micro-clusters can be prevented by reverting their centers to previous positions. The pseudo code of the merging procedure is given in Algorithm 2. If no micro-cluster is found within the radius r from data point p (i.e., $| p\mu C_s | < 1$), then we map the data point p to the grid (see Algorithm 3).

Algorithm 2 merging (p, μC)

```
1.  merge p into μC_s
2.  for each micro-cluster i in μC_s do
3.  // update micro cluster density, last updated time and center
4.  μC_i[w] ← μC_i[w] * 2^{−λ(t_c − μC_i[t])} + 1
5.  μC_i[c] ← CF¹ / μC_i[w]
6.  μC_i[t] ← t_c
7.  //Update density attraction between the adjacent micro-clusters
8.  for each j in μC_s where j>i do
9.  attr_ij ← attr_ij * 2^{−λ(t_c − attr_ij[t])} + 1
10. attr_ii[t] ← t_c
11. end for
12. end for
13. // prevent collapsing clusters
14. for each i, j in μC_s X μC_s and j > i do
15. if dist(μC_i[c], μC_i[c]) < r then
16. revert μC_i[c], μC_i[c] to previous positions
17. end if
18. end for
```

If there is no micro-cluster falls within the radius r from p, then unlike many existing algorithms, instead of creating a new outlier micro-cluster with point p, we map p to the grid g in the outlier buffer. This will increase the performance of the algorithm regarding time complexity. When a data point p is added to the grid g, we update the grid characteristics (number of data points n_g, last updated time t_g, and grid weight w_g). If the number of data points (n_g) in the grid g reaches minimum points μ, then we check the grid density. If the grid g is a dense grid, then we form a new micro-cluster from the data points in the grid g. The related grid g of the new micro-cluster is removed from the grid list. The pseudo code of the mapping procedure is given in Algorithm 3.

Algorithm 3 mapping(p, g)

```
1.  map the new data point p to the grid g;
2.  Update CV(n_g, t_g, w_g); // update characteristic vector
3.  n_g ← n_g + 1;
4.  w_g(t_c) ← w_g(t_g) * 2^{−λ(t_c − t_g)} + 1;
5.  t_g ← t_c;
6.  if n_g ≥ μ and w_g ≥ β / (N_g(1−2^{−λ})) then
7.  create new μC(w, c, t); // creating new micro-cluster
8.  w ← w_g;
9.  CF¹ ← Σ_{j=1}^{n} f(t − T_ij) p_ij;
10. c → CF¹ / w;
11. t ← t_c;
12. remove grid g from grid_list;
13. end if
```

The pruning process is shown in Algorithm 4. At every t_r time steps, MCDAStream executes this procedure to remove outlier micro-clusters and sporadic grids. We check the weights of all the micro-clusters and girds at time t_r,The micro-clusters and the grids with the weights less than a threshold are removed from the μC list and grid list, respectively, to release the memory space and to improve the MCDAStream algorithms' processing speed.

Algorithm 4 pruning( )

```
1)  for all grid g in grid_list do
2)  Δ(t_c,t_g) = β(1−2^{−λ(t_c − t_g+1)}) / (N_g(1−2^{−λ}));
3)  //detecting and removing sporadic grids
4)  if w_g < Δ then
5)  delete grid g from the grid_list;
6)  end if
7)  end for
8)  for each micro-cluster i in μC_list do
9)  //detecting and removing outlier μCs
10) if μC_i[w] < β.μ then
11) delete μC_i from μC_list;
12) end if
13) end for
```

The online component of MCDAStream maintains micro-clusters, which capture the density area of the data streams. The algorithm also captures the density attraction between the micro-clusters. However, to generate the final clusters from the micro-clusters, we use a modified conventional clustering algorithm. In the reclustering phase, we use a variant of DBSCAN to form the final (macro) clusters.Each micro-cluster μC is considered as a pseudo point located at the center of μC with the weight w. To generate the final clustering results, we adopt the density-connectivity concept from DBSCAN algorithm [4].

4. Experimental results

In this section, we present the experimental results of MCDAStream and compare its performance with the three publicly available density-based clustering methods DenStream [5], DStream-II[1] and DBStream[16]. We have implemented/interfaced our proposed algorithm in a publicly accessible R-extension called stream [28]. The Stream is an extensible framework that provides an interface for experimenting, interfacing and implementing with algorithms for several data mining tasks. It includes a growing number of data stream generators and algorithms for clustering data streams.

