M.S. THESIS

Hierarchical Density-Based Clustering for Data Stream over Sliding Window

슬라이딩 윈도우를 통한 데이터 스트림에 대한 계층적 밀도 기반 클러스터링

BY

Enkhbat Undraa

August 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Hierarchical Density-Based Clustering for Data Stream over Sliding Window

슬라이딩 윈도우를 통한 데이터 스트림에 대한 계층적 밀도 기반 클러스터링

지도교수 문봉기

이 논문을 공학석사 학위논문으로 제출함

2022 년 04 일

서울대학교 대학원
컴퓨터 공학부

Enkhbat Undraa

Enkhbat Undraa 의 석사 학위논문을 인주함

2022 년 06 월

Abstract

Hierarchical Density-based Clustering for Data Stream over Sliding Window

ENKHBAT UNDRAA
Department of Computer Science & Engineering
College of Engineering
Seoul National University

Data stream has become a hot topic of interest in recent years as its applications are increasing drastically. In addition, data streams are being continuously generated as a result of excessive usage of electronic devices and network. Thus, data streams have these unique characteristics differing to the static data such as speedy data point generation and possibly get to an unbounded size over time.

Due to the distinctive nature of data stream as mentioned above, requirements for the data stream clustering algorithms are becoming more and more complex. The basic requirements for clustering algorithms for static data are being able to extract arbitrary shape and numbers of clusters within the data. In addition, it is crucial for stream clustering algorithms to process incoming data fast and efficiently due to the time and space limitation. Although existing density-based stream clustering algorithms successfully find clusters of arbitrary shape and numbers within incoming stream, it has two user parameters density (ε) and minimum points per cluster(minPts) that has to be tuned carefully to obtain the desired clustering outcome.

In this paper, we propose a stream clustering algorithm called StreamHD, which is based on a hierarchical density-based clustering algorithm that can detect clusters of arbitrary shapes within the data stream. The proposed algorithm independently detects density thresholds of the clusters without much user intervention. In addition, StreamHD requires only two user parameters, window size and minPts which determines the number of neighboring points of the given point to consider when calculating core density and also determines the minimum cluster size. It can be said that StreamHD has the least user intervention among the stream clustering algorithms. Furthermore, experiment results on real and synthetic datasets have shown that our proposed algorithm performs the best among the comparison algorithms in terms of window processing time and cluster quality.

Keyword: Data Stream, Hierarchical Clustering, Density-based, Sliding window
Student Number: 2020-20623

Table of Contents

List of figures

List of tables

Chapter 1. Introduction

The aim of clustering algorithms is to exploit the underlying distribution of the data and partition the dataset into groups based on the given criterion (e.g., similarity, distance, density etc). The requirements for static data clustering algorithms include cluster discovery of arbitrary shape, an ability to deal with noisy data, minimal requirement for prior knowledge of the domain and density distribution.

When the data satisfies 3V characteristics of big data, it becomes almost impossible for the data to be processed as static data; thus, big data is considered as a data stream. Data streams have characteristics that differs from static data such as speedy data arrival and infinite length [1]. When clustering data stream or big data, stream clustering algorithms face a strict time and space limitation problems due to the incoming data stream that is fast-arriving and potentially unbounded in size [2].

There are several approaches that were proposed to solve the stream clustering problems. One of them is incremental clustering method [3], which has shown to be effective in data warehousing applications, but it cannot perform well with a limited memory and also faces computation overhead problem with insertion and deletion operations. Therefore, most of the existing stream clustering algorithms adopt online-offline-phase clustering model [4] to solve the temporal and spatial limitations of data stream. It considers infinite length of data stream and limited memory. In the online phase, summarization of the data stream is continuously performed and the summarized data is maintained in order to keep traces of the changing distribution of the data. Upon request from the user, the offline phase performs a static clustering algorithm on the summarized data to deliver the final clustering result at the time of the request.

Many existing stream clustering algorithms considered evolving nature of the data stream that even the older data points has an influence on the final clustering results; thus, adopts damped model the data points gradually fade and eventually results in cluster expiry.

Models adopting the damped window model relies heavily on the incoming order of the data stream and newly entered data points considered to have more weight than the older ones. Although there are several applications that advantages from this model, there are still other applications such as real-time traffic congestion reporting system that gains benefit from the clustering of unbiased data in a certain time frame. By adapting sliding window model for the stream data processing, only the recent data points in given time frame would be considered.

Some of the recent online-offline phase stream clustering algorithms adopted density-based clustering algorithm such as DBSCAN [5] as the offline-phase clustering algorithm. Although DBSCAN-based streaming algorithms were able to successfully detect arbitrary shaped data clusters within tolerable elapsed time, they still lack when it comes to the number of user parameters that needs careful tuning for better clustering results. When there are plenty of user parameters, it significantly increases user intervention to the algorithm and therefore, without careful parameter tuning, user may not be able to get the desired clustering quality from the algorithm. In other words, the more user parameter there are, the more difficult it will become to tune the parameters; thus, might result in decreased cluster quality.

With all the above drawbacks and limitations in mind, we propose a stream clustering algorithm called StreamHD which is based on a hierarchical density-based clustering algorithm HDBSCAN that can discover cluster from data stream with fewer user parameter. HDBSCAN was modified to be used as a clustering algorithm in the offline phase for micro cluster. With adopting sliding window processing model, this algorithm focuses on the recent data within given interval and considers all the data points with same weight while adapting to concept drifts. In online-phase of StreamHD, micro cluster structure keeps the statistical information of the data stream. Micro cluster radius threshold is auto-tuned based on the incoming data stream which makes the algorithm more sensitive to concept drift. In addition, StreamHD has very little user intervention by having only one user parameter outside of window size which is minPts, that determines the number of neighbor points to consider when calculating core distance of a point and also the minimum cluster size. StreamHD was then experimented in comparison to two stream clustering algorithms called DBStream and StreamSW, both of which adopted online-offline-phase clustering framework and showed great performance in previous studies [6]. With extensive experiments, our proposed algorithm, StreamHD has shown to produce good quality clusters with low latency.

The main contributions of this study are as follows:

- We propose a stream clustering algorithm StreamHD that shows the best cluster quality compared to the DBStream and StreamSW algorithms in all the input data stream with any density distribution.
- Our proposed algorithm StreamHD has only two user parameters which significantly decreases user intervention to the algorithm.
- In addition, StreamHD shows the shortest processing time per window in comparison to the DBStream and StreamSW algorithms which implies that it has shown the best performance.
- Micro-clusters of StreamHD are also well adapted to the concept drift that its radius threshold is calculated at the beginning of each window.

Chapter 2. Background

2.1. Data stream

Big data applications have been on the rise in recent years due to the fact that a large amount of data such as sensor data, network flow data is being generated at high-speed owing to the booming internet usage and technology advances in both hardware and software.

Furthermore, conventional data processing algorithms are no longer able to process big data within tolerable elapsed time because of special and temporal limitation problem [7].Thus, such big datasets are being considered as a data stream in recent data mining approaches [8].

Data stream is an ordered data sequence that is usually assumed to have unbounded size and the arrival rate of the data stream is very fast that there is not much time to process each of the data points. Therefore, repeated or random access to the data stream is considered almost impossible. With limited storage space and processing time, applications dealing with data streams are expected to focus on keeping the necessary information from the incoming data stream while overcoming the limitations mentioned. In addition, data stream exhibits concept drift over time where the underlying distribution changes over time [9].

2.2. Data stream processing window models

The underlying distribution of the data stream often changes overtime, which is called concept drift or concept shift. Stream processing applications often has interest in the recent data or a real-time data. Data processing window models are often employed to capture recent trend of the incoming data stream.
Window processing approaches aims to minimize the influence of historic outdated data to the recent data pattern and controls which part of the data stream would influence the pattern. Such window

processing approaches aids concept drift adaptability when adopted in stream clustering algorithms.

