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

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

1Research Scholar, JNTUA, Ananthapuramu
2Department of CSE, JNTU College of Engineering, Ananthapuramu
*Corresponding author E-mail: shyam4@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], SDDSStream [9], HDenStream [10], SOStream[11], HDDStream [12], PreDeConStream [13], FlockStream [14], LeaDenStream [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 (\(\mu C\)) 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 \(\mu C\)s. 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 micro-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], DUCStream [23], DENGRIS-Stream [24], ExCC [25], which are belongs to density grid-based clustering algorithms. 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 summarize 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 reclustering 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 interconnectivity 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, LeanDStream [15] introduced a concept in which micro-clusters are represented by mini-level 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 \( t \), is defined as follows:

\[
w_p(t_c) = w_p(t_0) * f(t_c-t_0) = 2^{-\lambda (t_c-t_0)}
\]

Where \( w_p(t_0) \) is the weight of the data point \( p \) at time \( t_0 \) and \( t \geq t_0 \). The initial weight of a data point is assumed to be 1.

Definition 2: (Grid density \( w_g \)): Density of a grid at current time \( t \), is defined as follows:

\[
w_v(t_c) = \sum_{p \in g} 2^{-\lambda(t_c-t_0)}
\]

The weight of a grid is updated at current time \( t_c \), with the last updated time \( t_0 \) as follows:

\[
w_v(t_c) = w_v(t_0) + f(t_c-t_0) + 1
\]

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 grid 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 \( \mu_C \)): A micro-cluster \( (\mu_C) \) at time \( t \) is determined as \( \mu_C \in \mathcal{W}(c, t) \) 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) \( w = \sum_{j=1}^{n} f(t-T_j) \) is the weight, \( w \geq \beta \mu_c \), where \( \beta \) is the parameter to determine the threshold of an outlier relative to \( \mu_C \) and \( \mu \) represents the minimum number of points.

2) \( C=\bar{F} \) is the center of \( \mu_C \), and \( C=F=\sum_{j=1}^{n} f(t-T_j)p_j \) is the weighted linear sum of the data points \( t \) is the last updated time of \( \mu_C \)

3) If all the data points are merged into the same \( \mu_C \), then the maximum weight of the micro-cluster is defined as follows:

\[
\mu_C \left[ \text{wmax} \right] = \lim_{t_0 \to \infty} \frac{1-2^{-\lambda(t_0-t_0)}}{1-2^{-\lambda}} = 1 \left( \frac{1}{1-2^{-\lambda}} \right)
\]

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 \( \lambda \)): 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 \( \lambda \) is defined as follows:

\[
\Delta(t_c,t_0) = \frac{\beta}{N_g} \sum_{j=0}^{t_c-t_0} 2^{-\lambda j} \leq \frac{\beta}{N_g(1-2^{-\lambda})}
\]
Where \( t_c \) is the current time, and \( t_p \) is the last updated time of a grid.

Definition 6 (Pruning time): For each micro-cluster \( \mu_c \), if no new data point is added to it, the weight of \( \mu_c \) will decay gradually. If \( \mu_c[w] < \beta \mu_c[w] \), then \( \mu_c \) 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 \( t_c \). Pruning time \( t_c \) is the minimum time for a \( \mu_c \) grid in time step \( t_c \) to be converted to an outlier to \( t_c \) (\( t_c > t \)) which is determined as follows:[5]:

\[
t_c = \frac{1}{\lambda} \log_2 \left( \frac{\beta \mu_c[w]}{\mu_c[w]} \right)
\]

3.2. The MCDADStream algorithm

At each time step, the online component of MCDADStream 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 clustering (offline) phase, MCDADStream generates the final clusters, on demand by the user. Algorithm 1 shows the complete outline of MCDADStream clustering method. The user-defined parameters \( \lambda \) (radius), \( \beta \) (intersecting parameter) are part of the base algorithm.

