DEVELOPMENT OF DENSITY BASED CLUSTERING ALGORITHMS FOR STREAMING DATA

Mr. Sirisup Laohakiat

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science and
Information Technology
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2016

การพัฒนาขั้นตอนวิธีจัดกลุ่มบนพื้นฐานความหนาแน่นสำหรับข้อมูลที่มีการไหลเข้าอย่างต่อเนื่อง

นายศิริสรรพ เหล่าหะเกียรติ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2559

Thesis Title	DEVELOPMENT OF DENSITY BASED CLUSTERING ALGORITHMS FOR STREAMING DATA
By	Mr. Sirisup Laohakiat
Field of Study	Computer Science and Information Technology
Thesis Advisor	Professor Chidchanok Lursinsap, Ph.D.
Thesis Co-Advisor	Assistant Professor Suphakant Phimoltares, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

.....Dean of the Faculty of Science
(Associate Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

.....Chairman
(Professor Thanaruk Theeramunkong, Ph.D.)

.....Thesis Advisor
(Professor Chidchanok Lursinsap, Ph.D.)

.....Thesis Co-Advisor
(Assistant Professor Suphakant Phimoltares, Ph.D.)

.....Examiner
(Assistant Professor Saranya Maneeroj, Ph.D.)

.....Examiner
(Associate Professor Chatchawit Aporntewan, Ph.D.)

.....External Examiner
(Associate Professor Ekkarat Boonchieng, Ph.D.)

ศิริสรรพ เหล่าหะเกียรติ : การพัฒนาขั้นตอนวิธีจัดกลุ่มบนพื้นฐานความหนาแน่นสำหรับข้อมูลที่มีการไหลเข้าอย่างต่อเนื่อง (DEVELOPMENT OF DENSITY BASED CLUSTERING ALGORITHMS FOR STREAMING DATA) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ศ. ดร. ชิดชนก เหลือสินทรัพย์, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม: ผศ. ดร. ศุภกานต์ พิมลธเรศ, 81 หน้า.

ข้อมูลที่มีการไหลเข้าอย่างต่อเนื่องได้มีบทบาทมากขึ้นในการทำเหมืองข้อมูล ทั้งนี้ข้อมูลชนิดนี้แตกต่างจาก ชุดข้อมูลทั่วไปตรงที่ ข้อมูลที่มีการไหลเข้าอย่างต่อเนื่อง จะค่อยๆสะสมตามเวลา มิใช่มีข้อมูลทั้งชุดที่สมบูรณ์เลย ตั้งแต่ต้น อีกทั้งข้อมูลที่มีการไหลเข้าอย่างต่อเนื่องมักมีขนาดใหญ่ ด้วยลักษณะเฉพาะเหล่านี้ ทำให้ต้องออกแบบขั้นตอนวิธีจัดกลุ่มแบบใหม่ขึ้น เพื่อใช้กับข้อมูลชนิดนี้ วิทยานิพนธ์ฉบับนี้ นำเสนอขั้นตอนวิธีจัดกลุ่มสำหรับข้อมูลที่มีการไหลเข้าอย่างต่อเนื่องบนพื้นฐานความหนาแน่น โดยขั้นตอนวิธีจัดกลุ่มที่นำเสนอนี้ ได้นำมาทดสอบกับขั้นตอนวิธีจัดกลุ่มอื่นๆที่ได้รับการใช้งานอย่างแพร่หลาย เพื่อแสดงให้เห็นถึงประสิทธิภาพของขั้นตอนวิธีจัดกลุ่มที่นำเสนอนี้

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	.....
		ลายมือชื่อ อ.ที่ปรึกษาหลัก	.....
สาขาวิชา	วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ	ลายมือชื่อ อ.ที่ปรึกษาร่วม	.....
ปีการศึกษา	2559		

5673104623 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS: DENSITY-BASED CLUSTERING / HYPER-CYLINDRICAL MICRO-CLUSTER / STREAMING DATA / UNSCHEDULED DATA REMOVAL / DIMENSION REDUCTION / LINEAR DISCRIMINANT ANALYSIS

SIRISUP LAOHAKIAT: DEVELOPMENT OF DENSITY BASED CLUSTERING ALGORITHMS FOR STREAMING DATA. ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., CO-ADVISOR: ASST. PROF. SUPHAKANT PHIMOLTARES, Ph.D., 81 pp.

Streaming data has played important role in many data mining applications. Different from traditional data sets which the whole data records are available at the beginning, streaming data accumulate over time, and usually due to its continuous flow of data records, the volume of this kind of data set is usually large. Several algorithms for clustering streaming data have been designed in accordance with these restrictions. In this study, some further constraints on the characteristics of the data set are considered; in order to design density based clustering algorithms which can cluster these data sets efficiently. The designed algorithms have been tested against some state-of-the-art algorithms to determine the effectiveness of the proposed algorithms.

Department:	Mathematics and Computer Science	Student's Signature .....
		Advisor's Signature .....
Field of Study:	Computer Science and Information Technology	Co-Advisor's Signature .....
Academic Year:	2016	

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Professor Chidchanok Lursinsap for the continuous support and guidance though out the course of my study. I would also like to thank my co-advisor Associate Professor Suphakant Phimoltares for the helpful suggestion on the research. Also, the dissertation committee: Professor Thanaruk Theeramunkong, Associate Professor Ekkarat Boonchieng, Associate Professor Chatchawit Aporntewan, and Assistant Professor Saranya Maneeroj whose useful suggestions and penetrating analysis help shape this study.

Finally, I would like to thank the Thailand Research Fund (TRF) for financial support of this research under the Royal Jubilee (RGJ) Scholarship.

CONTENTS

Page

Page

List of Figures

List of Tables

Chapter 1
INTRODUCTION

Nowadays, the applications of streaming data are found in various domains of data acquisition systems, such as real-time monitoring, web site analysis, and electronic business. Researches on mining streaming data have been widely conducted in the recent years. Among researches regarding mining streaming data, clustering algorithms for steaming data have been one of the topics attracting much interest from researchers in the past ten years [1] [2].

Algorithms for clustering streaming data differ from the traditional clustering algorithms in two respects. First, streaming data are generated continuously over a relatively long period of time. The volume of data set is so vast that it usually exceeds the size of the main memory. As a result, these algorithms must store data in some mathematical and statistical forms rather than the whole raw data to make it possible to practically cluster the data. Second, since streaming data are often found in real-time monitoring systems, online clustering algorithms that can process the past and present data promptly are more preferable.

Clustering algorithms for streaming data extend clustering approaches based on well-known clustering algorithms for traditional data sets, including K-mean clustering and density based clustering. The requirement of K-mean clustering algorithms for the number of clusters limits the potential of this algorithm in many applications which the number of clusters is unknown. On the other hand, density based clustering algorithm does not require the knowledge of the number of clusters, resulting in more flexibility in clustering. Due to this advantage, the designed algorithms are based on density based clustering in this study.

Existing algorithms which adopt density based clustering, such as DenStream [5], deal with the requirements of streaming data by using online-and-offline-phases clustering scheme,. The online component maps each incoming data sample into a spherical micro-cluster, whose process called micro-clustering. Data assigned to micro-clusters are discarded from the system to save memory storage. Keeping the

statistics of data inside, including mean, variance, and the number of data, each micro-cluster represents a local cluster of data. To determine the final clusters, the offline component derives each cluster from a set of overlapping micro-clusters. As a result, the shape of each final cluster can be arbitrary because it consists of a set of several connected small spheres. Moreover, the clustering process can be performed without the restriction over the number of clusters.

Extending idea of micro-clustering in density based clustering, this study imposes various constraints in accordance with data sets found in real application. Algorithms for clustering streaming data under the imposed constrains are designed and tested.

1.1 Objectives

The main objectives of this study are as follows:

- To develop a new density based clustering algorithm for streaming data in a high dimensional space with more efficient usage of main memory.
- To develop a new mathematical object for clustering the streaming data.

1.2 Problem statement

Given a stream data set in which data instances arrive continually, how data instances are grouped into clusters using density based clustering method efficiently. For the first algorithm, we focus on minimizing memory storage used during the computation. The second algorithm is designed to account for streaming data sets with high dimensionality.

1.3 Contribution

This dissertation proposes two density based clustering algorithms for streaming data. The first algorithm, named as HCMstream, has the following highlighted features:

First, we propose the use of a new mathematical object, hyper-cylindrical micro-clusters, as local cluster for grouping data instance. This shape has more

compactness than the traditional spherical micro-clusters. Second, the algorithm allows the user to removal each data record at any arbitrary time while the existing algorithms require all data records to have the same pre-specified lifetime. Finally, the clustering process is performed online phase without the need of offline phase as in other existing algorithms.

For the second algorithm, named as LLDstream, the novel features includes the following aspects. First, we integrate dimension reduction technique on LDA subspace into a density based clustering algorithm for streaming data. This technique allows more flexibility than the conventional methods which use feature selection technique based on the variance of each feature. Moreover, we propose unsupervised localized linear discriminant analysis. This technique allows us to use LDA projection technique with unlabeled data sets which traditional LDA method is not applicable to this kind of data sets.

1.4 Scopes of work

In this dissertation, the scope of work is constrained as follows:

- Data instances are assumed but not restricted to appear one instance at a time. If data instances come in batch, the algorithms process one data instance at a time.
- Each feature of streaming data set is either numerical or binary-value categorical.

The probability distribution of data is unknown.

1.5 Dissertation outline

The rest of this dissertation is organized as follow. Chapter 2 describes the related backgrounds. Chapter 3 provides the detail of HCMstream and the experimental result. Chapter 4 presents LLDstream and its experimental performances. Chapter 5 concludes the study.

Chapter 2
LITERATURE REVIEWS AND BACKGROUND

In this chapter, the background of clustering streaming data sets is presented. Existing state-of-the-art algorithms are reviewed and discussed. Then, some background concepts used in this study are presented.

2.1 Literature reviews

The existing algorithms for clustering streaming data can be divided into two groups based on the assumption of the nature of clusters. In the first group, the clusters are assumed to be static. Each datum has no expiration date. The topology of data distribution is fixed as long as the clustering process is concerned. All past clustered data and the new incoming data have the same contribution to cluster formation. For the second group, the aim of algorithms is to reveal the pattern of clusters for the most recently existing data. As a result, the more recent data have more contribution to cluster formation, whereas the influence of the past data would be deteriorated.

For algorithms concerning only static clusters, there were several interesting works in this aspect, some of which are based on K-means clustering and its derivatives. For streaming data, K-means algorithm have been adopted in several algorithms as follow. Guha et al. [3] proposed STREAM which used the one-pass k-median algorithm in a divide-and-conquer fashion to cluster stream data. They also proposed a facility location algorithm for relaxing the number of clusters during the intermediate steps to reduce the running time as well as to increase the stability of the algorithm.

Another algorithms are based on density-based clustering such as CompactStream [4]. Instead of using conventional spherical micro-clusters, it used elliptical micro-clusters in primary clustering. Tu et al. [5] extended hierarchical agglomerative clustering (HAC) to work with streaming data.

For algorithms tracking cluster evolvement, Clustream [6] was among the earliest algorithms. The clustering process is divided into two phases, namely online and offline component. In the online phase, data points are primarily clustered into micro-clusters which store the statistics of the data points inside. In offline phase, micro-clusters are used as representative points for k-means clustering to produce the predefined number of clusters which is the parameter provided by users.

Two recently published papers proposed a more flexible assumption regarding the shape of clusters by using ellipsoidal shape instead of spherical shape as in Clustream. HECES [7] uses grid-cells to calculate the statistical summary of streaming data. The grid-cells are replaced by a hyper-ellipsoidal shape from the covariance of grid-cells. Finally, overlapping ellipsoids are merged to form the final clusters. Lughofer et al. [8] also proposed evolving vector quantization (eVQ) which similarly uses ellipsoidal clusters. It includes incremental split-and-merge techniques to merge overlapping clusters together and split a cluster into disjoint clusters. Although these two algorithms allow more flexible shape of clusters, they still do not allow any arbitrary shape cluster.

DenStream [9] is a density-based clustering algorithm based on DBSCAN [10] with no constraint on the shape of cluster. In contrast to Clustream, DenStream does not require the number of clusters as its input parameter. Rather it uses the difference in the density of data points to distinguish clusters. Similar to Clustream, it adopts the online-offline two phases clustering scheme. In online phase, the primary clustering with micro-cluster is performed as in the case of Clustream. The micro-clusters are categorized into two groups as dense and sparse micro-clusters, called p-micro-clusters and o-micro-clusters, respectively.

In offline phase, differing from Clustream which uses K-mean clustering, DenStream uses the clustering process of DBSCAN. Intersected p-micro-clusters are grouped together to form a new cluster while o-micro-clusters are excluded from the clustering process. Denstream uses fading window model where micro-clusters not updated for some period of time would be gradually faded away. Zhou et al. [11] proposed SWClustering algorithm which adopted the clustering process of

DenStream. Unlike DenStream which uses fading scheme, it uses the exponential histogram of cluster features for dealing with evolving clusters.

D-Stream [12] is a grid-based clustering algorithm. Instead of using micro-clusters in online phase, data points are assigned into fixed grids in the feature space. The clustering process is essentially similar to DenStream. It groups dense consecutive grids into a cluster. Tu et al. [12] improved D-Stream by introducing the concept of attraction of grids which provides the information of how data points distribute inside a grid to refine the boundary of the clusters. Like D-Stream, ExCC [13] is a grid-based algorithm. It can process mixed attribute data as well as data with both numerical and categorical features. The clustering process is also divided into two phases as its predecessor. Dense cells are determined by a threshold computed adaptively from the number of total points in the cells and the size of each cell.