4.1. Datasets

For the evaluation of MCDAStream, both synthetic and real datasets were used.
Synthetic Datasets: We experimented with two synthetic datasets called Noisy Mixture of Gaussians and DS3 from CHAMELEON [27] clustering algorithm. Example points for the two synthetic datasets are shown in Figure 1. The Noisy Mixture of Gaussians dataset contains two-dimensional data points with 5% noise. The dataset DS3 introduced for CHAMELEON algorithm consists of 8000 data points with six clusters of different shape, size, as well as random noise points.

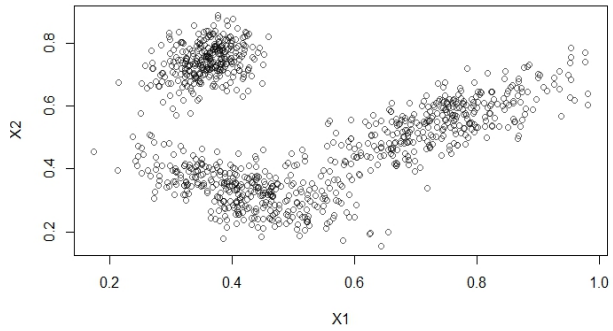

Fig. 1: A) Noisy Mixture of Gaussians.

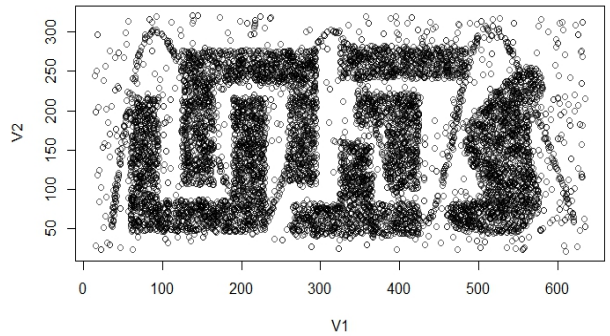

Fig. 1: B) DS3 of CHAMELEON Clustering Algorithm.

Figure 2 shows example clustering results of MCDAStream for Noisy Mixture of Gaussians and DS3 datasets. Micro-clusters are shown as circles in red color with a dotted circle representing each MC's assignment area. Blue crosses represent macro-clusters. Black lines connecting MCs describe the attraction between the micro-clusters.

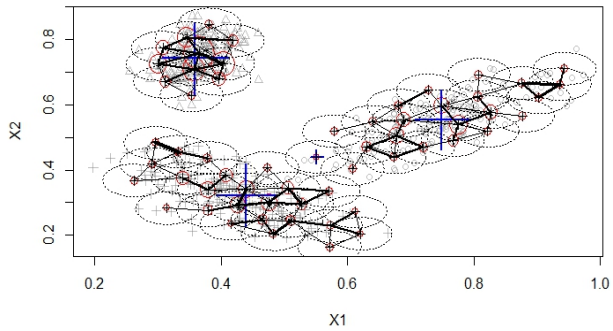

Fig. 2: A) Data Stream Clustering Result of MCDAStream on Noisy Mixture of Gaussians Dataset with Three Random Gaussians. Circles Represent Micro-Clusters and Crosses Represent Macro-Clusters.

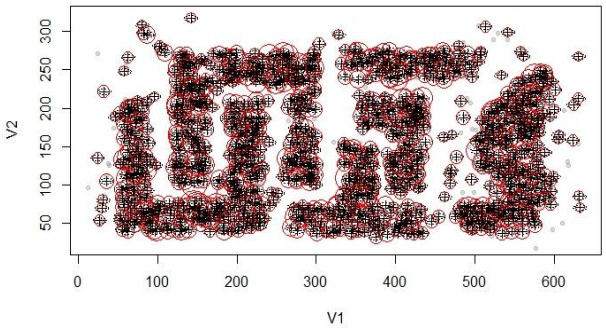

Fig. 2: B) Data Stream Clustering Result of MCDAStream on DS3 Dataset.

Real Dataset: To evaluate the MCDAStream and other existing algorithms capabilities with real-world data, we use a real dataset called the KDD CUP'99 (Network Intrusion Detection) dataset. KDD CUP'99 is a well-known and widely used dataset that was created for the Third International KDD Tools competition, and it is available from the UCI Machine Learning Repository [29]. It contains simulated network traffic with a wide variety of intrusions. It has 48,98,431 data points, and we considered all 34 numeric features out of 42 available attributes for clustering.