There are a few window-processing models, and the following two are the most widely known and used among the data stream clustering approaches.

2.2.1. Damped window model

In the damped window processing model, the historical concepts are considered to have impact on the clustering results. The damped window processing model assigns weight to the data points or micro clusters based on the incoming order such that the recent data points are assigned higher weight than those of the past.

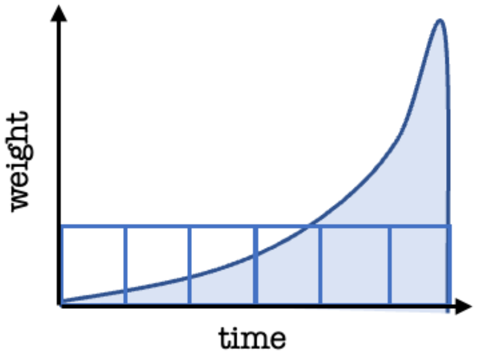


Figure 1. Damped window model

In Figure 1, the blue part shows weight distributed to the data stream based on the incoming order in damped window model.
As time flows, weight of each data point is decreased by a factor that determines the rate of decay called decay factor (**λ**). Once the data point fades to the point where it does not carry any weight, it is considered as an expired data point and no longer affects the pattern of the existing data points.

2.2.2. Sliding window model

The sliding window model only considers the recent datapoints within the given interval in the data stream. All the data points within the window are given the same weight and there is no bias towards the incoming order of the data points. First-In-First-Out principle is often utilized in this model, where the oldest data point of the window is removed when a new data point comes into the window. Figure 2

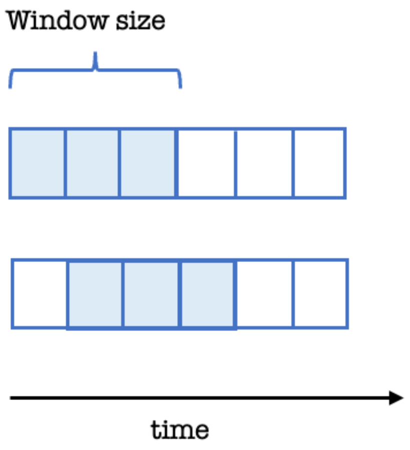


Figure 2. Sliding window model

has shown sliding window model concept where the window slides as the time goes. Size of the window can be fixed or variable-length based on the application such that a smaller window size adapts quickly to the concept drift, whereas a larger window size considers more data points; thus, the summarization accuracy can be higher in stable streams.

2.3. Data stream clustering

Data clustering is an unsupervised learning approach that can exploit several information from given data. When in streaming data environment, there exists a few obstacles for the static clustering algorithms.

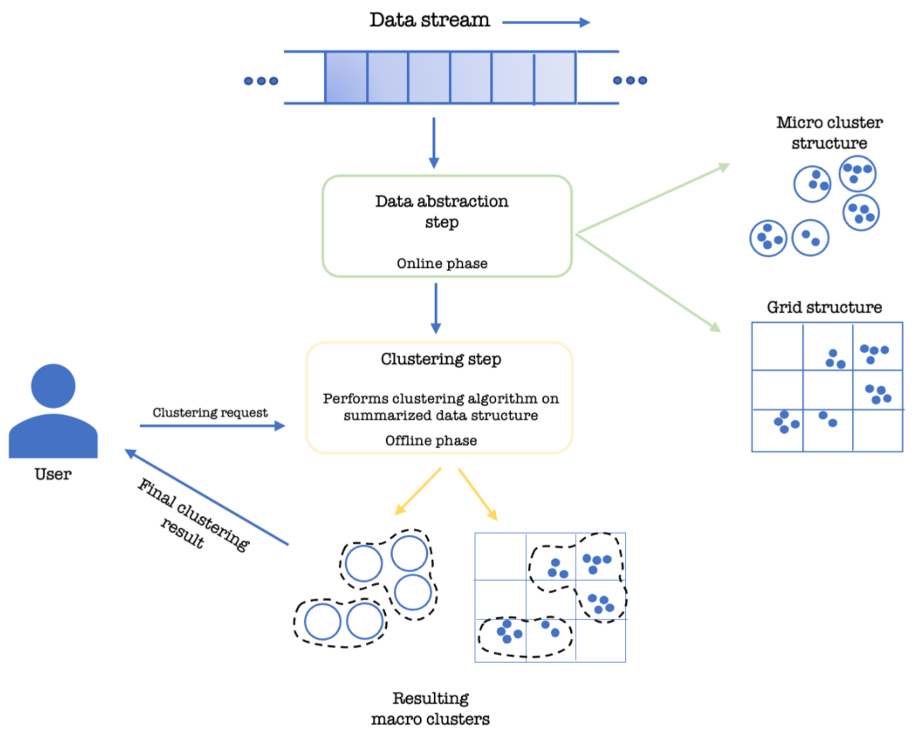


Figure 3. Online-offline-phase framework for stream clustering

Due to the data stream characteristics, stream clustering algorithms must process data once at arrival, and detect arbitrary shapes of clusters within data stream fast with as little latency as possible with restricted time and space resource. Thus, it is required for stream clustering algorithms to rapidly process and possibly summarize the massive and continuous incoming data in order to detect emerging data clusters. Also, due to the memory limitation, expired data points should be pruned periodically. Online-offline-phase framework is a widely used framework among the stream clustering algorithms and is shown in Figure 3.

Online phase, also known as a data abstraction step of the framework constantly maintains summarized statistics of the data points with the preferred approximation structure.

Offline phase is triggered by the clustering request from the user. In this step, the summarized statistics of the data stream is clustered into macro clusters and returned to user as a final clustering result. Density-based clustering approaches has shown remarkable results in detecting arbitrary shaped data clusters within data. Thus, density based clustering approaches were often used in the stream clustering algorithms offline phase.

2.4. Data stream approximation structures

Approximation-based stream clustering algorithms often approximate the data stream by maintaining the summarized information within data structures [10]. The following are the most widely used summarization data structures used in the approximation-based stream clustering algorithms.

2.4.1. Micro cluster structure

Micro cluster structure is summarization structure that keeps the statistical information about compactly positioned datapoints in given space and represents the datapoints as one micro cluster [11]. This structure is an extension of cluster feature structure - denoted by $\overline{\text{CF}}$ - proposed in the renowned BIRCH algorithm [12].

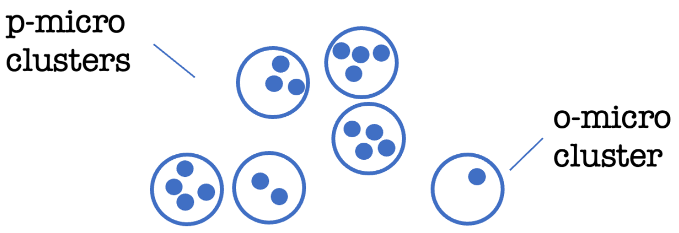


Figure 4. Micro cluster structure and types of micro clusters

In Figure 4, two types of micro clusters are shown. The blue points represent the data points from the data stream. Micro cluster is represented by a tuple MC ($\overline{\text{CF1}}$, $\overline{\text{CF2}}$, w, c, r, $\overline{\text{ts}}$) where $\overline{\text{CF1}}$ is the linear weighted sum of the points, $\overline{\text{CF2}}$, is the squared weighted sum of the points, w is the weight of the micro cluster which is the total number of data points belonging to the MC, c is the center, r is the radius and $\overline{\text{ts}}$ is the array of timestamps of the datapoints belonging to the MC . Radius of the MC is calculated based on the values of $\overline{\text{CF1}}$ and $\overline{\text{CF2}}$ as following.

$$r = \sqrt{\frac{\overline{\mathrm{CF2}}}{w} - \left(\frac{\overline{\mathrm{CF1}}}{w}\right)^2}$$

There are usually 2 types of micro clusters are maintained in the summarization step of a stream clustering algorithm. $p-micro\ cluster$ is a MC that satisfies following thresholds: $\mathrm{w} > \mathrm{MinPts}$ and $\mathrm{r} < \varepsilon$. $o-micro\ cluster$ is a MC that satisfies $\mathrm{r} < \varepsilon$ threshold but does not satisfy the $\mathrm{w} > \mathrm{MinPts}$ threshold which implies that the outliers are maintained by $o-micro\ cluster$. MinPts and ε thresholds are a user parameter in most of the algorithms that adopts micro cluster method and the value of the thresholds are usually determined by the user parameter values.