HPStream [14] is proposed to deal with high-dimensional data. The algorithm uses the subset of particular dimensions varying over different clusters to calculate the projected distance in clustering process.

While the aforementioned algorithms required two-phase clustering process, FlockStream [15] and MR-Stream [16], adopted a single phase clustering. The clustering results can be drawn promptly without the need to perform clustering in offline phase. Considering micro-clusters and incoming data points as agents in flocking model, FlockStream generates clusters based on the swarms formed by agents. It does not requires the exhaustive search of the nearest neighbor of a point when assigning micro-cluster to an incoming points. However, this benefit is traded off by the computation required in virtual space to simulate the movement of the agents. MR-Stream partitions feature space into grids similar to the process of D-Stream. Then, it employs tree structure to allow clustering with multiple resolutions.

Other algorithms employed different clustering methods. For example, SVStream [17] extended support vector machine clustering method to deal with stream data. RepStream [18] adopted graph-based clustering method with stream data.

The idea of removing data records can be traced back to IncrementalDBSCAN [19] designed for mining a large collection of data from different sources. Unlike

algorithms for streaming data, this algorithm can access to the whole data set at any time. The clustering process is based on DBSCAN with the ability to incrementally process insertion and deletion of new data records. By virtue of storing the whole data set, the cluster of a removed data record as well as its neighboring data points can be determined. As a result, the incremental update of the possessive cluster can be performed accordingly.

2.2 Clustering algorithms

The aim of clustering algorithms is to group data instances into clusters. Data instances which are closed together are grouped into the same cluster while data instances that are far from each other are assigned into different clusters. From Figure 1, we have data points shown in blue. From inspection, we can see that these data points consist of two clusters. We want to group these data points into clusters.

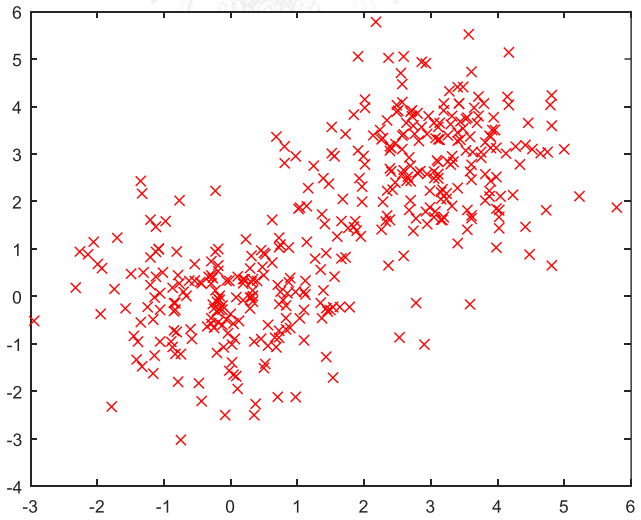


Figure 1 Example of data points in two dimensional data sets.

The resulting clusters are shown in Figure 2 where the members of one cluster are depicted in black while those of another cluster are depicted in red. The aim of clustering algorithm is to classify data points into group as shown in Figure 2.

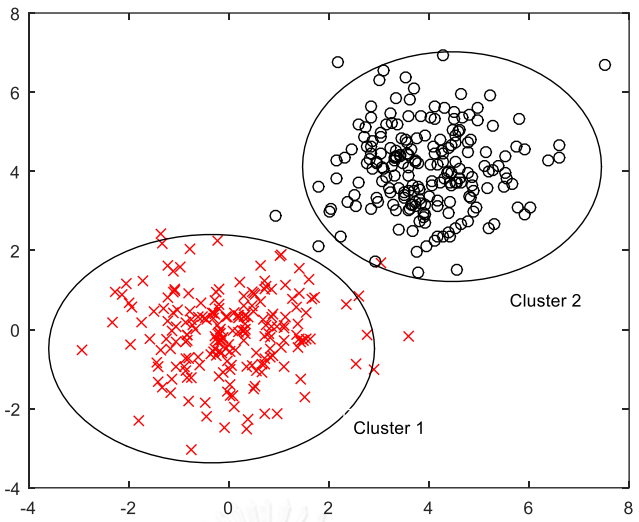


Figure 2 Example of clustering the data points in Figure 1.

2.3 Density based clustering algorithm

Density based clustering is one of the highly used clustering methods. From the pioneer algorithm-DBSCAN , several derivatives of this algorithm have been proposed and studied for these recent years. The brief idea of DBSCAN is presented as follow.

DBSCAN uses the concept of density connectivity to form clusters. The algorithm requires two parameter Eps and MinPts. Eps neighborhood of point p is defined as data points whose distance from p is less than Eps. Two points p and q are connected when 1) q is in the Eps neighborhood of point p and p is in the Eps neighborhood of point q and 2) Both Eps neighborhood of point p and q are more than MinPts. Each cluster is formed from a set of connected points.

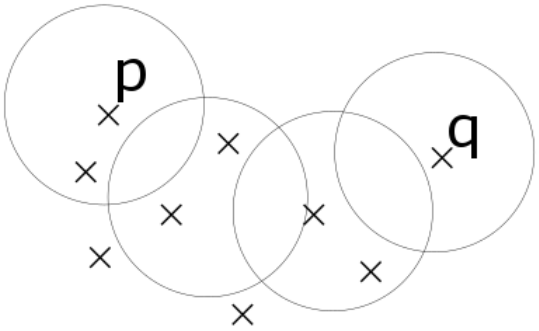


Figure 3 Operations of DBSCAN.

2.4 Definition of stream data

Let a data stream consists of a set of data instances $X = \{\mathbf{x_1}, \ldots, \mathbf{x}_i, \ldots\}$, where $\mathbf{x}_i$ indicates a data instance arriving at time stamp t_i. Each data instance $\mathbf{x}_i$ is a d-dimensional records, namely $\mathbf{x}_i = \left(x_i^1, \ldots, x_i^d\right)$.

Unlike traditional data set, stream data accumulates over time. The data set can grow indefinitely. In many cases, the whole data set might not be available at the beginning. Data instances can come individually or come in batch. We do not pose any constraint on the arrival of time t_i for each instance $\mathbf{x}_i$.

The set of stream data can grow indefinitely. As a result, traditional clustering algorithms which require the whole data set being available at the beginning cannot be applied to this kind of data set. Clustering algorithm for stream data need some features to deal with this kind of data set.

2.5 Algorithms for clustering stream data

As stream data can appear continually and indefinitely, the main storage might not be sufficient to store the whole data set. Moreover, since most of stream data are generated in monitoring tasks, real time processing is preferable for these data sets. In this section, some examples of well-known algorithms for stream data are presented and discussed.

2.5.1 DenStream

DenStream is an algorithms for clustering stream data extending the idea of density based clustering from DBSCAN. DenStream [9] is the density-based two-phase clustering scheme is adopted in this study. The so called two phase consists of online and offline clustering. Online clustering is performed online whenever a data instance arrive. Offline phase occurs when the user request the final clustering result. The algorithm would report the final cluster to the user. The outline of online clustering process of DenStream is shown in Figure 4.

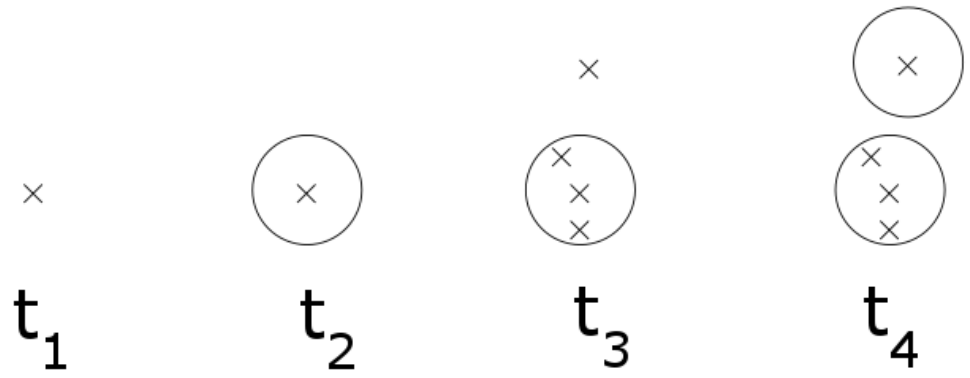


Figure 4 Online phase of DenStream. The figure shows four time stamps t_1, t_2, t_3, and t_4 where new data points occur in the feature space. DenStream would generate micro-clusters to cover all data points.

At time t_1, the incoming datum is shown in red. DenStream generates a micro-cluster to cover this datum as shown in t_2. If the subsequent new data appear inside the micro-cluster new micro-cluster would not be generated. At times t_3, a new datum occur outside the existing micro-cluster, we generate a new micro-cluster to cover this new datum as shown in t_4.

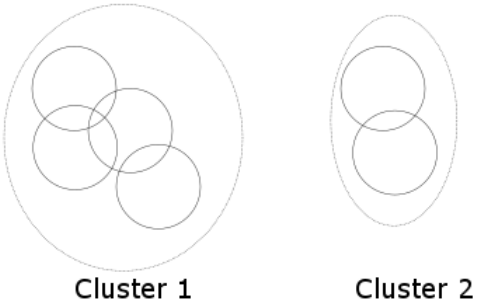


Figure 5 Offline phase of DenStream. DenStream define each overlapping micro-clusters as each cluster. Cluster 1 consists of four overlapping micro-clusters while Cluster 2 consists of two micro-clusters.

Figure 5 shows the operation of DenStream in offline phase. When the user requests the final clusters, DenStream forms clusters from sets of connected micro-clusters. From Figure, four micro-clusters are connected forming Cluster1 and two micro-clusters connected, forming Cluster2.

In DenSteam, the user is required to provide the upper bound of the radius designated as the parameter ε. Each micro-cluster mc is defined by $\{CF1, CF2, w\}$ where w is the weight of the data points assigned to mc, $CF1$ and $CF2$ are the linear sum and the square sum of the data points assigned to mc respectively. We find that for high dimensional data sets, this definition of micro-cluster has to store both $CF1$ and $CF2$ which requires two times more storage than that of LLDstream which stores only the linear sum of the data points inside. The radius of micro-cluster mc is calculated by $r = \sqrt{\sum_{j=1}^{D}\left(\frac{CF2_j}{w} - \left(\frac{CF1_j}{w}\right)^2\right)}$ where $CF1_j$ and $CF2_j$ are the linear sum and the square sum of mc on attribute j and D is the number of attributes of the data. In the online phase, an incoming datum $\mathbf{x}$ is assigned to its

nearest micro-cluster mc when after adding $\mathbf{x}$ into mc, the radius of mc is less than ε. In the offline phase, the algorithm adopts the variant of DBSCAN to cluster micro-clusters by using the centers of each micro-clusters as data points in DBSCAN. Micro-clusters with weight higher than the density threshold μ are used to generate the final clusters. Two micro-clusters mc_p and mc_q are connected when $D(\mathbf{c}_p, \mathbf{c}_q) \leq r_p + r_q$ where $\mathbf{c}_p$ and $\mathbf{c}_q$ are the centers of mc_p and mc_q respectively, $D(\mathbf{c}_p, \mathbf{c}_q)$is the distance between $\mathbf{c}_p$ and $\mathbf{c}_q$. The algorithm requires the radius threshold ε and the density threshold $\beta\mu$ as an input.

2.6 Linear discriminant analysis (LDA)

The objective of LDA is to find a set of projection vectors whose directions maximize class separability. Let the between-class scatter matrix $\mathbf{S_b}$ and the with-in class scatter matrix $\mathbf{S}_w$ be defined as [20]

$$\mathbf{S}_b = \frac{1}{M}\sum_{j=1}^{C} N^{(j)}(m^{(j)} - m_o)(m^{(j)} - m_o)^T$$

$$\mathbf{S}_w = \frac{1}{M}\sum_{j=1}^{C}\sum_{i=1}^{N^{(j)}}(x_i^{(j)} - m^{(j)})(x_i^{(j)} - m^{(j)})^T$$

where $x_i^{(j)}$ is the data instance i in class j, $m^{(j)}$ is the mean of data in class j, m_o is the global mean of the whole data set, C is the number of classes, $N^{(j)}$ is the number of data in class j, and $M = \sum_{j=1}^{C} N^{(j)}$ is the total number of data. However, if we consider all classes equally important, i.e., we want to determine the projected vectors that maximally separate all classes apart without concern over the number of data in each class, the coefficients $N^{(j)}$ in determining $\mathbf{S_b}$ can be dropped. Also, in determining the direction of the projected vectors, the term $\frac{1}{M}$ can be dropped, leading to the equation becoming

$$\mathbf{S}_b = \sum_{j=1}^{C}(m^{(j)} - m_o)(m^{(j)} - m_o)^T$$

$$\mathbf{S}_w = \sum_{j=1}^{C}\sum_{i=1}^{N^{(j)}}(x_i^{(j)} - m^{(j)})(x_i^{(j)} - m^{(j)})^T$$

In classical LDA, the projection matrix $\mathbf{V}$ is determined by maximizing Fisher criterion $J(\mathbf{V}) = trace\left(\left(\mathbf{V}^T\mathbf{S}_w\mathbf{V}\right)^{-1}\left(\mathbf{V}^T\mathbf{S}_b\mathbf{V}\right)\right)$ subject to the orthogonality constraint of $\mathbf{V}$. This optimization problem can be solved easily by finding the eigenvectors of $\mathbf{S}_w^{-1}\mathbf{S}_b$

2.7 Performance Indices

In comparing the performances among the compared algorithms, performance indices with respect to class labels are adopted. The clustering results were compared with the class labels provided in the data sets.