4.2. Clustering quality evaluation

To evaluate the clustering quality of MCDAStream, we use a simple measure, called purity[5]. The clustering purity is computed only for the data objects arriving in a predetermined window (horizon) since the weight of the data objects decay over time
At first, we experiment the clustering quality of MCDAStream on synthetic datasets. Figure 3 shows the purity results of MCDAStream compared to DBStream, DenStream, and D-Stream on Noisy Mixture of Gaussians and DS3 datasets. We can note that the clustering purity of MCDAStream is always higher than 98% and it is higher than the existing clustering algorithms.
We have also experimented clustering purity of MCDAStream on Network Intrusion Detection (KDD CUP'99) real dataset. Figure 4 shows the average purity results of MCDAStream on Network Intrusion Detection (NID) dataset. MCDAStream runs very well regarding high average purity on real dataset compared to DBStream, DenStream, and DStream.

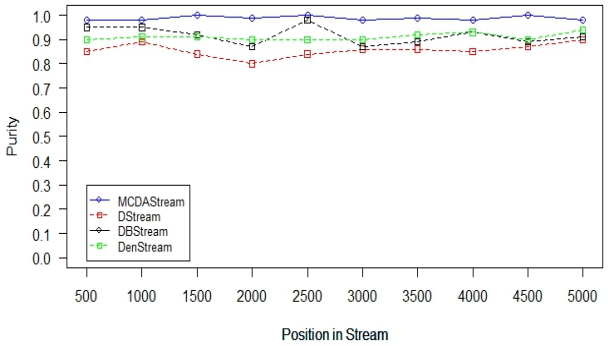

Fig. 3: A) Cluster Purity of MCDAStream on Noisy Mixture of Gaussians.

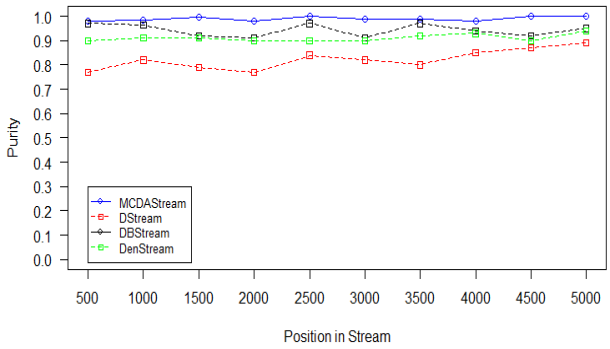

Fig. 3: B) Cluster Purity of MCDAStream on DS3 of CHAMELEON.

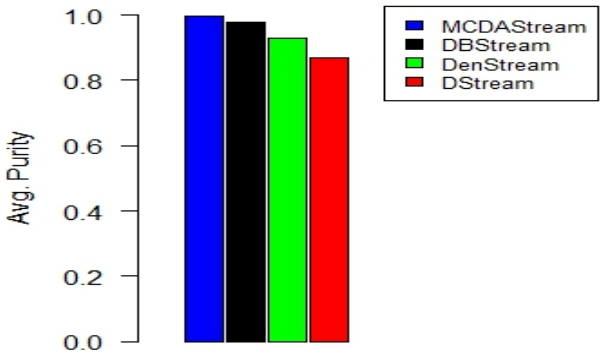

Fig. 4: Average Purity Results of MCDAStream on NID Real Dataset.

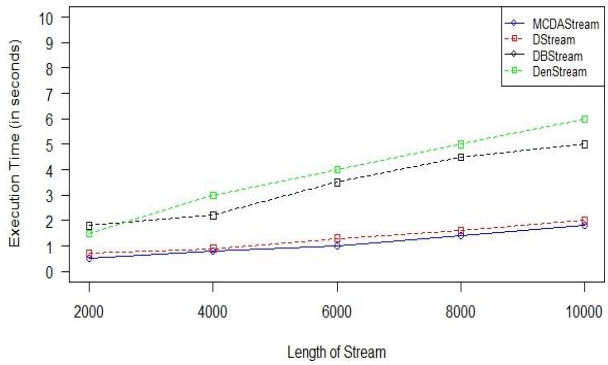

Fig. 5: Execution Time vs. Length of Stream.