2.4.1. Grid structure

Grid structure maintains data points by mapping them into the grid cells and keeps the statistical information in a grid characteristic vector CV [13]. The CV is a tuple $CV(w_g, t_g, l)$ where w_g is the grid density, t_g is the timestamp of the grid, l is a label where $l = \{active, expired, dense, sparse\}$.

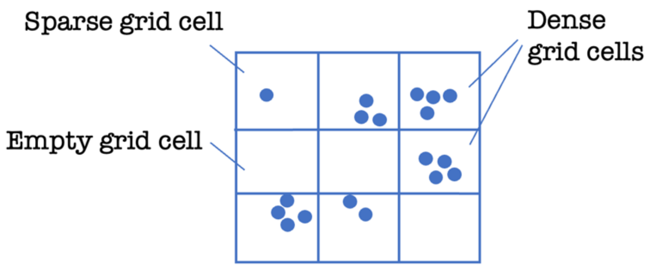


Figure 5. Grid structure and type of grid cells

Grid density w_g is calculated based on the number currently mapped data point and controlling threshold β. Grid density threshold is defined as $\frac{\beta * w_{gmax}}{n}$,where w_{gmax} is the sum of all grid cell weight and n is number of all grid cell. Grid is considered dense if the grid cell weight is greater than or equal to density threshold, and sparse elsewhere. Macro clusters are generated based on the adjacent dense grid cells. In Figure 5, types of grid cells and grid structure is shown.

When adopting grid structure for the summarizing structure of an algorithm, pruning step is crucial because once a grid cell in the adjacent dense cells become sparse or empty due to the concept drift of the data stream, it seriously affects the clustering quality.

2.5. HDBSCAN algorithm

Hierarchical density-based clustering algorithm HDBSCAN algorithm produces clusters of varying density. It has user parameter minPts that determines the number of neighbors to consider when calculating core points and is also used as the value for minimum cluster size for the resulting clusters. It first builds a weighted connected graph from the data. Then, it builds minimal spanning tree on top of the graph which represents hierarchy of connected components from completely connected to completely disconnected at varying threshold levels. This way, clusters of various density distribution can be discovered. Then flat clustering can be extracted to produce final clustering result. The modified version of this algorithm is used in the StreamHD algorithm and will be explained further in detail in Chapter 4.

Chapter 3. Related Works

For many years, different clustering approaches has been proposed and being used widely in various applications. There are number of stream clustering problems, and many approaches are still being proposed to solve them.

The CLUSTREAM [4] algorithm proposed an online-offline-phase stream clustering framework to overcome the temporal limitation problem. The online phase maintains micro-clusters and the offline phase returns macro clusters as clustering result by taking the micro clusters as an input data. The micro clusters store the approximated statistical information about the data stream while the macro clustering uses this data to produce the clustering result whenever the clustering request comes in. This online-offline-phase model has been widely used among the existing stream clustering methods.

Density-based clustering algorithms partition the data into clusters based on the underlying density profile of the data. Dense regions of arbitrary shape and has been successfully exploited from the data with density based clustering approaches, unlike distance based clustering algorithms. However, careful tuning of the parameters is needed and the density-based clustering algorithms often face with cluster quality trade-off. In streaming data environment, another biggest challenge is to estimate density of the data in single iteration because of the temporal limitation problem. Although stream clustering algorithms that adopted incremental clustering method such as Incremental DBSCAN [3] has shown good performance in some data warehousing environment, it needs a good memory space; thus, cannot perform well with limited memory. DISC [14] is a recent state-of-the-art method that expedites the incremental operations under the batch updates within sliding window model. This method reduces the incremental insertion and deletion costs by removing the redundant retrievals of data points during the batch update along with the traversal and index optimization updates. However, it still faces drawback of the dynamic connectivity problem.

Online-offline-phase summarization model has been a fairly good solution to the temporal and special limitation problems. Denstream [15] is a density-based stream clustering approach which utilized the micro cluster data structure as the online phase and the most widely used density-based algorithm DBSCAN as the offline phase. Micro cluster size is not pre-defined in this algorithm, which means micro cluster can grow in unbounded size. Denstream suffers from computation overhead and reduced cluster quality when the incoming data is sparse.

Grid based clustering approaches are also part of the density-based clustering approaches. A grid structure is used to estimate the underlying density profile. DStream [13] is a stream clustering algorithm that utilized grid structure as the online-phase and DBSCAN as the offline-phase. Grid structure differs from the micro-cluster structure by the frequency of the density status change. Dense and sparse grids are maintained and empty grids are periodically pruned in the online phase. Downside of Dstream algorithm is that when the empty grids are not pruned frequently, it affects the cluster quality of the DStream. The ρ-double-approximate DBSCAN [16] is dynamic version of the approximate DBSCAN [17]. It manages a set of grouped points as a cell to reduce the computational cost and uses a grid-based index to perform the approximate range searches. Theoretically, it supports the insertion and deletion in a near-constant time with a grid-based approach. However, it does not show practically good performance for the appropriate parameter settings. StreamSW [18] is a stream clustering algorithm that adapted both grid and micro cluster structure for the online-phase and DBSCAN as the offline-phase clustering algorithm. This algorithm utilizes sliding window model for processing the data stream. Although the clustering quality of this algorithm is good as it solves the problem faced by DStream by only maintaining outliers with the grid and makes it impossible for empty grids to affect the cluster quality as only micro clusters are considered as input in the offline phase. However, the cluster quality and performance speed decreases as the window size increases as the approximation gets coarser.

Graph-based clustering algorithms transform data into a graph representation [19]. Vertices of the graph is the data points to be clustered, and the edges are given weights based on the similarity between data points. Then, there are several approaches [20] [21] that exploits cluster in the data by building minimal spanning tree with various greedy algorithms such as Boruvka's algorithm [22], Prim's algorithm [23], Kruskal's algorithm [24], etc. Each one of them has its advantages and disadvantages based on the application. HDBSCAN [25], a representative algorithm has adopted the graph based clustering method. An undirected, weighted graph is built from the data, and a minimum spanning tree is built on top of the graph to exploit hierarchy of density clusters. The original algorithm adopted Prim's algorithm to build minimal spanning tree on top of the graph. HDBSCAN* [20] algorithm then accelerated the HDBSCAN by adopting much faster Dual Tree Boruvka algorithm to build minimal spanning tree which accelerated the overall performance of the algorithm. Furthermore, an important concept called the cluster stability was proposed in HDBSCAN which enables the algorithm exploit clusters of different density distribution from data without need of ε parameter which is a distance threshold. HASTREAM [21] is a stream clustering algorithm that adapts HDBSCAN as its offline-phase clustering algorithm. In the online-phase of this algorithm, it adapted the micro cluster structure.

HASTREAM, Dstream and Denstream all considers evolving nature of the data stream; thus adapted the damping window model for processing the data stream. Although damping window model has its advantages, it clearly cannot be a good fit for applications that require a focus on data for a given time interval without bias.