In this study, normalized mutual information matric (NMI) [21], Rand index (RI) [21, 22], Adjusted Rand index (AR) [23] and Hubert's index (HI) [23] were chosen as the performance indices. These indices consider both the homogeneity of clusters as well as the number of clusters obtained from the clustering algorithms. The values of these indices generally range from 0 to 1, excepting AR and HI which can be negative value. The larger value indicates that the obtained clusters are more similar to the class label.

Let Ω be the set of clusters obtained from the clustering algorithm and $\mathbb{C}$ be the set of clusters obtained from the class label. The mutual information of these two sets $I(\Omega, \mathbb{C})$ can be calculated by

$$I(\Omega, \mathbb{C}) = \sum_k \sum_j p(w_k \cap c_j) \log\left(\frac{p(w_k \cap c_j)}{p(w_k)p(c_j)}\right)$$

where $p(w_k)$, $p(c_j)$, and $p(w_k \cap c_j)$ are probabilities of a datum inside cluster w_k, class c_j, and the intersection between cluster w_k and class c_j, respectively;.NMI is obtained from

$$NMI = \frac{2I(\Omega, \mathbb{C})}{H(\Omega)H(\mathbb{C})}$$

where $H(\Omega) = -\sum_k p(w_k)\log(p(w_k))$ and $H(\mathbb{C}) = -\sum_k p(c_j)\log\left(p(c_j)\right)$. Rand index views clustering as a series of decisions in choosing $\binom{N}{2}$ pairs of data in the set.

Let a true positive (TP) be a decision of assigning two data from the same class into the same cluster, a true negative (TN) be a decision of assigning two data from different classes into different clusters, a false positive (FP) be a decision of assigning two data from different classes into the same cluster and a false negative (FN) be a decision of assigning two data from the same class into different clusters. Rand index can be calculated by

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

Adjusted Rand index is the derivative of Rand index derived from $\frac{RI - E[RI]}{\max(RI) - E[RI]}$, where $E[RI]$ is the expected value of RI, and $\max(RI)$ is its maximum value. AR is calculated by

$$AR = \frac{\sum_k \sum_j \binom{|w_k \cap c_j|}{2} - S_3}{\frac{S_1 + S_2}{2} - S_3}$$

where $S_1 = \binom{|w_k|}{2}$, $S_2 = \binom{|c_j|}{2}$, and $S_3 = \frac{2S_1 S_2}{N(N-1)}$.

Hubert's index is defined by the difference of the probability of agreement and the probability of disagreement. It is calculated as

$$HI = \frac{(TP + TN) - (FP + FN)}{TP + TN + FP + FN}$$

NMI uses the concept of mutual information in information theory to reflect the similarity between the resulting clusters and the class labels. The similarity is captured in terms of joint probability density of data items in the two comparing sets of clusters. The dependence between the two sets reflects their similarity. More value of NMI indicates more dependence of Ω and $\mathbb{C}$, and thus more similarity. On the other hand, the other indices, namely RI, AR, and HI capture the similarity based on counting the agreement between all pairs of data items. Rand index straightforwardly counts the number of agreeing pairs while Hubert's index includes

the penalty of disagreeing pairs in the calculation by subtracting disagreeing pairs $(FP + FN)$ from the agreeing pairs $(TP + TN)$. As a result, HI is always less than RI. Since RI usually has value between 0.5 to 1, AR is proposed with the inclusion of correction for chance in the calculation. By subtracting RI with its expectation by assuming the hypergeometric distribution, where items are drawn randomly from clusters without replacement, AR equals 0 when RI equals its expected value, indicating that the clustering result is not better than that from the random process.

Chapter 3
Proposed algorithm 1: HCMstream

In this dissertation, two algorithms for clustering stream data are proposed. The first algorithm, named HCMstream [24], is designed to improve two respects of the existing algorithms.

First, existing clustering algorithms for stream data such as DenStream, DStream, and CluStream require two phase operations, that are online and offline phases. When operating in offline phase, the algorithm cannot process the incoming data in real time, thus some memory buffer is required to solve this problem. On the other hand, an algorithm that does not require offline phase can operate without requirement for buffer memory, leading to more efficient use of the resources.

Second, the density-based algorithms might generate too many micro-clusters leading to inefficiency in computation and memory usage. Some algorithms, for example, CluStream, and E-stream [25], reduce the number of clusters by introducing the merging operation. Yet all these algorithms use an enlarged spherical-shape cluster to cover the merged micro-clusters. With the radial enlargement of a sphere, the resulting enlarged cluster will cover too much unwanted space, possibly leading to false clustering results, which is the drawback of using a larger spherical micro-cluster to cover the merged micro-cluster.

Finally, the existing stream data clustering algorithms aim at capturing the dynamic of clusters by treating the impact of new data points more significantly than those of the old ones. Several approaches have been proposed, including Landmark Window Model, Sliding Window Model, and Fading Window Model [2]. Essentially, these approaches allow the weight of each data point to last over a predefined period of time before being excluded from the clustering. Since the old data records are removed from the system in a predefined time, this removal scheme can be considered as *scheduled removals* of data.

However, the objective of clustering streaming data is not necessarily limited to observing evolving clusters. For example, when data are the online processing of

bank accounts via a web site, we would not treat the old accounts less importantly than the new ones. Moreover, some customers might close their accounts at any time they prefer, leading to *unscheduled removal* of data records from the system at any arbitrary time. In this situation, we want an algorithm that considers clusters as the result of all data points in the time horizon with the capacity of allowing removals of data records at any arbitrary time.

To address the issue regarding the merging of micro-clusters and the unscheduled removals of data records, we introduced a new method based on our proposed concept of hyper-cylindrical micro-clustering called HCMstream. The enlargement of a hyper-cylinder is restricted in one direction, leading to the resulting micro-cluster able to maintain the compactness of the original volume before the merging. The final clusters are obtained from sets of connected micro-clusters. Therefore, the algorithm can recognize non-convex clusters.

HCMstream allows the removals of data records at arbitrary time, in contrast to existing algorithms which fix the lifetime of each data record at constant, and nullify the impact of the expired record. The lifetime of each data record in HCMstream is not predefined. Without the whole data set stored in memory, we cannot determine which data record belong to which micro-cluster. In removing a data record, we use the reverse clustering process by mapping a removed record into a different kind of data point called a *destructor*. When destructors appear in any area considerably, indicating that there are many records corresponding to data points in the vicinity being removed, the cluster in that area would be dissociated. The space at which the cluster of destructors formerly occupied would become vacant, indicating that there are not enough data points forming cluster in that area. By treating the removed records and the added records as data points with different labels, we can use the clustering process to process both operations without keeping the whole data set.

In addition, unlike existing algorithms which require online and offline phases in clustering, HCMstream performs clustering in online fashion by incrementally updating adjacency matrix whenever there is a change in the configuration of the micro-clusters. As a result, no offline phase is required in this algorithm.

In summary, HCMstream includes the following features:

- HCMstream uses hyper-cylindrical micro-clusters generated by merging traditional spherical micro-clusters, resulting in less micro-clusters while their compactness is maintained.
- The algorithm allows the removals of data at any arbitrary time which is a new feature not addressed in other existing algorithms for streaming data.
- The clustering process is performed online. No offline phase is required.

3.1 Definitions

In this section, definitions of a hyper-sphere and a hyper-cylinder are presented. Then, we will present the concepts of a spherical micro-cluster and a cylindrical micro-cluster for capturing data points into local clusters.

Definition 1: A hyper-sphere $S(\mathbf{c}, r)$ in an n-dimensional space with a fixed center $\mathbf{c}$ and radius r is a set of data points $\mathbf{x}_i = (\mathbf{x}_i^1, \ldots \mathbf{x}_i^d)$ defined as follows

$$S(\mathrm{c}, r) = \{\mathbf{x}_i | \|\mathbf{x}_i - \mathrm{c}\| \leq r\}$$

Definition 2: A hyper-cylinder $Cyl(\mathrm{c}, r, \mathbf{l}, L)$ in n-dimensional space with center $\mathbf{c}$ and radius r , and length L extending in the direction of a unit directional vector $\mathbf{l}$, is a set of data points $\mathbf{x}_i = (\mathbf{x}_i^1, \ldots \mathbf{x}_i^d)$ defined as follows

$$Cyl(\mathrm{c}, r, \mathbf{l}, L) = \{\mathbf{x}_i | \|(\mathbf{x}_i - \mathrm{c}).\mathbf{l}\| \leq L, \text{ and } \|\mathbf{x}_i - \mathrm{c}\text{-}((\mathbf{x}_i - \mathrm{c}).\mathbf{l})\| \leq r\}.$$

Note that length L is defined as the distance from center $\mathbf{c}$ to either end of the hyper-cylinder. From the definition, a two-dimensional cylinder is a rectangle with the width equal to $2r$ and the length of $2L$ while a three-dimensional cylinder is a cylinder with circular cross section of radius r with the length of $2L$ as shown in Figure 6. In n dimensional space, a hyper-cylinder is defined with cross section of $n-1$ dimensional hyper-sphere and one dimensional axial length. Regardless of n

dimensions, the terms hyper-sphere and hyper-cylinder will be shortened and renamed as sphere and cylinder in this study for ease.

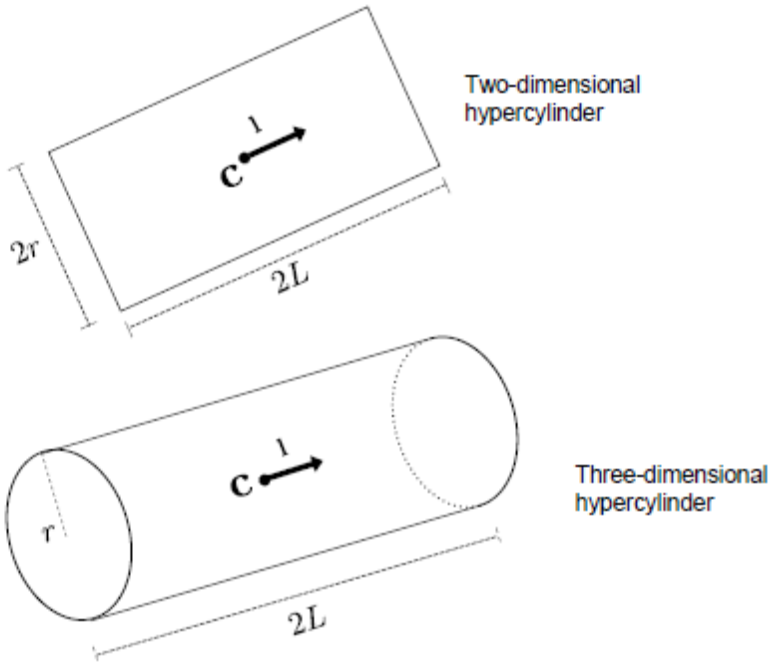


Figure 6 Two shapes of hyper-cylinder in different dimensions. The top shape is the hyper-cylinder in a 2-dimensional space. The bottom shape is the hyper-cylinder in a 3-dimensional space

Definition 3: A spherical micro-cluster, denoted as $\mathcal{SM}$, is a cluster of data points represented by a sphere $S(\mathbf{c}, r)$ centered at $\mathbf{c}$ with a radius r.
A spherical micro-cluster is defined by two parameters as $\mathcal{SM} = \{\mathbf{c}, N\}$, where $\mathbf{c}$ is the center of $\mathcal{SM}$ and N is the number of data points inside. Note that r is not a parameter of a spherical micro-cluster because in this algorithm the radius of all micro-cluster is fixed at a constant provided from users.

Definition 4: A cylindrical micro-cluster, denoted as $\mathcal{CM}$, is a cluster of data points represented by a cylinder $Cyl(\mathbf{c}, r, \mathbf{l}, L)$ centered at $\mathbf{c}$ with a constant radius r, an axial unit vector $\mathbf{l}$, and length L.

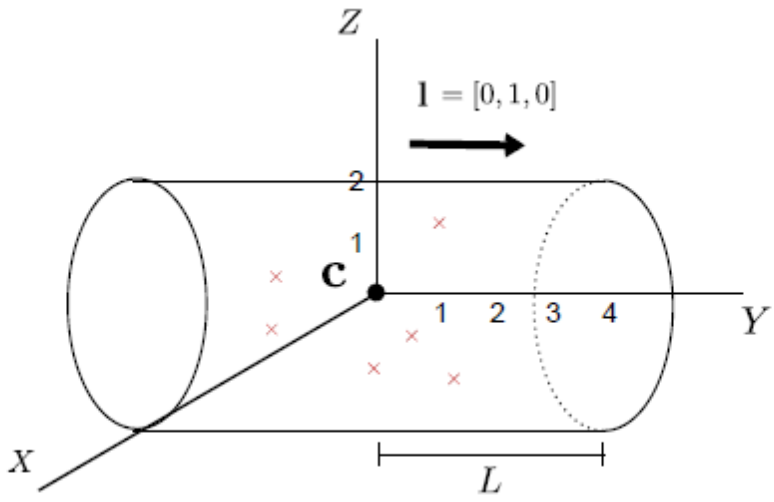


Figure 7 An example of a cylindrical micro-cluster in a 3-dimensional space..