Algorithm 1 MCDADStream
Input: A Data Stream (DS), \( \lambda, \beta, \mu, \alpha, \beta \)
Output: Arbitrary shape clusters
Method:
1. \( t_c \leftarrow 0 \)
2. \( t_c \leftarrow \frac{1}{\lambda} \log_2 \left( \frac{\beta \mu_c[w]}{\mu_c[w]} \right) \)
3. while the data stream is active do
4. read next data point \( p \) from DS at time step \( t \)
5. // Finding nearest micro-clusters within fixed radius \( \rho \)
6. \( \mu_c \leftarrow \text{Find nearest micro-clusters to } p \text{ in } \mu_c \) \text{list}
7. if \( | \mu_c | \geq 1 \) then
8. merging(\( p, \mu_c \));
9. else
10. mapping(\( p, g \));
11. end if
12. if \( t_c \mod t = 0 \) then
13. pruning(\( \lambda \));
14. end if
15. \( t \leftarrow t + 1 \)
16. end while
17. // offline phase
18. if clustering request arrives then
19. for each \( i \) and \( j \) in \( \mu_c \) \text{list where } \( j > i \) do
20. if \( \Delta t_{ij} \geq \mu, \mu \) then // 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 MCDADStream, first, we read a data point \( p \) from the data stream, and we find all the micro-clusters (\( \mu_c \)) 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 \( \lambda \). 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., \( | p \mu_c[w] | < 1 \)), then we map the data point \( p \) to the grid (see Algorithm 3).

Algorithm 2 merging (\( p, \mu_c \))
1. merge \( p \) into \( \mu_c \);
2. for each micro-cluster \( i \) in \( \mu_c \), do
3. //update micro-cluster density, last updated time and center
4. \( \mu_c[w] = \mu_c[w] + 2^{-\lambda(t_c-t_i)} \)
5. \( \mu_c[t] = t_c \)
6. \( \mu_c[c] = c \)
7. //Update density attraction between the adjacent micro-clusters
8. for each \( j \) in \( \mu_c \), where \( j > i \) do
9. attr\( _i \leftarrow \mu_c[w] \); 
10. attr\( _i \leftarrow \mu_c[w] + 1 \)
11. end for
12. end for
13. \( \mu_c \) to previous positions
14. end if
15. \( \mu_c \) to previous positions
16. end if
17. // prevent collapsing clusters
18. end if
19. end for
20. end if
21. end for
22. end for
23. end for
24. end if

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 \( \mu \), 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 \( \Delta V(n_g, t_g, w_g); // update characteristic vector \)
3. \( n_g = n_g + 1 \); 
4. \( w_g(t_c) = w_g(t_c) \times 2^{-\lambda(t_c-t_i)} \); 
5. \( t_g = t_c \);
6. if \( n_g \geq \mu \) and \( w_g \geq \mu \), then
7. create new \( \mu_c \); // creating new micro-cluster
8. \( \mu_c = \mu_c \times \mu_c \); 
9. \( \mu_c = \sum_{i} \mu_i \); 
10. \( c = \frac{\sum_{i} \mu_i \cdot c_i}{\sum_{i} \mu_i} \); 
11. \( t_c = t_g \);
12. remove grid \( g \) from grid list;
13. end if
14. end if

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

Algorithm 4 pruning( )
1. for all grid \( g \) in grid list do
2. \( \Delta(t_c) = \frac{1}{2^{-\lambda(t_c-t_i)}} \); 
3. \( n_g = n_g - 2^{-\lambda(t_c-t_i)} \); 
4. //detecting and removing sporadic grids
5. if \( w_g \geq \Delta \) then
6. delete grid \( g \) from \( \mu_c \) list;
7. end if
8. end if
9. for each \( \mu_c \) list do
10. delete \( \mu_c \); //detecting and removing outlier \( \mu_c \); 
11. if \( w_g | \mu_c[w] | < \beta \mu_c[w] \); 
12. end if
13. end for
14. end if
15. end for
16. end if
17. end if
18. 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 \( \mu C \) is considered as a pseudo point located at the center of \( \mu C \) 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.

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 DStream 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.
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