Chapter 4. Proposed Method

4.1. StreamHD Algorithm

Our proposed StreamHD algorithm adapted the online-offline-phase model that has shown its efficiency in overcoming temporal limitation problem in previous studies. Figure 6 shows the general concept of stream clustering algorithm in terms of online-offline-phase framework. Online phase of the algorithm maintains incoming data stream summarization statistic information with the micro cluster structure and the offline phase of the algorithm performs modified HDBSCAN on the currently active micro clusters upon clustering request from the user.

4.2.1. Online phase

Online phase of StreamHD uses the sliding window model with micro cluster structure to approximate and represent the summarization of data stream efficiently.

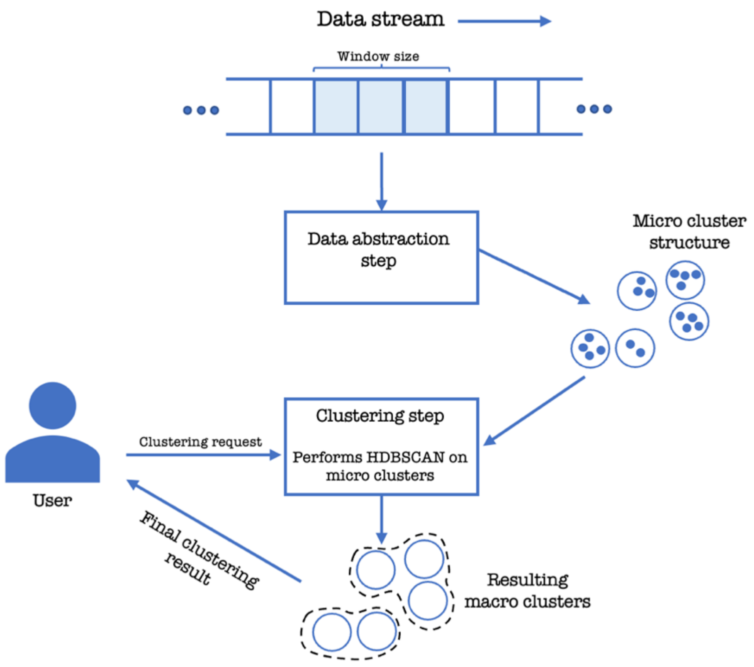


Figure 6. StreamHD algorithm overview

Unlike other algorithms that has user parameter ε ,which also is the value for the radius threshold, our algorithm does not have the ε user parameter. Therefore, to make sure that micro clusters are adapted well to the data stream concept drift, our method calculates micro cluster radius threshold periodically unlike the previous approaches with fixed micro cluster radius. Once every full window slide, micro cluster radius threshold is calculated as the average core-distance of the first 10% of the data points of the window.

2 types of micro clusters are maintained in the online phase of StreamHD. p-micro cluster which is a micro cluster that contains amount of data points surpassing the minPts threshold and o-micro cluster that does not contain enough data points to surpass the minPts threshold and maintains outlier data points.
As the window slides, incoming new data points might transform an outlier micro cluster into a p micro cluster and discarding data points

```
Algorithm 1 StreamHD(DS, MinPts, N)
  $t_c \leftarrow 1$
  while $DS$ is active do
      if $t_c mod N == 0$ then
          $\varepsilon = max(\{Coredist(p_{tc}...p_{tc+N/10})\})$
          Pruning()
          Try to merge data point $p$ into its closest $p-micro-cluster\ c_p$
          if $r_c p \leq \varepsilon$ then
              Merge p into $c_p$
          else Find closest $o-micro-cluster\ c_o$ from data point $p$
              Merge p tentatively into $c_o$
              if $r_{co} \leq \varepsilon$ then
                  Merge p into $c_o$
                  if $w \geq minPts$ then
                      Transform $c_o$ into $p-micro-cluster\ c_{pn}$
              else Merge p into new $o-micro-cluster\ c_{on}$

                  if Cluster request == True then
                      Pruning()
                      $\{Coredist\} \leftarrow ComputeCoreDistance(microclusters)$
                      $MSTree \leftarrow CalculateMSTree(\{Coredist\})$
                      $Result \leftarrow ExtractFlatResult(MST tree)$
                      return $Result$
```

Figure 7. StreamHD Algorithm

might result in p micro clusters transform into o micro cluster or existing o micro clusters might disappear due to the lack of data points. The disappeared micro clusters are periodically pruned in the online phase to save memory space and to keep the micro clusters up to date.

If a datapoint p fits into the micro cluster it expands the micro cluster. If the expanded micro cluster is p micro cluster, it simply updates the micro cluster statistics. If the o micro cluster is expanded, after updating the statistics of the o-micro cluster, if the weight surpasses the minPts threshold, it transforms into a p-micro cluster.

When the user requests triggers offline phase of the StreamHD, accelerated HDBSCAN produces clustering results using p-micro clusters as an input. The StreamHD algorithm is shown in Figure 7.

4.2.2. Offline phase

In the offline phase of the StreamHD, accelerated version of HDBSCAN - HDBSCAN* is performed on the centers of micro clusters that are being maintained at the time of the clustering request. When an offline clustering request came in, micro clusters are pruned to make sure that only the micro clusters containing the current window data points are passed to the offline stage. The macro clusters produced as a result of the HDBSCAN* is the final clustering that is returned to the user. Figure 8 shows the general concept of offline-phase of the StreamHD algorithm.

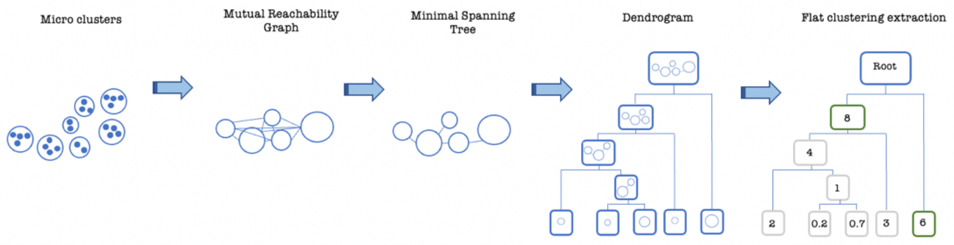


Figure 8. StreamHD offline-phase overview

HDBSCAN* is an accelerated version of the HDBSCAN algorithm. We modified the HDBSCAN* to produce macro-clusters from micro-clusters. Micro clusters are considered as a virtual data point with weight.

For a given fixed value k, (parameter minPts in our algorithm), HDBSCAN defined a new distance metric, called the mutual reachability distance.

For any micro cluster X_i, the core-distance of X_i, denoted $k(X_i)$ is defined to be the distance to the k^{th} nearest neighbor of X_i.

Then, given micro clusters Xi and Xj , and their respective core-distance, mutual reachability distance can be defined as the maximum of the Euclidean distance between the center of the micro clusters denoted as $d(Xi, Xj)$ and their respective core distances:

$$d_{mreach}(Xi, Xj) = \begin{cases} max(\kappa(Xi), \kappa(Xj), d(Xi, Xj)) & , Xi \neq Xj \\ 0 & , Xi = Xj \end{cases}$$

Then, mutual reachability graph can be built based on the micro clusters and mutual reachability distance. Mutual reachability graph *MRG (V, E, w)* is a connected undirected weighed graph. The set of vertices V is represented by the micro clusters available at the time of clustering result request. Then the Mutual reachability graph *MRG (V, E, w)* can be defined as follows.

$$MRG\ (V, E, w) = \begin{cases} V = \{MC\} \\ E = \{e(u, v) | u, v \in V\} \\ w(e) = d_{mreach}(u, v) \end{cases}$$

For the mutual reachability graph, a vertex v is considered directly reachable to the vertex u if there exists an edge between the vertices. Therefore, adjacency list of a vertex v contains all the vertices that are directly reachable from the vertex v.

$$AdjE\ (v) = \{\forall u \mid u \in V \ and \ e(v, u) \in E\}$$

Connected components of the Mutual Reachability Graph $MRG\ (V, E, w)$ are a set of vertices that each vertex has at least one adjacent vertex.

$$ConnectedComponent\ (V, E) = \{v \in V \mid AdjE(v) \neq \emptyset \wedge AdjE(v) \neq \{v\}\}$$

Minimum spanning tree (MST) is a connected subset of the edges of a connected undirected weighed graph. Building a MST from the $MRG\ (V, E, w)$ is a crucial step before extracting the clusters from the graph. MST is built one edge at a time, adding lowest weighing edge that connects the current tree to a vertex in the $MRG\ (V, E, w)$that is not in connected to the tree yet.