A cylindrical micro-cluster is defined using four parameters as $\mathcal{CM} = \{\mathbf{c}, N, \mathbf{l}, L\}$, where $\mathbf{c}$ is the center of $\mathcal{CM}$ and N is the number of data points inside, $\mathbf{l}$ is the axial unit vector, and L is the length from one end to its center. Figure 7 shows an example of a 3-dimensional cylindrical micro-cluster. There are six data points depicted as red crosses. The axial unit vector is $\mathbf{l} = [0,1,0]^T$ and the length of cylinder is 4. The cluster is represented by $\mathcal{CM} = \{[0,0,0]^T, 6, [0,1,0]^T, 4\}$. From the definitions of spherical and cylindrical micro-cluster, we define the following sets. Let $\boldsymbol{\mathcal{C}} = \{\mathcal{CM}_1, \ldots, \mathcal{CM}_{|\mathcal{C}|}\}$ be a set of cylindrical micro-clusters where $\mathcal{CM}_i$ denotes the i th cylindrical micro-cluster and $\mathcal{CM}_i = \{\mathbf{c}_i, N_i, \mathbf{l}_i, L_i\}$. Let $\boldsymbol{\mathcal{S}} = \{\mathcal{SM}_1, \ldots, \mathcal{SM}_{|\mathcal{S}|}\}$ be a set of spherical micro-clusters where $\mathcal{SM}_i$ denotes the i th spherical micro-cluster and $\mathcal{SM}_i = \{\mathbf{c}_i, N_i\}$. Let $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{C}} \cap \boldsymbol{\mathcal{S}} = \{\mathcal{M}_1, \ldots, \mathcal{M}_{|\mathcal{M}|}\}$ be a set of micro-clusters where $\mathcal{M}_i$ can be either spherical or cylindrical micro-cluster.

To allow the unscheduled removals of existing data in streaming environment, we define two types of data points as a constructor and a destructor as follow.

Definition 5: A constructor is a data point contributing to the formation of clusters.

Definition 6: A destructor is a data point contributing to the dissociation of clusters.

In this algorithm, an incoming data point is labeled as either a constructor or a destructor. A constructor is generated in response to the arrival of a new data record by mapping the record into a point in the feature space. It functions as an ordinary data point in other clustering algorithms. When constructors appear densely in a specific area, a cluster is formed in that area. On the other hand, a destructor is generated in response to the request for removing an existed data record by re-mapping that record into a data point. As a result, a destructor is a data point that re-appear in the same location as its counterpart constructor. When destructors appear densely in a specific area, indicating that significant amount of data records corresponding to data points in that area no longer exist, clusters in that area are consequently nullified.

3.2 Overview of HCMstream Operation and Structure

A new data point arrives in response to either adding a new data record or removing an existing data record. Adding a data record generates a constructor, while deleting one produces a destructor. Clusters are formed in the area where constructors appear densely and dissolved in the area where destructors appear densely. To process incoming data only once without keeping the raw data set, HCMstream uses micro-clusters to keep the statistics of local data.

There are two types of micro-clusters in HCMstream, i.e., spherical and cylindrical micro-clusters. A spherical micro-cluster is used as a fundamental unit to represent a local cluster. A new micro-cluster is first generated as a spherical micro-cluster. Then, several overlapping spherical micro-clusters will form a cylindrical micro-cluster. A set of overlapping micro-clusters forms a single cluster. Figure 8 shows clusters formed by overlapping micro-clusters. Micro-clusters 1, 2, and 4 are spherical while micro-cluster 3 is cylindrical. Micro-clusters 1 and 2 overlap with each other, forming one cluster denoted as Cluster 1 whereas micro-clusters 3 and 4 form another cluster denoted as Cluster 2.

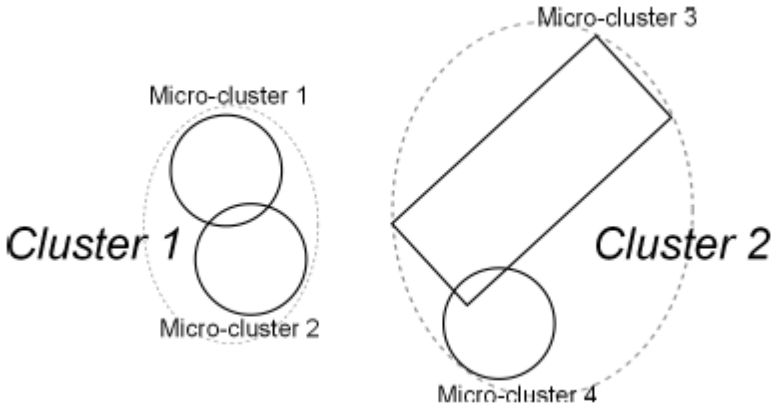


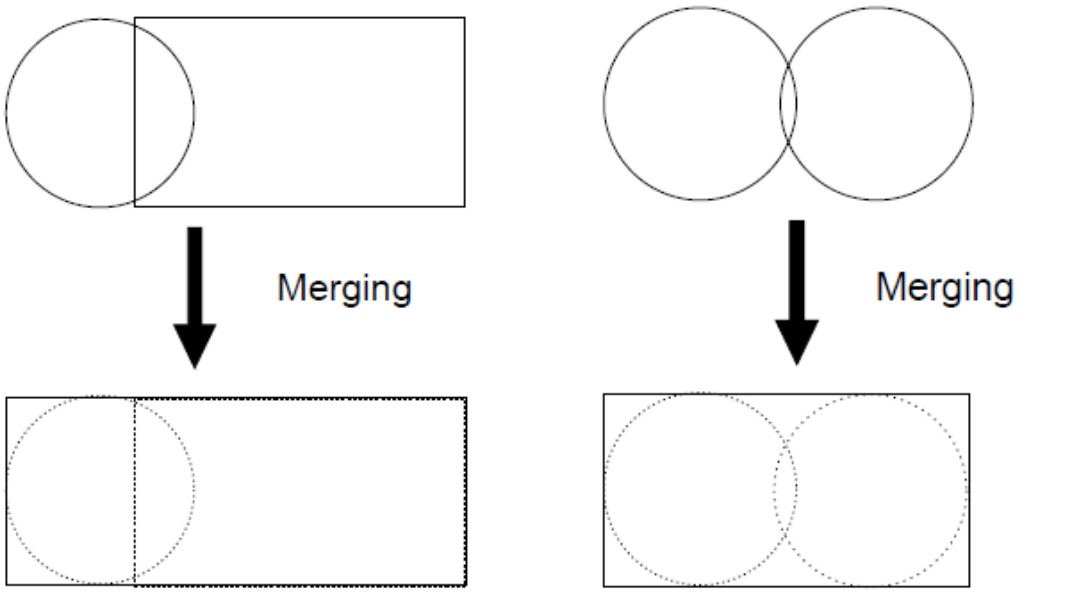
Figure 8 An Example of how clusters are formed by connected micro-clusters

3.2.1 Micro-clustering constructors

When an incoming datum is a constructor, the algorithm would assign it to a micro-cluster, a process called micro-clustering. HCMstream uses the distance between the new data point and the micro-clusters as the criterion for the assignment. If the point falls into any micro-cluster, it is assigned to that micro-cluster. Otherwise, a new spherical micro-cluster is created with the center at that data point.

3.2.2 Merging micro-clusters

After micro-clustering, if the incoming datum is assigned to a spherical micro-cluster, the algorithm would determine whether this micro-cluster can be merged with other micro-clusters. HCMstream allows two types of merging, namely merging a spherical micro-cluster into a cylindrical micro-cluster and merging several spherical micro-clusters into a new cylindrical micro-cluster as shown in Figure 9a and Figure 9b, respectively. The criteria and details of the merging will be explained in detail later.

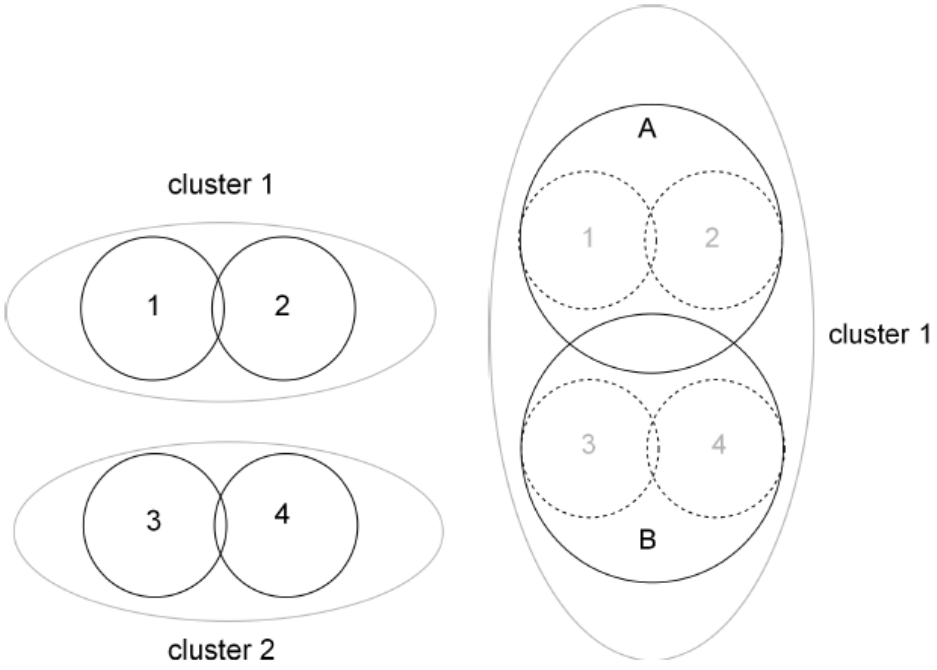


(a) Merging a spherical micro-cluster into a cylindrical micro-cluster.

(b) Merging two spherical micro-clusters into a new cylindrical micro-cluster.

Figure 9 Two merging types of micro-clusters. The left image shows the first merging type and the right image shows the second merging type.

As briefly introduced before, the use of a larger spherical micro-cluster as the result of the merging can lead to incorrect clustering results, due to the over expansion of the sphere. This problem is illustrated in Figure 10a and Figure 10b. In Figure 10a, micro-clusters 1 and 2 overlap with each other forming cluster 1 denoted by elliptic dash line, while micro-clusters 3 and 4 forming cluster 2. If the overlapping micro-clusters are merged into a larger spherical micro-cluster, micro-clusters 1 and 2 are merged into micro-cluster A, while micro-clusters 3 and 4 are replaced by micro-cluster B as shown in Figure 10b. The newly created micro-clusters A and B are so large that they overlap with each other, incorrectly forming only one cluster as cluster 1 as shown in Figure 10b. As a result, replacing merged micro-clusters by a larger spherical micro-cluster leads to the incorrect clustering result of one cluster as shown in Figure 10b while the correct clustering result should be two clusters as in Figure 10a before the merging.



(a) Micro-clusters without merging. There are two final clusters.

(b) When overlapping micro-clusters are merged into a larger spherical micro-cluster, two micro-clusters resulting from the merging overlap with each other, forming one final cluster.

Figure 10 Merging micro-clusters into a larger spherical micro-cluster causes incorrect clusters.

HCMstream uses a cylindrical micro-cluster as the result of the merging. An n dimensional cylindrical micro-cluster has a fixed cross section of an $n-1$ dimensional sphere while its length can be extended due to the merging of a spherical micro-cluster. The advantage of using a cylindrical micro-cluster over a spherical micro-cluster can be intuitively illustrated in Figure 11. Micro-cluster A and B are connected with each other. Two lower images show the scenarios when using a sphere and a cylinder to cover the two micro-clusters. Irrelevant space denoted by shaded area caused by using a sphere is much larger than that using a cylinder. In contrast to the expansion in all dimension of a sphere due to the enlarging radius, the expansion of a cylindrical micro-cluster is restricted to a single dimension of the

axial direction. As a result, the cylindrical micro-cluster can be enlarged in the most compact fashion.

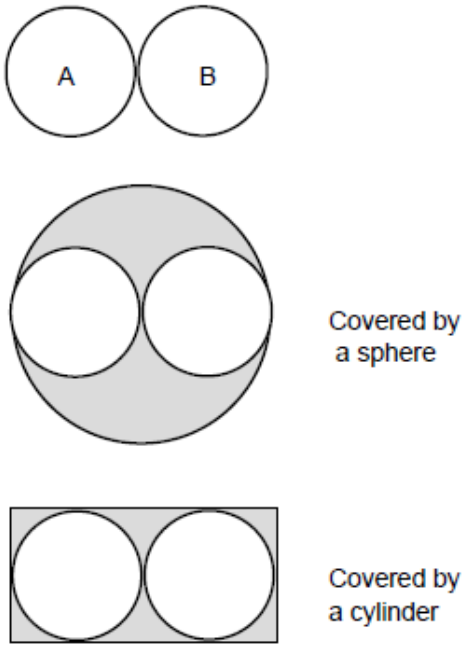
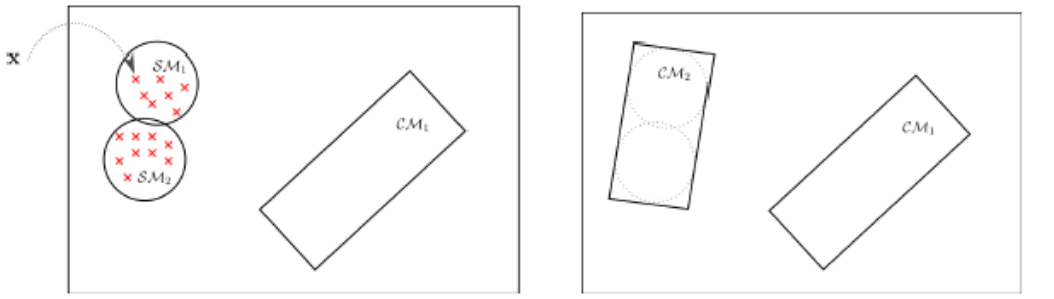


Figure 11 Illustration of covering two connected micro-clusters by a sphere and a cylinder.