4.3 Scalability results

Execution Time: The efficiency of the MCDAStream clustering algorithm is measured by the execution time. The execution time of MCDAStream is effected by the number of data points processed per time unit. We use the KDD CUP'99 dataset to evaluate the efficiency of MCDAStream against DBStream, DenStream, and D-Stream. Figure 5 shows the execution time in seconds for KDD CUP'99 dataset. We can note that the execution time of MCDAStream and other clustering methods grow linearly as the stream proceeds. We can also note that the MCDAStream has lower execution time compared to DBStream, DenStream, and D-Stream. The time complexity of MCDAStream is minimized by using the grid-based method in the online phase. It allows us to reduce the merging time complexity from o(μC) to o(1). Searching nearest micro-clusters within fixed radius r from point p can be done using linear search in O(ndk), where n is the number of points clustered, d is the dimensionality of the data, and k is the number of micro-clusters.

Memory Usage: Space complexity of MCDAStream algorithm depends on the number of micro-clusters and grids that are stored in μC_list and grid list, respectively. The memory usage of MCDAStream is o(μC+g) which is measured by the number of micro-clusters and grids.

In data steam clustering, each point is processed individually, and we have captured some statistics averaged over 500 point intervals for Noisy Mixture of Gaussians dataset. Table 1 shows the number of micro-clusters used by the algorithms MCDAStream, DBStream, and DenStream for Noisy Mixture of Gaussians dataset. These numbers are directly associated to the memory used by the respective algorithms. We can note that MCDAStream uses less memory compared to DBStream and DenStream algorithms.

Table 1: Number of Micro-Clusters for Noisy Mixture of Gaussians Dataset

No. of Points	No. of micro-clusters MCDAStreamr = 0.05	DBStreamr = 0.05	DenStreamepsilon = 1
500	33	54	57
1000	48	67	89
1500	63	70	96
2000	74	77	106

5. Conclusion

In this paper, we have presented MCDAStream, an efficient density-based algorithm for clustering data streams. The algorithm clusters the data stream based on micro-cluster density and attraction. The algorithm captures the density attraction between the micro-clusters in the online phase, and it uses the same information for reclustering in the offline phase. Our MCDAStream clustering algorithm uses density micro-clustering and density grid-based clustering to find high-quality clusters with considerably less computation time and memory. To evaluate the performance of MCDAStream, experiments were conducted on both synthetic and real datasets. The experimental results show that the proposed clustering algorithm has high quality and low computation time compared to existing methods.

References

[1] Chen Y, Tu L, "Stream Data Clustering Based on Grid Density and Attraction." ACM Transactions on Knowledge discovery Data, 3(3): Article No. 12, 2009.

[2] Han J. and Kamber, M. "Data Mining Concepts and Techniques." 2nd Ed. Burlington: Morgan Kauffman, 2006.

[3] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. a. Gama, "Data stream clustering: A survey," ACM Computing Surveys, vol. 46, no. 1, pp. 13:1–13:31, Jul. 2013.

[4] Ester M., Kriegel H., Sander J., and Xu X. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: Proc. of 2nd International Conference on Knowledge Discovery, pp. 226–231, 1996.

[5] Cao F, Ester M, Qian W, Zhou A. "Density-Based Clustering Over an Evolving Data Stream with Noise." In Proc. the SIAM Conference on Data Mining, April 2006, pp.328-339.https://doi.org/10.1137/1.9781611972764.29.

[6] Tasoulis D K, Ross G, Adams N M. "Visualising the Cluster Structure of Data Streams." In Proc. the 7th International Conference on Intelligent Data Analysis, Sept. 2007, pp.81-92.https://doi.org/10.1007/978-3-540-74825-0_8.

[7] Menasalvas E, Ruiz C, Spiliopoulou M. "C-DenStream: Using Domain Knowledge on a Data Stream." In Proc. the 12th International Conference on Discovery Science, Oct. 2009, pp.287-301.