The extraction of the clusters can begin after building the MST. A dendrogram is built before extracting the cluster results. The root node of the dendrogram contains all the micro clusters. Starting from the complete MST, edges having largest weights are removed from the MST. By removing the edges, subcomponents of the currently considered cluster will be produced. Without consideration of the weight of the micro cluster and assume micro cluster as a virtual point, clusters discovered from single micro clusters that weighs more than the minimum cluster size threshold might get ignored. A subcomponent that produces a cluster can be either connected component of another micro-cluster, or can be a single micro-cluster. Single or connected components of a micro cluster, weight of a subcomponent must exceed the minPts threshold in order to be considered as a cluster at current hierarchical level. If the weight cannot exceed the minPts the density threshold, it cannot be considered as a possible cluster. When all the edges are removed, the hierarchical clustering of the micro clusters can be represented as a dendrogram.

The flat extraction of the clusters can be calculated w.r.t the cluster stability measure. Cluster stability (denoted as σ) of cluster C can be defined as

$$\sigma(C) = \sum_{mc \in C} w(mc) * \left(\lambda_{max}(mc, C) - \lambda_{min}(C)\right)$$
$$= \sum_{mc \in C} w(mc) * \left(\frac{1}{\varepsilon_{min}(mc, C)} - \frac{1}{\varepsilon_{max}(C)}\right)$$

where $w(mc)$ is the weight of the micro cluster mc, and $\lambda = \frac{1}{\varepsilon}$ is density threshold, $\lambda_{max}(mc, C)$ is the maximum density threshold value where the mc does not belong to cluster C anymore and $\lambda_{min}(C)$ is the minimum density threshold where the mc belong to cluster C. ε_{min} and ε_{max} are the ε thresholds that can be extracted for each cluster from the dendrogram.

Flat clustering is then the set of clusters extracted from the dendrogram that maximizes the sum of cluster stabilities of each cluster excluding the root node of the dendrogram. If the set of clusters is $\{C_1, C_2, C_3, \dots, C_n\}$, then we wish to select clusters that maximize the following value

$$\sum_{i \in I} \sigma(C)$$

with constraint that $\forall i, j \in I$ with $i \neq j$, $C_i \cap C_j = \emptyset$.

In Flat Clustering Extraction step in Figure 8, the nodes that are highlighted in green are the selected clusters as a result of flat clustering extraction. It can be seen that the sum of cluster stabilities of selected clusters is the largest.

Lastly, the final clustering result, which is the flat clustering extracted from the dendrogram is returned to the user.

Chapter 5. Experiments and Results

In evaluation, StreamHD is compared with two of the state-of-the art stream clustering algorithms.

DBStream is an stream clustering algorithm based on shared density graph of micro clusters. StreamSW is an approximation stream clustering algorithm based on micro cluster and grids. Both of the algorithms have shown good results in previous studies.

The parameters of each algorithms were tuned to achieve the highest cluster quality for each window size.

5.1. Experiment Environment

The experiments were conducted on a stand-alone machine with 2.90GHz Intel(R) Core i7 CPU with 32GB RAM and 500GB SSD disk running Ubuntu 20.01 LTS. Datasets were loaded on to the memory at the time of the experiment; thus, the disk did not have any influence on the experiment results. All the algorithms were implemented on Java with JDK 1.8.0 by us.

5.2. Datasets

5.2.1. Synthetic Datasets

To demonstrate unique clustering qualities of StreamHD, 2 synthetic datasets were used in evaluation.

Maze is a 2-dimensional uniform distributed synthetic dataset with label.

2D is also a 2-dimensional synthetic dataset we generated for this study that consists of 3 clusters that have a different underlying density distribution (Shown in Figure 9). In 2D dataset, HDBSCAN results were used as a label(Shown in Figure 10).

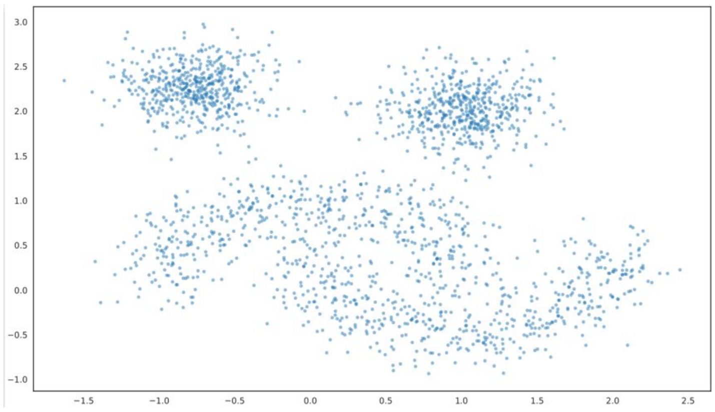


Figure 9. 2D synthetic data

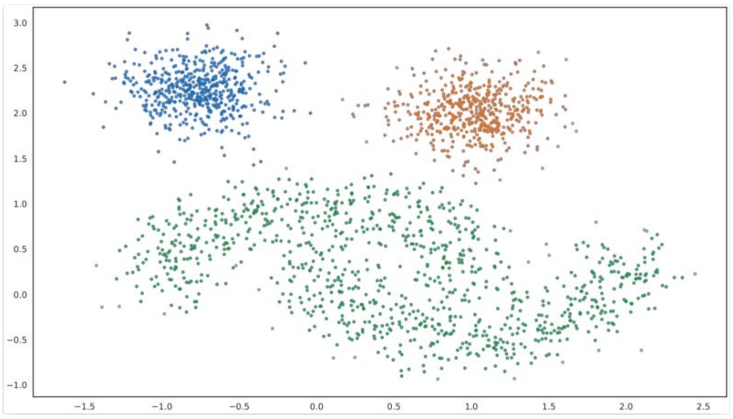


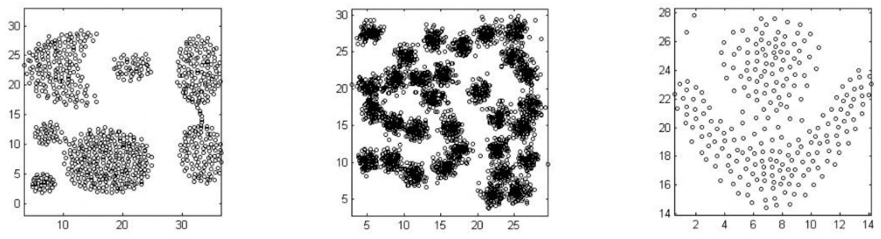
Figure 10. 2D synthetic data with HDBSCAN label

5.2.2. Real dataset

DTG [26] is a real dataset collected from the digital tachograph devices attached to commercial vehicles in a metropolitan city. Each record includes time, location, speed and the acceleration information of each vehicle and was generated every 10 seconds. The 2D coordinates of the location (latitude, longitude) were used in this experiment. The total number of records is approximately 300 million.

5.2.3. Static clustering datasets

To further examine the cluster qualities, we run the proposed algorithm on few of the common labeled static data clustering datasets Iris, Aggregation, D31, Flame, Spiral and Jain [27] [28] [29] [30] [31]. Some of the datasets were shown in Figure 11. Iris is a 4-dimensional dataset and the rest is all 2 dimensional.