Figure 12 illustrates the operations when a constructor, depicted as a red cross, arrives. Figure 12a shows the micro-clustering step when the constructor is assigned to $\mathcal{SM}_1$. Note that although we depict several constructors in $\mathcal{SM}_1$ and $\mathcal{SM}_2$ to show that these two micro-clusters have dense constructors, actually the previous constructors are not retained in the memoryFigure 12b shows the merging step when $\mathcal{SM}_1$ is merged with $\mathcal{SM}_2$ forming a new cylindrical micro-cluster denoted as $\mathcal{CM}_2$.



(a) A new datum $\mathbf{x}$ labeled as a constructor is micro-clustered into $\mathcal{SM}_1$.

(b) The last updated micro-cluster $\mathcal{SM}_1$ is merged with its overlapping micro-cluster $\mathcal{SM}_2$ to form $\mathcal{CM}_2$.

Figure 12 Illustration of the operations of HCstream when an incoming datum is a constructor.

3.2.3 Micro-clustering destructors and removing micro-clusters

When a destructor arrives, HCMstream assigns it to the nearest micro-cluster. When destructors occur densely in any area, the cluster in that area is dissociated by removing the part of the micro-cluster with dense destructors. Each time when dissolving a cluster, HCMstream removes a part of the cluster equivalent to one spherical micro-cluster.

The removal of a spherical micro-cluster is illustrated in Figure 13. There are three spherical micro-clusters denoted by $\mathcal{SM}_1$, $\mathcal{SM}_2$, and $\mathcal{SM}_3$ forming one cluster denoted as Cluster 1. When adding a new destructor to $\mathcal{SM}_2$ causes the number of destructors in $\mathcal{SM}_2$ to go beyond the threshold, $\mathcal{SM}_2$ is removed from the feature space, leaving only $\mathcal{SM}_1$ and $\mathcal{SM}_3$ forming two separated clusters denoted as Cluster 1 and Cluster 2.

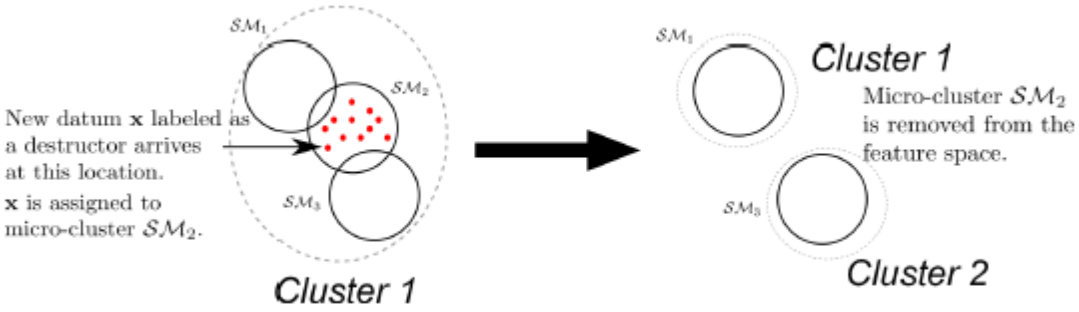
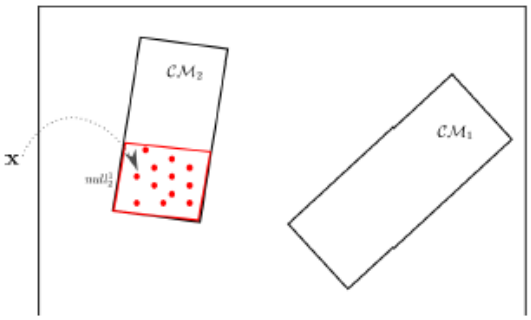


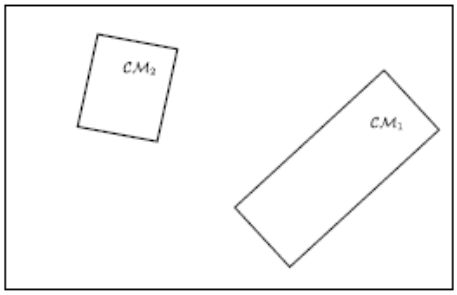
Figure 13 An example of removing destructors which results in the separation of original cluster into two clusters.

On the other hand, since a cylindrical micro-cluster consists of several micro-cluster being merged together, we cannot simply remove the whole cylindrical micro-cluster. Only the part of it with dense destructors should be removed while other parts without dense destructors must be preserved. To locate destructors inside a cylindrical micro-cluster, we introduce another type of a micro-cluster called a null micro-cluster. When a null micro-cluster has sufficient destructors, it is removed together with the part of the cylindrical micro-cluster superimposed by that null micro-cluster, leading to breaking the cylindrical micro-cluster into parts.

Figure 14 depicts the operations when a destructor, depicted as a red circle, arrives. Figure 14a shows the micro-clustering step when the destructor is assigned to $\mathcal{CM}_2$, where a null micro-cluster $null_2^1$ is used to locate the destructors. Figure 14b shows the breaking of $\mathcal{CM}_2$ by removing $null_2^1$ together with the part of $\mathcal{CM}_2$ superimposed by $null_2^1$, leading to $\mathcal{CM}_2$ being shortened.



(a) A new datum x labeled as a destructor is micro-clustered into $\mathcal{CM}_2$. A null micro-cluster $null_2^1$ depicted as a red rectangle is used to locate the destructors.



(b) $null_2^1$ is removed from the feature space together with a part of $\mathcal{CM}_2$.

Figure 14 Illustration of the operations of HCstream when an incoming datum is a destructor.

The algorithm requires two input parameters from user, namely np, and r. Parameter np is the threshold for determining outliers. When the number of constructors in any spherical micro-cluster goes beyond this value, it is considered that data points in that micro-cluster are real data, not outliers or noise. Thus, the micro-cluster is taken into consideration in clustering. On the other hand, micro-clusters whose data points inside are less than np are not used in clustering. r is the fixed radius of spherical and cylindrical micro-clusters. The radius of all micro-clusters is set at constant for the whole process.

The clustering result is reported in the form of adjacency matrix $A = [A_{ij}{}^{|\mathcal{M}|\times|\mathcal{M}|}]$ such that $A_{ij} = 1$, when $\mathcal{M}_i$and $\mathcal{M}_j$are connected, and $A_{ij} = 0$, otherwise. Where $\mathcal{M}_i$and $\mathcal{M}_j$ are micro-clusters i and j which can be either spherical or cylindrical micro-clusters. Connected components of the graph induced by A define the resulting clusters.

HCMstream updates A in an incremental fashion. When there are changes in the configuration of micro-clusters resulting from either creating, merging, or removing micro-clusters, the algorithm would recheck the connection of the micro-clusters being modified and other micro-clusters connected to the modified micro-

clusters for updating the adjacency matrix. As a result, the algorithm does not require the offline phase.

3.3 Algorithm

Clustering incoming constructors

Algorithm 1: Clustering incoming datum

1 Calculate the distances between $\mathbf{x}$ to $\mathcal{CM}_j$, $\forall \mathcal{CM}_j \in \boldsymbol{\mathcal{C}}$ using equation (1).
2 If $\mathbf{x}$ satisfies the conditions in equation (2) with respect to the nearest micro-cluster $\mathcal{CM}_j$
3 Update the parameters of $\mathcal{CM}_j$ using equation (3)
4 Return
5 EndIf
6 Calculate $\mathrm{d}(\mathbf{x}, \mathcal{SM}_k)$, $\forall \mathcal{SM}_j \in \boldsymbol{\mathcal{S}}$ using equation (4)
7 If $\mathbf{x}$ satisfies the condition in equation (5) with respect to the nearest micro-cluster $\mathcal{SM}_K$
8 Update the parameter of $\mathcal{SM}_K$ using equation (6)
9 If $\mathrm{N_K} > \mathrm{np}$
10 Go to line 16
11 EndIf
12 Return
13 EndIf
14 Let $\mathcal{SM}_{\mathrm{new}}(\mathbf{c}_{\mathrm{new}}, \mathrm{N}_{\mathrm{new}}) = (\mathbf{x}, 1)$, $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{S}} \cup \mathcal{SM}_{\mathrm{new}}$
15 Return
16 Calculate distances from $\mathcal{SM}_K$ to $\mathcal{CM}_j$, $\forall \mathcal{CM}_j \in \boldsymbol{\mathcal{C}}$, using equation (7)
17 If $\mathcal{SM}_K$ can be merged into cylindrical micro-cluster $\mathcal{CM}_J$, according to condition (8)
18 Update $\mathcal{CM}_J$, using equation (9) and (10)
19 Update adjacency matrix $\mathbf{A}$ by transferring all connected components of $\mathcal{SM}_K$ to $\mathcal{CM}_J$

20 Remove $\mathcal{SM}_{\mathrm{K}}$ from the feature space

21 Return

22 EndIf

23 Calculate $\mathrm{d}(\mathcal{SM}_{\mathrm{k}}, \mathcal{SM}_{\mathrm{i}})$, $\forall \mathcal{SM}_{\mathrm{i}} \in \boldsymbol{\mathcal{S}}$ and $\mathrm{N_i} > \mathrm{np}$

24 If New cylindrical micro-cluster $\mathcal{CM}_{\mathbf{new}}$ can be created from $\mathcal{SM}_{\mathrm{K}}$

25 Create $\mathcal{CM}_{\mathbf{new}}$ using equation (11)

26 Update A by transferring all connected components of micro-clusters in $\boldsymbol{\mathcal{S}}_{\mathbf{merge}}$ to $\mathcal{CM}_{\mathbf{new}}$

27 $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{S}}_{\mathbf{merge}}$ and $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{C}} \cup \mathcal{CM}_{\mathbf{new}}$

28 EndIf

29 Return

3.3.1 Micro-clustering constructors

HCMstream tries to merge an incoming constructor with existing cylindrical micro-clusters first (lines 1-5). Let $\mathbf{x}$ be an incoming constructor and $\mathcal{CM}_j$ be a cylindrical micro-cluster defined by $\mathcal{CM}_j = \{\mathbf{c}_j, N_j, \mathbf{l}_j, L_j\}$. $d_{para}(\mathbf{x}, \mathcal{CM}_j)$ and $d_{perp}(\mathbf{x}, \mathcal{CM}_j)$ are the parallel and perpendicular distances from $\mathbf{c}_j$ to $\mathbf{x}$ with respect to the directional vector $\mathbf{l}_j$ and $\delta_{\mathbf{xj}} = \mathbf{x}\text{-}\mathbf{c}_j$.
The distances are determined from

$$\begin{aligned} d_{para}(\mathbf{x}, \mathcal{CM}_j) &= |\delta_{\mathbf{xj}} . \mathbf{l}_j| \\ d_{perp}(\mathbf{x}, \mathcal{CM}_j) &= \|\delta_{\mathbf{xj}} - (\delta_{\mathbf{xj}} . \mathbf{l}_j)\mathbf{l}_j\| \end{aligned} \quad (1)$$

Point $\mathbf{x}$ is inside $\mathcal{CM}_j$ when

$$\begin{aligned} d_{para}(\mathbf{x}, \mathcal{CM}_j) &\leq L_j, \text{ and} \\ d_{perp}(\mathbf{x}, \mathcal{CM}_j) &\leq r \end{aligned} \quad (2)$$

We assign $\mathbf{x}$ to $\mathcal{CM}_j$ such that among j that satisfy condition (2) $J = \underset{j}{\mathbf{argmin}}\ d_{perp}(\mathbf{x}, \mathcal{CM}_j)$ If $\mathbf{x}$ can be assigned to $\mathcal{CM}_J$, the parameters of $\mathcal{CM}_J$ are updated as

$$N_J^{new} = N_J + 1 \quad (3)$$

If $\mathbf{x}$ cannot be assigned to any cylindrical micro-clusters, it is checked with spherical micro-clusters (lines 6-8).The distance between point $\mathbf{x}$ and micro-cluster $\mathcal{SM}_k$ defined by $\mathcal{SM}_k = \{\mathbf{c}_k, N_k\}$ is calculated by

$$d(\mathbf{x},\mathcal{SM}_k) = \|\mathbf{x} - \mathbf{c}_k\| \tag{4}$$

Let $K = \underset{k}{\mathbf{argmin}}\ d(\mathbf{x}, \mathcal{SM}_k)$, point $\mathbf{x}$ is inside $\mathcal{SM}_k$ when

$$d(\mathbf{x},\mathcal{SM}_K) \leq r, \tag{5}$$

where r is the radius of $\mathcal{SM}_K$. If $\mathbf{x}$ is in $\mathcal{SM}_K$, its parameters are updated as

$$\mathbf{c}_K^{new} = \frac{\mathbf{c}_K N_K + \mathbf{x}}{N_K + 1}$$
$$N_K^{new} = N_K + 1. \tag{6}$$

If $\mathbf{x}$ is not inside any existing micro-clusters, a new micro-cluster $\mathcal{SM}_{new}$ is created to cover $\mathbf{x}$ (lines 14).