[8] Jing K, Liu L, Guo Y et al. “A Three-Step Clustering Algorithm over an Evolving Data Stream.” In Proc. the IEEE Int. Conf. Intelligent Computing and Intelligent Systems, Nov. 2009, pp.160-164.
[9] Ren J, Ma R. “Density-Based Data Streams Clustering over Sliding Windows.” In Proc. the 6th Int. Conf. Fuzzy systems and Knowledge Discovery, Aug. 2009, pp.248-252.https://doi.org/10.1109/FSKD.2009.553.
[10] Lin J, Lin H. “A Density-Based Clustering over Evolving Heterogeneous Data Stream.” In Proc. The 2nd Int. Colloquium on Computing, Communication, Control, and Management, Aug. 2009, pp.275-277.https://doi.org/10.1109/CCCM.2009.5267735.
[11] Dunham M, Isaksson C, Hahsler M. “SOStream: Self Organizing Density-Based Clustering over Data Stream.” In Lecture Notes in Computer Science 7376, Perner P (ed.), Springer Berlin Heidelberg, 2012, pp.264-278.
[12] Zimek A, Ntoutsi I, Palpanas T et al. “Density-Based Projected Clustering over High Dimensional Data Streams.” In Proc. The 12th SIAM Int. Conf. Data Mining, April 2012, pp.987-998.
[13] Spaus P, Hassani M, Gaber M M, Seidl T. “Density-Based Projected Clustering of Data Streams.” In Proc. the 6th Int. Conf. Scalable Uncertainty Management, Sept. 2012, pp.311-324.
[14] Pizzuti C, Forestiero A, Spezzano G. “A Single Pass Algorithm for Clustering Evolving Data Streams based on Swarm Intelligence.” Data Mining and Knowledge Discovery, 2013, 26(1): 1-26.https://doi.org/10.1007/s10618-011-0242-x.
[15] Amineh A, Teh Ying W “LeaDen-Stream: A Leader Density-Based Clustering Algorithm over Evolving Data Stream.” Journal of Computer and Communications, pp. 26-31, 2013.
[16] Hahsler M, and Matthew B. “Clustering Data Streams Based on Shared Density between Micro-Clusters.” IEEE Transactions on Knowledge and Data Engineering, 2016.https://doi.org/10.1109/TKDE.2016.2522412.
[17] Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases. In Proc. ACM SIGMOD International Conference on Management of Data, June 1996, pp.103-114.https://doi.org/10.1145/233269.233324.
[18] Li J, Gao J, Zhang Z, Tan P N. An incremental Data Stream Clustering Algorithm Based on Dense Units Detection. In Proc. the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, May 2005, pp.420-425.
[19] Chen Y, Tu L. Density-Based Clustering for Real-Time Stream Data. In Proc. the 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Aug. 2007, pp.133-142.https://doi.org/10.1145/1281192.1281210.
[20] Tan C, Jia C, Yong A. A Grid and Density-Based Clustering Algorithm for Processing Data Stream. In Proc. the 2nd Int. Conf. Genetic and Evolutionary Computing, Sept. 2008, pp.517-521.
[21] Ng W K, Wan L, Dang X H et al. Density-Based Clustering of Data Streams at Multiple Resolutions. ACM Trans. Knowledge Discovery from Data, 2009, 3(3).
[22] Ren J, Cai B, Hu C. Clustering over Data Streams Based on Grid Density and Index Tree. Journal of Convergence IT, 2011, 6(1): 83-93.
[23] Yang Y, Liu Z, Zhang J et al. Dynamic Density-Based Clustering Algorithm over Uncertain Data Streams. In Proc. the 9th Int. Conf. Fuzzy Systems and Knowledge Discovery, May 2012, pp.2664-2670.https://doi.org/10.1109/FSKD.2012.6233800.
[24] Teh Ying W, Amini A, DENGRIS-Stream: A Density-Grid Based Clustering Algorithm for Evolving Data Streams over Sliding Window. In Proc. International Conference on Data Mining and Computer Engineering, Dec. 2012, pp.206-210.
[25] Kaur S, Bhatnagar V, Chakravarthy S. Clustering Data Streams using Grid-Based Synopsis. Knowledge and Information Systems, June 2013.
[26] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, “Density-Based Clustering of Data Streams at Multiple Resolutions,” ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 3, pp. 1–28, 2009.https://doi.org/10.1145/1552303.1552307.
[27] George K, Eui-Hong H, Vipin K., “CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling” IEEE Computer, pp. 68-75, August 1999.
[28] M. Hahsler, M. Bolanos, and J. Forrest, stream: Infrastructure for Data Stream Mining, 2015, R package version 1.2-2.
[29] Bache K, Lichman M (2013). \UCI Machine Learning Repository." URL http://archive.ics.uci.edu/ml.