(a) Aggregation dataset (b) D31 dataset (c) Flame dataset

Figure 11. Static clustering datasets.

5.3. Evaluation Method

The cluster quality of the algorithms were evaluated by the Adjusted Rand Index (ARI) [32] evaluation method.
ARI determines whether 2 cluster results are similar to each other and can be calculated as follows.

$$ARI = \frac{RI - expected(RI)}{\max(RI) - expected(RI)}$$

RI stands for the rand index that calculates the similarity between given 2 cluster results by considering all data points partitioned into the same cluster. The ARI value equals to 0 when all the data points are assigned into a different cluster in the 2 clustering results and 1 when the clustering results match.

5.4. Results

5.4.1. Cluster quality over Sliding window

StreamHD was compared to summarization-based stream clustering algorithms DBStream and StreamSW. First, we measured quality of clusters over sliding window. Figure 12 and 13 shows the ARI results of clustering algorithms over varying window size on MAZE and DTG datasets respectively. It can be seen from the figure that StreamHD shows the best cluster quality among the compared algorithms in varying window size. StreamSW showed the best results when the window size was smaller, but the cluster quality dropped significantly when the window size increases. Even though the StreamSW produces high-quality clusters, it suffers a poor cluster quality when handling a bigger size of data at once.

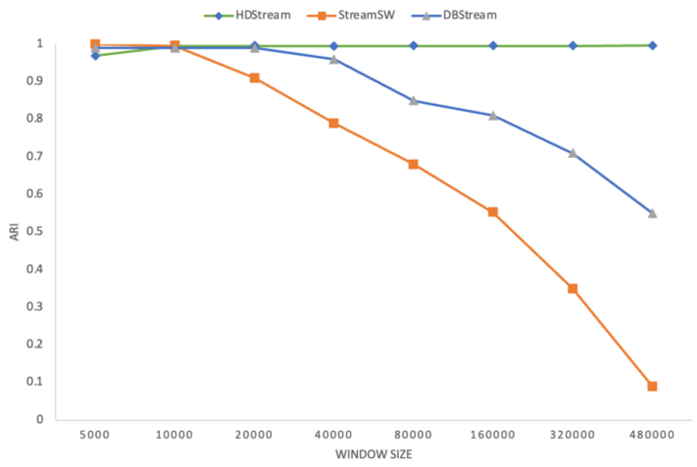


Figure 12. ARI with MAZE dataset over varying window size

Although DBStream produces relatively good quality clusters compared to the StreamSW, it still suffers scalability problem when it comes to handling a big data at once.

Both StreamSW and DBStream algorithm suffers one of the summarization-based clustering problems that when summarizing large amount of data, the cluster quality drops. It is because the summarized representation become coarser when the incoming data increases. StreamHD was able to overcome that problem because the underlying HDBSCAN algorithm is able to detect clusters of different densities; so that the coarse summarization did not affect the cluster quality.

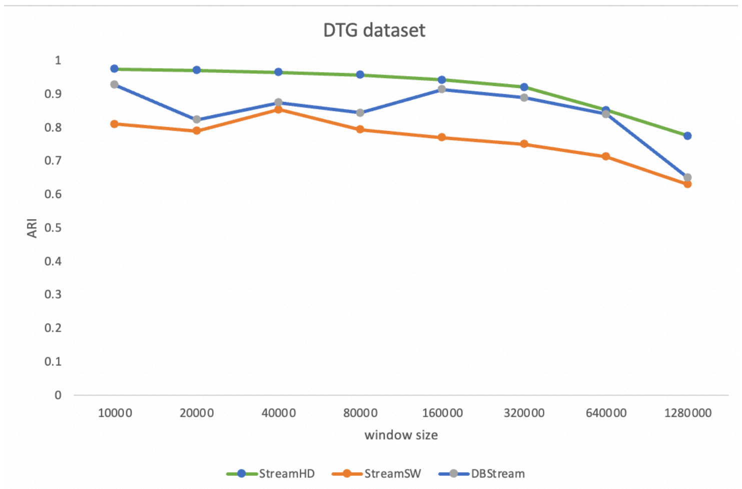


Figure 13. ARI with DTG dataset over varying window size

5.4.2. Latency (processing time) over Sliding window

As for the latency of the window for each algorithm, it was assumed that the clustering request from the user was came right after one full window slides, which means that online phase has maintained a full window of data points and then the offline phase started right at the end of the window. Our proposed algorithm StreamHD has shown the lowest latency on both the DTG and MAZE datasets as shown in Figure 14 and 15, respectively.

DBStream is the slowest among all the algorithms on MAZE dataset, and this is because it maintains the shared density among the micro clusters. Therefore, it can be said that there is a trade-off between processing time and cluster quality for the DBStream algorithm. It has performed slightly better on DTG dataset because the sparseness of the dataset. As opposed to the StreamSW, which maintains micro cluster and grid structure at the same time, it can be time consuming to find micro clusters and grid for a given data point, DBStream was able to overcome the overhead of sparsity with the shared density maintenance.

StreamHD was able to show lowest latency because even though it calculates a hierarchical tree from the micro clusters, the computation was faster due to the accelerated HDBSCAN algorithm that efficiently builds MST.

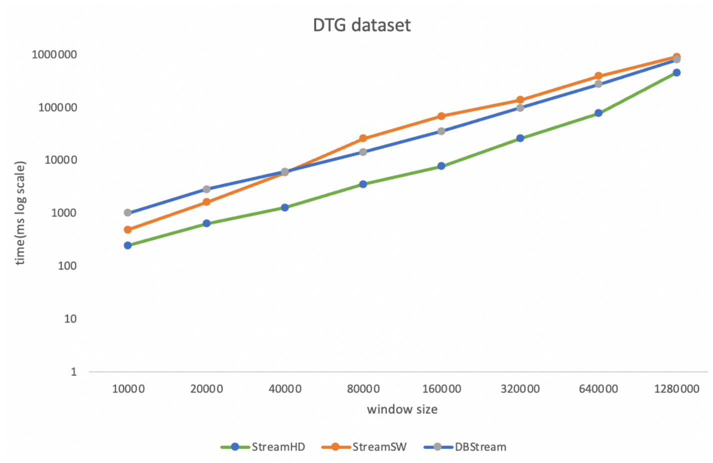


Figure 14. Latency with DTG dataset over varying window size

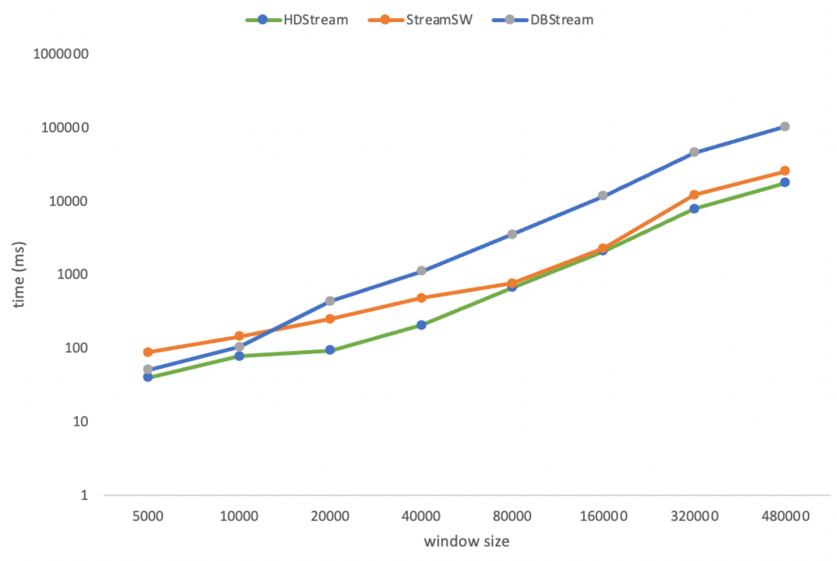


Figure 15. Latency with MAZE dataset over varying window size

5.4.3. Cluster quality over non-uniform density distribution

To better demonstrate the ability of StreamHD, we run the algorithms on a synthetic dataset with varying density profile. As shown in Figure 13, StreamHD has produced clusters with the best quality with ARI of 0.957 with respect to the HDBSCAN labels of 2D synthetic data shown in Figure 9.