3.3.2 Merging micro-clusters

When $\mathcal{SM}_K$ is updated through adding a new data point, if the population in the micro-cluster is dense (line 10), we would try to merge it with existing cylindrical micro-cluster first (lines 16-20). Let $\Delta_{\text{jK}} = \mathbf{c}_j - \mathbf{c}_K$, the distance between $\mathcal{SM}_K$ and $\mathcal{CM}_j$ is calculated by

$$d_{para}(\mathcal{SM}_K, \mathcal{CM}_j) = |\Delta_{\text{jK}}.\mathbf{l}_j|$$
$$d_{perp}(\mathcal{SM}_K, \mathcal{CM}_j) = \|\Delta_{\text{jK}} - (\Delta_{\text{jK}}.\mathbf{l}_j)\mathbf{l}_j\| \tag{7}$$

$\mathcal{SM}_K$ and $\mathcal{CM}_j$ overlap when

$$d_{para}(\mathcal{SM}_K, \mathcal{CM}_j) \leq L_j + r, \text{and}$$
$$d_{perp}(\mathcal{SM}_K, \mathcal{CM}_j) \leq 2r \tag{8}$$

We merge $\mathcal{SM}_K$ with $\mathcal{CM}_J$ such that among j that satisfy condition (8), $J = \underset{j}{\text{argmin}}\ d_{perp}(\mathcal{SM}_K, \mathcal{CM}_j)$. Condition (8) indicates that $\mathcal{SM}_K$ can be merged with $\mathcal{CM}_J$ when its center lies inside the shaded area shown in Figure 15.

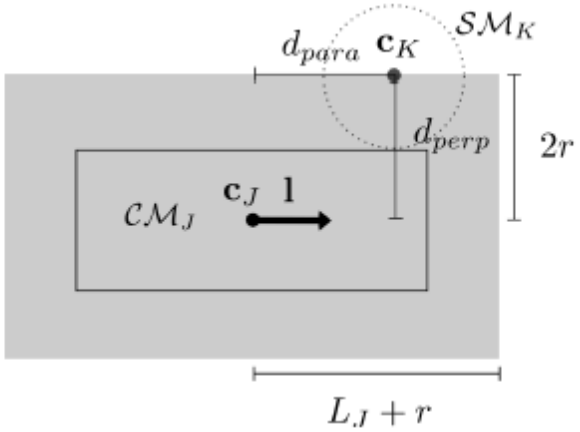


Figure 15 Merging boundary of cylindrical micro-cluster $\mathcal{CM}_J$.

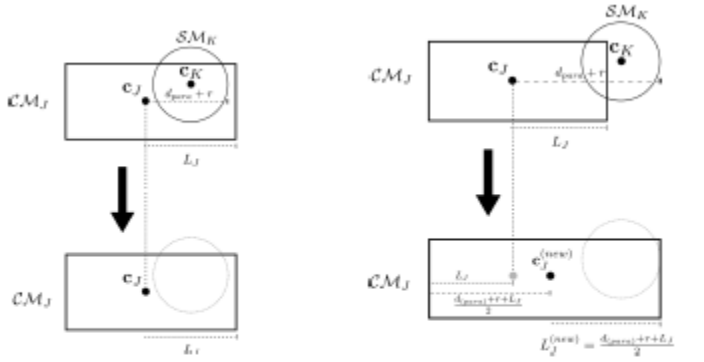
After $\mathcal{SM}_K$ is merged into $\mathcal{CM}_J$, $\mathcal{SM}_K$ is removed from the feature space. The number of data in $\mathcal{CM}_J$, N_J^{new}, is updated as

$$N_J^{new} = N_J + N_K \tag{9}$$

If $d_{para} + r \leq L_J$, $\mathcal{CM}_J$ can cover $\mathcal{SM}_K$ as shown in.Figure 16 a. The configuration of $\mathcal{CM}_J$ remains the same, as a result, other parameters are not updated. On the contrary, when $d_{para} + r > L_J$, $\mathcal{CM}_J$ cannot cover $\mathcal{SM}_K$ as shown in Figure 16b. The length of $\mathcal{CM}_J$ must be extended so that $\mathcal{CM}_J$can cover $\mathcal{SM}_K$. As a result, the new center $\mathbf{c}_J^{new}$ and the length L_J^{new} of $\mathcal{CM}_J$ become

$$L_J^{new} = \frac{d_{para} + L_J + r}{2}$$
$$\mathbf{c}_J^{new} = \mathbf{c}_J + \frac{d_{para}+L_J+r}{2}.\text{sign}(\Delta_{\text{KJ}}.\mathbf{l}_J)\mathbf{l}_J \tag{10}$$

where $\Delta_{KJ} = \mathbf{c}_K - \mathbf{c}_J$ and $\text{sign}(\Delta_{\text{KJ}}.\mathbf{l}_J)$ is the sign of the dot product between Δ_{KJ} and $\mathbf{l}_J$.



(a) $\mathcal{CM}_J$ merging with $\mathcal{SM}_K$ when its length remains the same. (b) $\mathcal{CM}_J$ merging with $\mathcal{SM}_K$ when its length is extended.

Figure 16 Two cases of merging $\mathcal{SM}_K$ into $\mathcal{CM}_J$.

If $\mathcal{SM}_K$ cannot be merged with other cylindrical micro-cluster, it is subsequently determined if it overlaps other spherical micro-clusters. The overlapping spherical micro-clusters are merged together and replaced by a new cylindrical micro-cluster (lines 23-28). Two spherical micro-clusters $\mathcal{SM}_K$ and $\mathcal{SM}_i$ overlaps if $\mathbf{c}_K - \mathbf{c}_i \leq 2r$. Let $\boldsymbol{\mathcal{S}}_{merge} = \{\mathcal{SM}_1, \ldots, \mathcal{SM}_m\}$ be a set of overlapping micro-clusters for creating a new cylindrical micro-cluster $\mathcal{CM}_{new}$ whose parameters $\{\mathbf{c}_{new}, N_{new}, \mathbf{l}_{new}, L_{new}\}$are determined as follows. Let $\bar{\mathbf{c}}$ is the mean of all centers in $\boldsymbol{\mathcal{S}}_{merge}$, $\delta_i = \mathbf{c}_i - \bar{\mathbf{c}}$, and $\mathbf{C} = \{\delta_1, \ldots, \delta_m\}$.

$$\begin{aligned} \mathbf{c}_{new} &= \bar{\mathbf{c}} \\ N_{new} &= \sum_{i}^{m} N_i \\ \mathbf{l}_{new} &= \text{the largest principal component of } \mathbf{C} \\ L_{new} &= |\delta_I . \mathbf{l}_{new}| + r \end{aligned} \tag{11}$$

where $I = \underset{j}{\mathbf{argmax}} \, |\delta_i . \mathbf{l}_{new}|$.

The number of points N_{new} inside $\mathcal{CM}_{new}$ is the total number of data in all of the spherical micro-clusters in $\boldsymbol{\mathcal{S}}_{merge}$. The center $\mathbf{c}_{new}$ is the mean of the spherical micro-clusters. The axial direction $\mathbf{l}_{new}$ is set along the alignment direction of the spherical micro-clusters.

3.3.3 Micro-clustering destructors and removing micro-clusters

Algorithm 2: Removing destructors from a micro-cluster

1	Determine $\mathrm{d}(\mathbf{x}, \mathcal{SM}_{\mathrm{K}})$ using equation (4).
2	If $\mathbf{x}$ is inside its nearest micro-cluster $\mathcal{SM}_{\mathrm{K}}$ according to condition (5)
3	$\mathrm{N}_{\mathrm{K}} = \mathrm{N}_{\mathrm{K}} - 1$
4	If $\mathrm{N}_{\mathrm{K}} = 0$
5	Remove $\mathcal{SM}_{\mathrm{K}}$ from the feature space.
6	Update the connection information of relevant micro-clusters in $\mathbf{A}$.
7	EndIf $\mathcal{SM}_{\mathrm{K}}$
8	Else
9	Determine $\mathrm{d}(\mathbf{x}, \mathcal{CM}_{\mathrm{J}})$, by using equation (1)
10	If $\mathbf{x}$ is inside a null micro-cluster $\mathbf{null}_{\mathrm{i}}^{\mathrm{J}}$ in $\mathcal{CM}_{\mathrm{J}}$
11	$\mathrm{N}_{\mathrm{i}}^{\mathrm{J}} = \mathrm{N}_{\mathrm{i}}^{\mathrm{J}} + 1$
12	If $\mathrm{N}_{\mathrm{i}}^{\mathrm{J}} > \mathrm{N}_{\mathrm{J}}/\mathrm{L}_{\mathrm{J}}$
13	Remove $\mathbf{null}_{\mathrm{i}}^{\mathrm{J}}$ and update $\mathcal{CM}_{\mathrm{J}}$ using either (12) or (13).
14	Update the connection information of relevant micro-clusters in $\mathbf{A}$.
15	EndIf
16	Else
17	Create a new null micro-cluster for $\mathbf{x}$ in $\mathcal{CM}_{\mathrm{J}}$.
18	EndIf
19	EndIf

Algorithm 2 shows the process for clustering and removing destructors. Assume that the incoming datum $\mathbf{x}$ is a destructor. We try to assign it to a spherical micro-cluster first by determining the distance between $\mathbf{x}$ and all spherical micro-

cluster according to equation (4). If none of spherical micro-cluster can cover $\mathbf{x}$, we determine if it is covered by any cylindrical micro-cluster using condition (2).

In the case that a destructor is located outside all micro-clusters, it is assigned to the closest micro-cluster by comparing $d(\mathbf{x}, \mathcal{SM}_K)$ and $d(\mathbf{x}, \mathcal{CM}_J)$. $\mathbf{x}$ is assigned to $\mathcal{SM}_K$ if

$$d(\mathbf{x}, \mathcal{SM}_K) - r < (d_{perp}(\mathbf{x}, \mathcal{CM}_J) - r).\sqrt{\frac{n}{n-1}}$$

Otherwise, it is assigned to $\mathcal{CM}_J$. Note that since $d_{perp}(\mathbf{x}, \mathcal{CM}_J)$ is the projected distance in $n-1$ dimensions, the coefficient $\sqrt{\frac{n}{n-1}}$ is added so that it can be compared with $d(\mathbf{x}, \mathcal{SM}_K)$ which is in n dimensions.

If $\mathbf{x}$ is assigned to a spherical micro-cluster, we reduce the number of data points in that micro-cluster by one (Line 3). On the other hand, if a destructor is assigned to a cylindrical micro-cluster, a null micro-cluster is introduced to capture the destructors. Each null micro-cluster has the same shape as the cylindrical micro-cluster with a constant length of $2r$ as shown in.Figure 17

Let $null_i^J$ be the i-th null micro-cluster centered at $\mathbf{c}_{null_i}$ in $\mathcal{CM}_J$ with two parameters N_i^J and d_i^J. N_i^J is the number of destructors inside null micro-cluster $null_i^J$ and d_i^J is the signed distance between the center of $null_i^J$ and $\mathcal{CM}_J$. The signed distance d_i^J is calculated as follows.

$$d_1^J = (\mathbf{c}_{null_i} - \mathbf{c}_J).\mathbf{l}_J$$

Figure 17 shows a null micro-cluster $null_1^J$ inside $\mathcal{CM}_J$ with $N_1^J = 4$ and signed distance $d_1^J = -d$. The destructors inside $null_1^J$ are depicted as red circles.

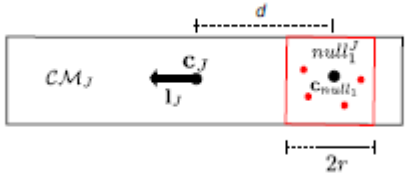


Figure 17 Null micro-cluster $null_1^J$ defined on $\mathcal{CM}_J$.

When the number of destructors inside a null micro-cluster exceeds the predefined threshold, the null micro-cluster is removed. In this study, the threshold value is set to N_J/L_J which is equal to the average density of data inside $\mathcal{CM}_J$. There are two scenarios of removing a null micro-cluster $null_i^J$ from $\mathcal{CM}_J$. Let $d_i^J = d$. The first scenario occurs when $null_i^J$ is at either end of $\mathcal{CM}_J$, which implies that $|d| \geq L_J - r$.

Removing $null_i^J$ only shortens the length of $\mathcal{CM}_J$. The parameters of $\mathcal{CM}_J$ are updated as follows.

$$\begin{aligned} L_J^{new} &= \frac{L_J + |d| - r}{2} \\ \mathbf{c}_J^{new} &= \mathbf{c}_J - \frac{L_J + |d| - r}{2}\text{sign}(d).\mathbf{l}_J \\ N_J^{new} &= N_J - N_i^J \end{aligned} \tag{12}$$

After updating the length of $\mathcal{CM}_J$, the structure of $\mathcal{CM}_J$ must be redefined according to the new length. The shortened micro-cluster can be either cylindrical or spherical depending on the length of the remaining part.

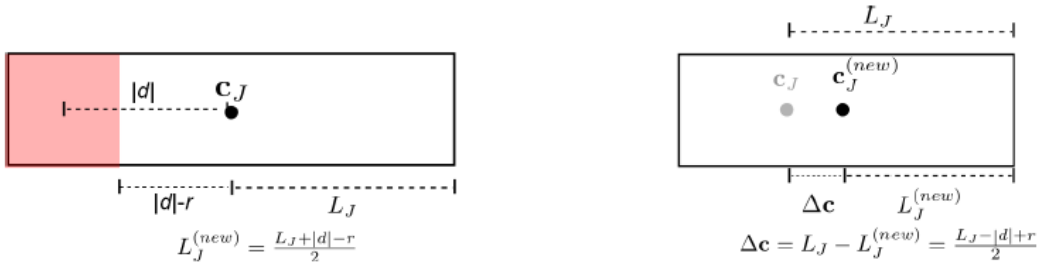


Figure 18 The result of removing a null micro-cluster at one end of a cylindrical micro-cluster and calculating parameters of $\mathcal{CM}_J$. The left image is the situation before removing the null micro-cluster shown in shaded area. The right image is the result of removal and new parameters of both micro-clusters.

If $L_J^{new} \geq r$, $\mathcal{CM}_J$ remains to be a cylindrical micro-cluster. Otherwise, it is replaced by a spherical micro-cluster $\mathcal{SM}_{new}$ with parameters $\{N_J^{new}, \mathbf{c}_J^{new}\}$.

Figure 18 illustrates the diagram for calculating the updated parameters when no new micro-cluster is generated. The shaded area denotes the null micro-cluster to be removed.

In the second scenario, when $|d| < L_J - r$, removing $null_i^J$ causes $\mathcal{CM}_J$ to be broken into two parts. The length of each part is $L_J + |d| - r$ and $L_J - |d| - r$. The longer part remains to be $\mathcal{CM}_J$ while the shorter part becomes a new micro-cluster. The parameters of $\mathcal{CM}_J$ are updated according to equation (12).
The new micro-cluster can be either a spherical or cylindrical micro-cluster by the same criterion. If $L_J - |d| - r > r$, then the new micro-cluster is a cylindrical micro-cluster $\mathcal{CM}_{new}$ with a set of parameters $\{\mathbf{c}_{new}, N_{new}, \mathbf{l}_{new}, L_{new}\}$ calculated as follow.

$$\begin{aligned} L_{new} &= \frac{L_J - |d| - r}{2} \\ \mathbf{c}_{new} &= \mathbf{c}_J + \frac{L_J + |d| + r}{2}\text{sign}(d).\mathbf{l}_J \\ N_{new} &= 0 \\ \mathbf{l}_{new} &= \mathbf{l}_J \end{aligned} \tag{13}$$

On the other hand, if $L_J - |d| - r < r$, then the new micro-cluster is $\mathcal{SM}_{new}$ with parameters $\{\mathbf{c}_{new}, N_{new}\}$ from equation (13). Figure 20 shows the result of removing a null micro-cluster and the diagram for calculating the new parameters. The null micro-cluster is depicted as shaded area. When the null micro-cluster is removed, $\mathcal{CM}_J$ is broken into two parts. The longer part is $\mathcal{CM}_J$ while the shorter part is $\mathcal{CM}_{new}$. We set $\mathcal{CM}_{new}$ as a blank cylindrical micro-cluster with $N_{new} = 0$. If no data point is added into this micro-cluster, it will be excluded from determining the final clusters.

The process for clustering and removing destructors in a cylindrical micro-cluster is summarized in lines 9-18 in Algorithm 2.

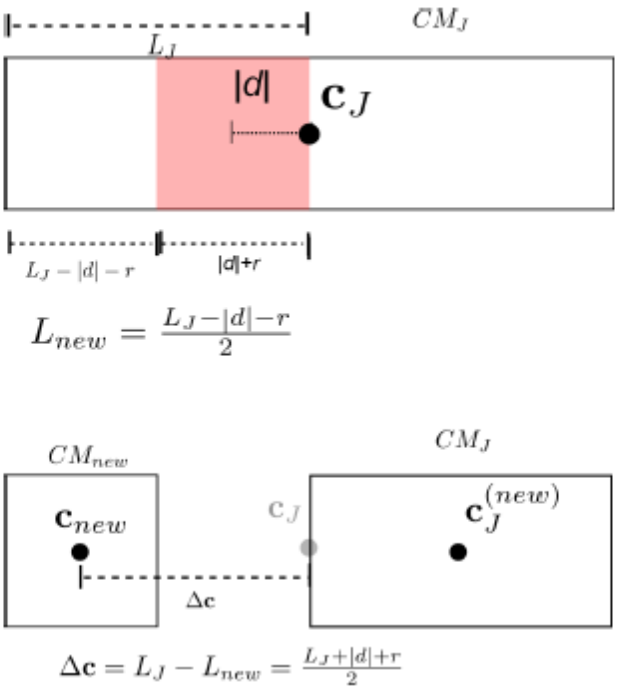


Figure 19 The result of removing a null micro-cluster inside a cylindrical micro-cluster and calculating parameters of $\mathcal{CM}_J$. The upper image is the situation before removing the null micro-cluster shown in shaded area. The lower image is the result of removal and new parameters of both micro-clusters.

Note that during micro-clustering process, the centers of spherical micro-clusters can move from their original positions through parameters update. As a result, some data points might lie outside their possessive micro-clusters. If these data points are constructors, there would not be a problem because our micro-clustering method, which is based on the coverage of micro-cluster, would not assign very far apart points into the same micro-cluster. As a result, micro-clusters would not move too far from their original locations. Moreover, since clustering reflects the collective behavior of data points, it can tolerate individual slight inaccuracies. Small samples of data outside micro-clusters do not have significant impact on the configuration of the resulting clusters.

For the case of destructor, due to changes in the configuration of micro-clusters, some destructors might not necessarily be assigned to the same micro-cluster to which their counterpart constructors belong. However, this would not be a problem as well because we can ensure that at least it must be assigned to the nearby micro-cluster. When there are significant number of destructors, the collective behavior would be reflected by the relevant micro-clusters despite some small inaccuracies during the micro-clustering.

3.4 The complexity of HCMstream

In this section, the computational complexity of the algorithm is analyzed. Let $|\mathcal{C}|$ be the number of cylindrical micro-clusters, and $|\mathcal{S}|$ be the number of spherical micro-clusters. During micro-clustering process, when a datum is assigned to a micro-cluster, computing the distances from the datum to all micro-clusters takes the following time complexity.

$$\mathbf{O}(|\mathrm{C}|)+\mathbf{O}(|\mathrm{S}|)$$

After micro-clustering during the merging process the algorithm, the time complexity of computing the distances from $\mathcal{SM}_K$ to all other micro-clusters is equal to

$$\mathbf{O}(|\mathrm{C}|)+\mathbf{O}(|\mathrm{S}|)$$

Thus the total time complexity is

$$\mathbf{O}(|\mathrm{C}|)+\mathbf{O}(|\mathrm{S}|) <\mathbf{O}(|\mathrm{M}|)$$

where $|\mathbf{M}|$ is the total number of micro-clusters. This algorithm operates in linear scale of the number of micro.

3.5 Experimental results

Two sets of experiments were conducted to evaluate the performance of the proposed algorithm with respect to the other state-of-the-art algorithms. The first set of the experiments was performed with synthetic data sets in 2-dimensional featured spaces. The aim of this set of experiments is to illustrate how micro-clusters and

clusters are formed in HCMstream, compared with other algorithms. Furthermore, we varied the parameters of the algorithms to show the resulting clusters of each algorithms.

The second set of experiments used real data sets in a high dimensional featured space. Since these data sets are too complicated to visualize, we used many well-known similarity measures often used in evaluating clustering results, including normalized mutual information matric (NMI) [21], Rand index (RI) [21, 22], Adjusted Rand index (AR) [23] and Hubert's index (HI) [23].

When comparing the clustering performance of HCMstream with other algorithms, all incoming data must be a constructor, because the function of unscheduled record removal is not available in other algorithms. In algorithms which allow fading weight of old records such as DenStream and D-Stream, we disabled this feature, so that their clustering performance can be compared with that of HCMstream. Moreover, to make the comparison clearer, we slightly modify the parameters of DenStream and D-Stream so that they used the same parameters as those of HCMstream. np was used as the threshold for determining dense local clusters. r was the parameter for determining the size of the micro-cluster. For D-Stream, since it uses hyper-cubical grids for primary clustering, to make r equivalent to that used in HCMstream, each edge of the grids in D-Stream was set at $2r$.

3.5.1 Experiments with synthetic data

We used two synthetic data sets in this set of experiments. The first data set consists of 5300 two-dimensional data points with three clusters as shown in Figure 20. This data set was used to illustrated the influence of the parameters on the clustering results as well as how micro-clusters and clusters were formed. The clustering results of HCMstream were compared with those of DenStream and D-Stream.

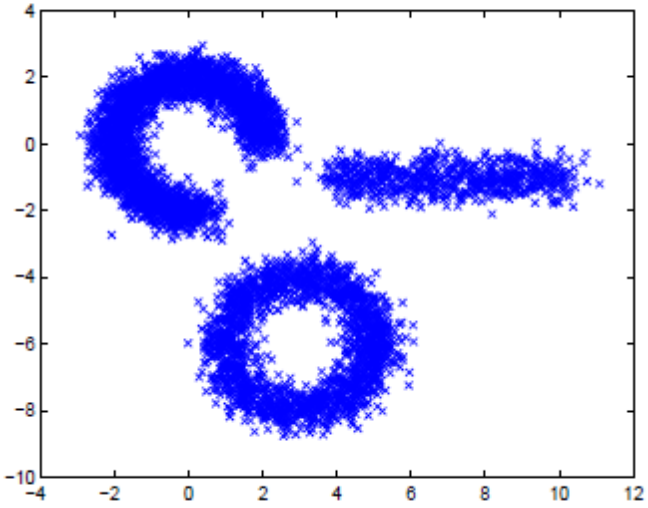


Figure 20 Two-dimensional raw data.

The second data set consisted of two-dimensional data points forming six clusters including convex and non-convex ones with varying densities. Moreover, some 7% additional amount of data were uniformly and randomly generated as noise to the original data set as shown in Figure 21.

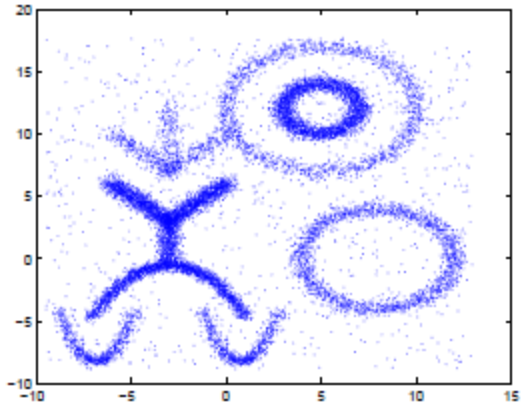


Figure 21 Synthetic data set with non-convex clusters and noise.

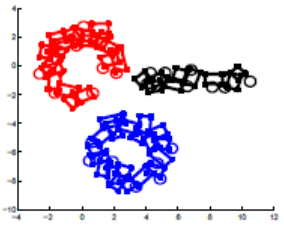
3.5.2 Synthetic Data Sets without Noise

The numbers of micro-clusters and the numbers of clusters were compared among the three algorithms as shown in Table 1. The parameters (r, np) were set at (0.35,5) and (0.5,10) respectively. The clustering results with different values of r are illustrated in and, respectively.

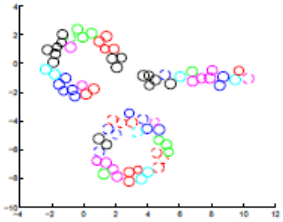
DenStream uses the variance of data points assigned to each micro-cluster as the effective radius of that micro-cluster. The effective radius is usually less than the actual radius for a micro-cluster to cover all data points assigned to it. As a result, compared to the micro-clusters in HCMstream which cover all data points assigned to them, micro-clusters in DenStream can be considered as being shrunk. Due to this shrinkage, micro-clusters that are closed together are less likely to connect with each other, leading to more fragments of clusters in DenStream as shown in Figure 22b and Figure 23b. From Table 1 although micro-clusters produced by DenStream in both cases were not more than those of HCMstream, DenStream could not recognize the three clusters correctly.

When r = 0.35, D-Stream also failed to recognize the three clusters as well. Since grids in D-Stream are not allowed to move, each grid cannot re-adjust its position according to actual data inside. In some unfortunate situations, they can incorrectly join two separated clusters. In contrast, HCMstream allows the adjustment of the centroids of micro-clusters. Micro-clusters obtained from HCMstream can re-adjust their original positions toward the area where data points are more concentrated, leading to less incorrectly joining separated clusters.

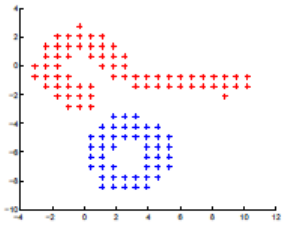
HCMstream can recognize the three clusters correctly. Table \ref{tab:vary_np} shows the clustering results when np is varied. When np was larger than 30, HCMstream failed to recognize the three clusters. This results from the fact that when the threshold for dense micro-cluster np is larger, more micro-clusters are excluded from the clustering process, leading to more fragments of clusters. When $r = 0.5$ and $np = 5$, HCMstream could find only two clusters because micro-clusters around the border of different clusters became connected with each other. However, when np was larger, these micro-clusters were excluded from the clustering process. Consequently, HCMstream could correctly recognize the three clusters when np was between 10 to 30.



(a) Clustering result from HCMstream.

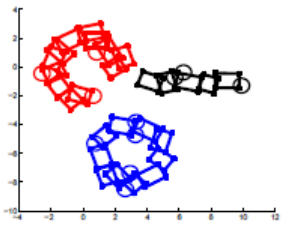


(b) Clustering result from DenStream.