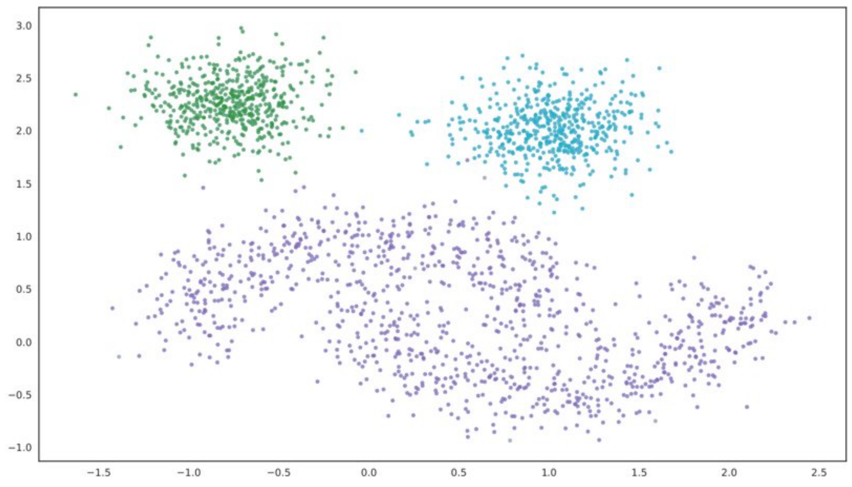


Figure 16. Clustering result of StreamHD on 2D synthetic data

DBStream has shown relatively good cluster quality as shown in Figure 14. Although the clustering results shown good quality, it achieved relatively low ARI of 0.561 because it discovered 4 clusters among the dataset which was supposed to be 3. Careful parameter tuning was performed when achieving this result and due to the non-uniform density profile of the synthetic data, it is almost impossible for DBStream to exploit the right labels from the data with fixed value of **ε**.

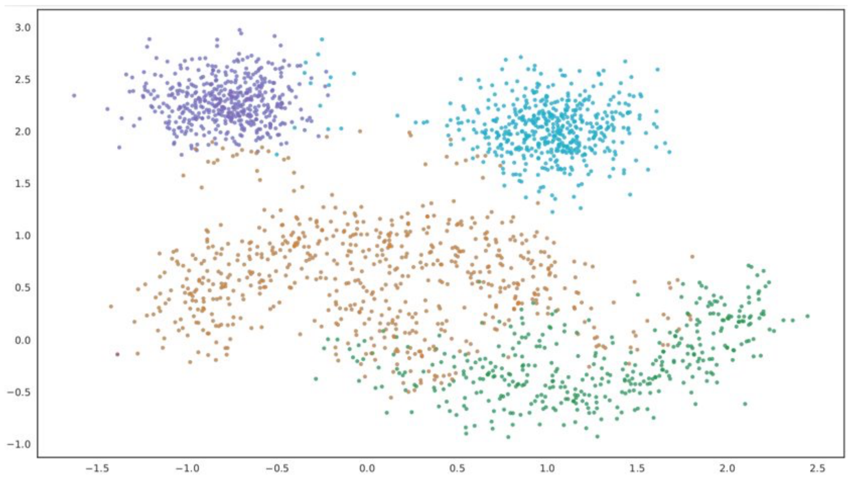


Figure 17. Clustering result of DBStream on 2D synthetic data

StreamSW has shown ARI of 0.698, which is higher than DBStream. However, as shown in the Figure 16, it only extracted 2 clusters from the dataset. Similar to the DBStream, StreamSW could not discover the same clusters as the label as it only considers a fixed value of **ε** when exploring clusters within the dataset. In addition, parameter modification has a crucial role in achieving the best result in StreamSW.

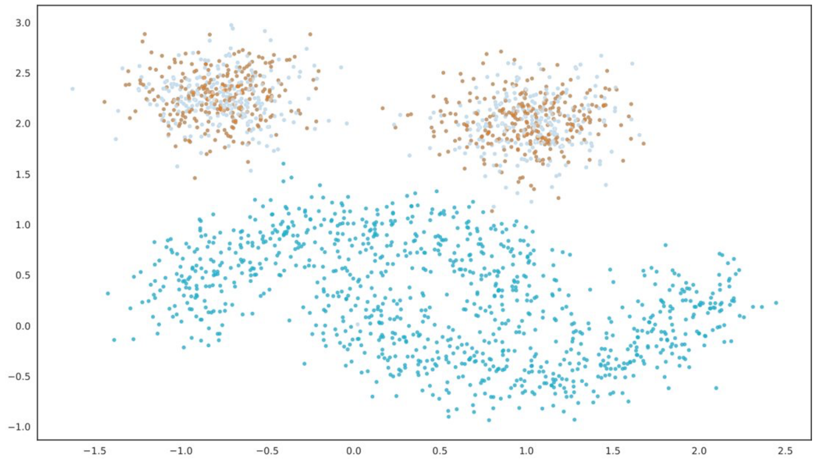


Figure 18.Clustering result of StreamSW on 2D synthetic data

5.4.4. Cluster quality of static datasets

Dataset	Dimension	# of attributes	StreamHD ARI
Aggregation	2	788	0.90
D31	2	3100	0.878
Spiral	2	312	0.938
Jain	2	373	0.94
Flame	2	240	0.938
Iris	4	150	0.82
Average ARI			**0.90**

Table 1. Clustering quality on various static datasets

To further evaluate StreamHD clustering quality, we used various static clustering datasets of varying dimensions with label. The first three columns show the name, dimension and number of attributes of the datasets. The fourth column shows the ARI value computed with respect to ground truth labels of the datasets. All the datasets were considered to exist in one window. It can be shown that even though there exists an approximation, our approach has shown relatively good ARI on all the datasets averaging at 0.90.

Chapter 6. Conclusion

In this paper, we proposed StreamHD, a stream clustering algorithm based on hierarchical density-based clustering over sliding window. It was shown that StreamHD can discover varying density distribution from the data with the least amount of user parameter; thus, leading to less user intervention to the model. Also, we proposed a heuristic that calculates the micro cluster radius threshold to be adaptable to concept drift corresponding to the varying density cluster exploration. Furthermore, when conducting a comparative experiment on synthetic and real-world dataset with some of the existing state-of-the-art algorithms that adapted same online-offline phase framework as our proposed algorithm, StreamHD has shown the best performance that produces the highest quality clusters with the lowest latency among the compared algorithms.

Bibliography

[1] A. Jain, "Data clustering: 50 years beyond k-means.," *Springer,* no. Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 3-4, 2008, September.

[2] Gong, S. Y. Zhang and Y. Ge, "Clustering stream data by exploring the evolution of density mountain," *Proceedings of VLDB Endowments,* vol. 11, no. 4, pp. 393-405, 2017.

[3] M. Esther, H. Kriegel, J. Sander and X. Xu, "Incremental clustering for mining in a data warehousing environment.," *In Proceedings of VLDB Conference,* pp. 323-333, 1998.

[4] C. Aggarval, S. Phillip, J. Han and J. Wang, "A framework for clustering evolving data streams," *In Proceedings of VLDB Conference,* pp. 81-92, 2003.

[5] M. Esther, H. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," *KDD,* vol. 96, no. 34, pp. 226-231, 1996.

[6] M. Carnein, D. Assenmacher and H. Trautmann, "An empirical comparison of stream clustering algorithms," *In Proceedings of the computing frontiers conference,* pp. 361-366, 2017.

[7] R. Patrigi and A. Ahmed, "Big data: The v's of the game changer paradigm.," *IEEE 18th International Conference on High Performance Computing and Communications,* pp. 17-24, 2016.

[8] I. Louhi, L. Boudjeloud-Assala and T. Tamisier, "Big Data Clustering using Data Streams Approach," *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies (BDAW '16),* no. Article 42, pp. 1-8, 2016.

[9] Y. Miyata and H. Ishikawa, "Concept drift detection on data stream for revising DBSCAN cluster," *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics,* pp. 104-110, 2020.

[10] C. Aggarval, J. Han, J. Wang and P. Yu, "On Clustering Massive Data Streams: A Summarization Paradigm," *Aggarwal C.C. (eds) Data Streams. Advances in Database Systems,* vol. 31, 2007.

[11] A. Amini and T. Wah, "Density micro-clustering algorithms on data streams: A review.," *In World Congress on Engineering,* vol. 2188, pp. 410-414, 2012.

[12] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: an efficient data clustering method for very large databases," *In Proceedings of the ACM*

SIGMOD International Conference on Management of Data, pp. 103-114, 1996.

[13] Y. Chen and L. Tu, "Stream data clustering based on grid density and attraction.," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 3, no. 3, pp. 1-27, 2009.

[14] B. Kim, K. Koo, J. Kim and B. Moon, "DISC: Density-Based Incremental Clustering by Striding over Streaming Data," *IEEE 37th International Conference on Data Engineering (ICDE),* pp. 828-839, 2021.

[15] F. Cao, M. Estert, W. Qian and A. Zhou, "Density-based clustering over an evolving data stream with noise," *In Proceedings of the SIAM International Conference on Data Mining,* pp. 328-339, 2006.

[16] J. Gan and Y. Tao, "Dynamic density based clustering.," *In Proceedings of the ACM International Conference on Management of Data,* pp. 1493-1507, 2017.

[17] J. Gan and Y. Tao, "On the hardness and approximation of Euclidean DBSCAN," *ACM Transactions on Database Systems (TODS),* vol. 42, no. 3, pp. 1-45, 2017.

[18] K. Reddy and C. Bindu, "StreamSW: A density-based approach for clustering data streams over sliding windows," *Measurement,* vol. 144, pp. 14-19, 2019.

[19] P. Foggia, G. Percannella, C. Sansone and M. Vento, "A graph-based clustering method and its applications," *In International Symposium on Brain, Vision, and Artificial Intelligence,* pp. 277-287, 2007.

[20] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," *IEEE International Conference on Data Mining Workshops (ICDMW),* pp. 33-42, 2017.

[21] M. Hassani, P. Spaus and T. Seidl, "Adaptive multiple-resolution stream clustering," *In International Workshop on Machine Learning and Data Mining in Pattern Recognition ,* pp. 134-148, 2017.

[22] V. King, "A simpler minimum spanning tree verification algorithm.," *Algorithmica,* vol. 18, no. 2, pp. 263-270, 1997.

[23] S. Gass and M. Fu, " Prim’s Algorithm," in *Encyclopedia of Operations Research and Management Science*, Boston,MA, Springer, 2013.

[24] S. Gass and M. Fu, "Kruskal’s Algorithm," in *Encyclopedia of Operations Research and Management Science*, Boston, MA, Springer, 2013.

[25] R. Campello, D. Moulavi and Sander.J, "Density-based clustering based on hierarchical density estimates," *In Pacific-Asia Conference on Knowledge Discovery and Data Mining,* pp. 160-172, 2013.

[26] "How to Use a Digital Tachograph," StoneRidge, [Online]. Available: https://www.optac.info/uk/digital-tachograph/.

[27] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics,* Vols. 7, Part II, pp. 179-188, 1936.

[28] A. Gionis, H. Mannila and P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 1, no. 1, pp. 1-30, 2007.

[29] H. Chang and D. Yeung, "Robust path-based spectral clustering," *Pattern Recognition,* vol. 41, no. 1, pp. 191-203, 2008.

[30] C. Veenman, M. Reinders and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 9, pp. 1273-1280, 2002.

[31] A. Jain and M. Law, "Data clustering: A user's dilemma," *Lecture Notes in Computer Science,* vol. 3776, pp. 1-10, 2005.

[32] J. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," *In International Conference on Artificial Neural Networks,* pp. 175-184, 2009.

초록

데이터 스트림은 응용 프로그램이 급격히 증가함에 따라 최근 몇 년 동안 뜨거운 관심 주제가 되었다. 또한, 전자 기기 및 네트워크의 과도한 사용으로 인해 데이터 스트림이 지속적으로 생성되고 있다. 따라서 데이터 스트림은 빠른 데이터 포인트 생성과 같은 정적 데이터와 다른 고유한 특성을 가지며 시간이 지남에 따라 무한한 크기에 도달할 수 있다.

위에서 언급한 데이터 스트림의 고유한 특성으로 인해 데이터 스트림 클러스터링 알고리즘에 대한 요구 사항은 점점 더 복잡해지고 있다. 정적 데이터에 대한 클러스터링 알고리즘의 기본 요구 사항은 데이터 내에서 임의의 모양과 클러스터 수를 추출할 수 있어야 한다. 또한 스트림 클러스터링 알고리즘은 시간과 공간의 제약으로 인해 들어오는 데이터를 빠르고 효율적으로 처리하는 것이 중요하다. 기존의 밀도 기반 알고리즘은 들어오는 스트림 내에서 임의의 모양과 숫자의 클러스터를 성공적으로 찾았지만 밀도(ε)가 클러스터당 최소 포인트(minPts)와 함께 사용자 고정 매개변수인 경우가 많기 때문에 밀도 변화를 발견할 때는 여전히 부족하다.

본 논문에서는 데이터 스트림 내에서 밀도가 다른 클러스터를 감지할 수 있는 계층적 밀도 기반 클러스터링 알고리즘을 기반으로 하는 StreamHD 라는 스트림 클러스터링 알고리즘을 제안한다. 제안된 알고리즘은 사용자 개입을 최소화하면서 클러스터의 밀도 임계 값을 독립적으로 감지합니다. 또한 StreamHD 는 코어 밀도를 계산할 때 고려하고 최소 클러스터 크기도 결정하는 주어진 지점의 인접 지점 수를 결정하는 minPts 와 윈도우 크기의 두 가지 사용자 매개변수만 필요하다. StreamHD 는 스트림 클러스터링 알고리즘 중 사용자 개입이 가장 적다고 할 수 있다. 또한 실제 데이터셋과 합성 데이터셋에 대한 실험 결과 우리가 제안한 알고리즘이 각 윈도우 처리 시간과 클러스터 품질 측면에서 비교 알고리즘 중 가장 우수한 성능을 보였다.

주요어: 데이터 스트림, 계층적 클러스터링, 밀도 기반 클러스터링, 슬라이딩 윈도우
학번: 2020-20623

Acknowledgement

Throughout the writing of this thesis, I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Bongki Moon, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. I am extremely grateful that you took zme on as a student and continued to have faith in me over the past two years.

I would like to acknowledge my colleagues from Database Systems Lab for their wonderful collaboration. I want to thank you for your support and for all of the opportunities I was given to further my research.

I also would like to thank my mom, dad, brother and sister for their unconditional love and support throughout all these years. I especially want to thank my parents for encouraging me to further my studies.

I could not have completed this thesis work without the support of my partner, who provided enormous emotional support and happy distractions to rest my mind outside of my research.

Finally, I want to thank Seoul National University GSFS Scholarship for funding me throughout my Master's studies.

This accomplishment would not have been possible without every one of you and I will forever carry on this gratitude towards you all.