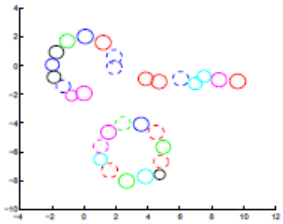


(c) Clustering result from D-Stream.

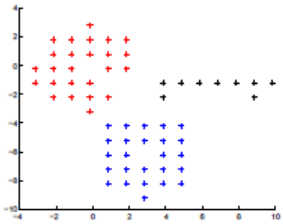
Figure 22 Clustering results of three algorithms with parameters $r = 0.35$ *and* $np = 5$.



(a) Clustering result from HCMstream.



(b) Clustering result from DenStream.



(c) Clustering result from D-Stream.

Figure 23 Clustering results of three algorithms with parameters $r = 0.5$ *and* $np = 10$.

Table 1 The comparison of clustering results of 2-dimensional synthetic data obtained from HCMstream, DenStream,and D-Stream.

$r = 0.35$, $np = 5$	HCMstream	DenStream	D-Stream
Number of micro-clusters	68	78	112
Number of clusters	3	45	2
$r = 0.5$, $np = 10$	HCMstream	DenStream	D-Stream
Number of micro-clusters	38	31	59
Number of clusters	3	22	3

Table 2 Clustering results of HCMstream with varying np, *when* $r = 0.35$ *and* 0.5.

$r = 0.35$, np	5	10	20	30
Number of micro-clusters	68	45	38	26
Number of clusters	3	3	3	6

$r = 0.5$, np	5	10	20	30	40
Number of micro-clusters	41	38	29	27	20
Number of clusters	2	3	3	3	5

Table 3 Clustering results of HCMstream with varying r.

$np = 10$, r	0.3	0.35	0.45	0.55	0.65
Number of micro-clusters	62	48	42	33	29
Number of clusters	3	3	3	3	2

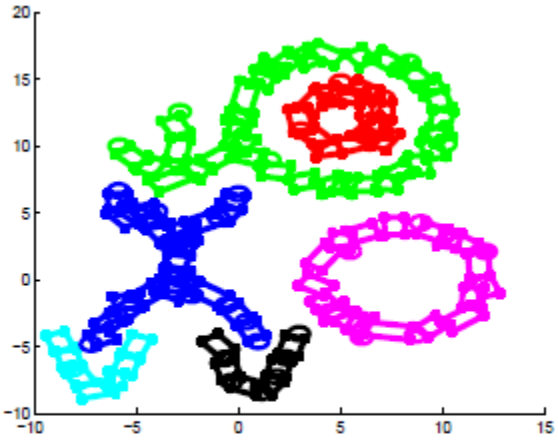
Table 3 shows the influence of r. At r =0.65, micro-clusters at the border of different clusters were so large that they become connected with each other, forming a single cluster as discussed before. As a result, HCMstream found only 2 clusters.

3.5.3 Synthetic Data Sets With Noise

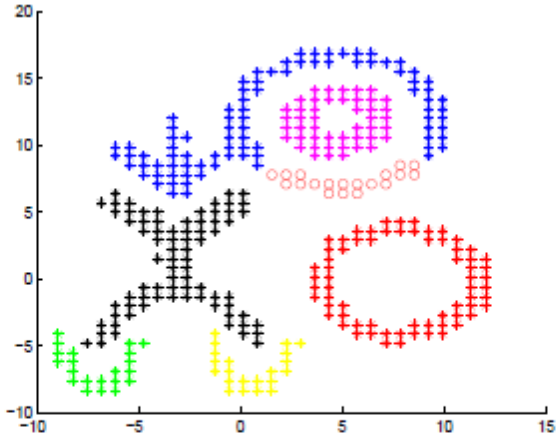
The clustering result of HCMstream is shown in *Figure 24*a with parameters $r = 0.5$ and $np = 8$. As micro-clusters of DenStream generated more than ninety final clusters, we do not present the result of DenStream here. *Figure 24Figure 24 Synthetic data set with more complicated clusters.*

b shows the clustering results from D-stream. For D-stream, r and np were set at 0.35 and 8 respectively from trial-and-error process so that the resulting clusters are as closed to the correct clusters as possible.

Figure 25 shows the numbers of micro-clusters and number of clusters produced from HCMstream and D-Stream. HCMstream yielded micro-clusters three times less than D-Stream. Moreover, while HCMstream correctly detected seven clusters, D-stream failed to combine the cluster represented by blue asterisks and the cluster represented by red circles into one cluster.



(a) Clustering results from HCM-stream with $r = 0.5$, $np = 8$.



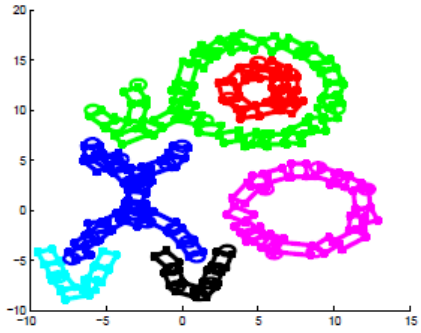
(b) Clustering results from D-stream with $r = 0.35$, $np = 8$.

Figure 24 Synthetic data set with more complicated clusters.

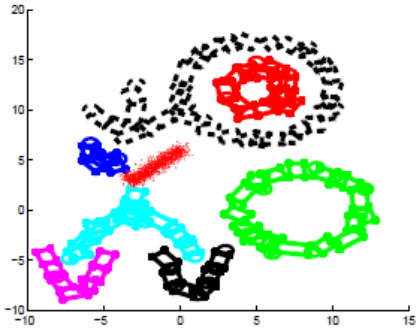
	HCMstream	D-Stream
Number of micro-clusters	132	337
Number of clusters	6	7

Figure 25 Number of micro-clusters and number of clusters produced by HCMstream and D-stream.

The process of removing destructors in HCMstream was tested with this data set. The clustering result prior to the removal of destructors is shown in Figure 24. After removing the destructors, the resulting final clusters are shown in Figure 24b. Note that the possessive cluster of destructors was split into two clusters, one in blue and another one in magenta.



(a) The clustering result of the given data set prior to the removal of the destructors.



(b) The clustering result after the removal of destructors. The removed points are shown in red dots. The possessive cluster of the destructors was split into two sub-cluster denoted in blue and magenta.

Figure 26 Removing data records.

3.5.4 Experiments with Real Data Sets

Four public real data sets were used as benchmarks for comparing the clustering results of HCMstream with other algorithms, two of which were KDD cup 99 and Forest cover type from UCI Machine Learning Repository [26] which have been widely used as standard validating data sets for clustering streaming data. The descriptions of the four data sets are shown below.

- KDD cup 99 is a series of TCP connections categorized as normal or other 22 attack connections. There are 23 classes, out of which three classes

represent more than ninety percent of all records. Each record consists of 42 features, of which 34 are continuous and 8 are categorical.

- Forest cover type data set is the observations of actual forest cover types in 30×30 square meter cell with independent 54 cartographic variables including 44 qualitative binary variables and 10 quantitative variables. The cover type consists of 7 classes.
- Hopkins 155 data set is a benchmark for motion segmentation [27]. The data set consists of the coordinates of tracked points of the moving objects in video frames. The tracked points of the same object are labeled as the same class. We used the sequence in the file *2T3RCTP_truth.mat* which is the sequence of three objects where one object is fixed, another is translating, and the other object and the camera are rotating. There are totally 24 frames in the data set with 470 tracked points. The coordinates are recorded in X-, Y-, and Z- axes. As a result, the data set consists of 470 records with 3×24=72 variables in each record.
- Iris data set contains 150 instances with 4 independent variables. The records are categorized into three classes of 50 points each. One class is clearly separated from the other two classes which are connected together to form a single cluster. As a result, though there are three classes in the data set, it contains only two clusters. This data set will be used in testing the function of data records removal, so that it is easy to visualize the result of the algorithm.

All features of the data sets were normalized to have value between 0 to 1. For each data set, the order of data was randomly shuffled and tested with the algorithms. The average results from ten repetitions were reported. For KDD 99 cup and Forest cover type data sets, 100,000 first records were used in the experiments. For Hopkins 155 data set, all 470 records were used.

3.5.5 Experiments on Clustering Performances

The clustering results of five algorithms, namely DenStream [9], D-Stream [12], ExCC [13], hierarchical agglomerative clustering for streaming data proposed by Tu et al. [5] which will be shortened as HAC, and K-mean clustering, were used to compared with HCMstream. HCMstream is most similar to DenStream in terms of density-based clustering concept. Furthermore, the idea of spherical micro-clustering in HCMstream was adopted from DenStream. As a result, DenStream was chosen to comparewith our algorithm to show the improvement on clustering results of our algorithm.

D-Stream was chosen because its clustering process is very similar to that of DenStream excepting that instead of using micro-clusters, D-Stream uses grids in primary local clustering. ExCC is the derivative of D-Stream. These two algorithms are essentially the same in clustering process, excepting that ExCC automatically calculates the threshold of dense grid rather than requires users to provide it as a parameter. Other algorithms that use idea totally different from HCMstream were also used for comparison. HAC uses hierarchical clustering which requires the number of clusters as an input parameter. We set it equal to the number of classes in each data set. The classic K-mean method was used as a benchmark algorithm. The number of clusters for K-mean method was set as the number of classes, same as that of HAC. The clustering results of KDD, Forest cover and Hopkins 155 are shown in Table 4 KDD cup 99 data set., Table 6, and Table 8, respectively. Note that excepting K-mean clustering, all algorithms chosen here can process arbitrary-shape clusters. Some recent algorithms such as HECES or eVQ-AMS were not used in benchmarking here because they cannot recognize non-convex clusters.

For KDD cup 99 data set in Table 4, HCMstream outperformed all compared algorithms in all four performance indices. In terms of the number of micro-clusters, it produced less micro-clusters than DenStream and D-stream. The fact that K-mean did not perform well indicated that the clusters of this data set were not in spherical shape. This implies that any clustering algorithm allowing clusters with arbitrary shape may produce better results. Note that the number of micro-clusters and the

number of clusters were not integer because the average results of ten repetitions were showed in the table.

For Forest cover type data set, HCMstream did not show obvious improvement over other algorithms in terms of performance indices. HCMstream produced less micro-clusters than D-Stream and ExCC, but almost equal to DenStream. Note that the overall performances of this data set were comparatively lower than those of the other two data sets. This indicated that the data in the same class did not form very homogeneous clusters. Rather, they scattered over several clusters leading to less degree of homogeneity.

For Hopkins 155 data set in Table 8, HCMstream outperformed other algorithms in terms of NMI, AR, RI, and HI. The number of micro-clusters of HCMstream was not obviously smaller than that of D-steam as in the two previous data sets because we set the parameter r of D-Stream at 0.35 while r in HCMstream was 0.25.

The clustering results of varying np were shown in Table 5 and Table 7. For KDD cup 99 data set, when np increased, the number of clusters did not increase, indicating that in this range, there was no cluster fragments resulting from micro-clusters in the same cluster being excluded from the clustering process as discussed earlier. For Forest cover type data set, that the numbers of micro-clusters were approximately the same as those of final clusters indicated that micro-clusters were quite separated from each other. Many clusters were formed by a single micro-cluster as seen from the fact that when np increased, the number of micro-clusters and the number of clusters went down together. As the dimensionality of Forest cover type is larger than that of KDD cup 99, its data points are more sparse, resulting in micro-clusters being more separated.

Note that although DenStream did not perform well with 2-dimensional synthetic data sets, its performances were acceptable with high dimensional data sets. This indicates that the micro-clustering method based on the variance of micro-cluster is not efficient in low-dimensional space. However, in high dimensional space, data are so sparse that the micro-clusters are not highly connected with each other

as in the case of low dimensional space. Therefore, the issue of fragmented clusters in DenStream did not cause significant drawback here.

Table 4 KDD cup 99 data set.

Performance indices	HCMstream $[r, np] = [0.35, 30]$	DenStream $[r, np] = [0.35, 30]$	D-Stream $[r, np] = [0.35, 30]$	ExCC $r = 0.35$	HAC $C = 22$	K-mean $C = 22$
NMI	0.8421	0.6836	0.6283	0.6361	0.6304	0.5977
AR	0.8278	0.4937	0.4819	0.4849	0.3913	.3563
RI	0.9198	0.7801	0.7267	0.7279	0.7398	0.7277
HI	0.8387	0.5601	0.4534	0.4558	0.4796	0.4553
number of clusters	17.8	27	9	6	22	22
number of micro-clusters	37	67	64	45	-	-

Table 5 KDD cup 99 data set varying np.

Performance indices	10	20	30	40	50	60	70	80	90	100
NMI	0.7827	0.8030	0.8151	0.8586	0.8669	0.8641	0.8669	0.8664	0.8732	0.8753
AR	0.7801	0.7973	0.8088	0.8410	0.8471	0.8490	0.8551	0.8549	0.8623	0.8653
RI	0.8961	0.9043	0.9098	0.9255	0.9284	0.9292	0.9321	0.9321	0.9354	0.9368
HI	0.7922	0.8086	0.8197	0.8510	0.8568	0.8584	0.8643	0.8642	0.8709	0.8737
number of micro-clusters	68	45	38	34	34	28	27	26	22	23
number of clusters	26	17	16	17	18	17	15	16	13	13

Table 6 Forest cover type data set.

Performance indices	HCMstream $[r, np] = [0.6, 40]$	DenStream $[r, np] = [0.6, 40]$	D-Stream $[r, np] = [0.6, 40]$	ExCC $r = 0.6$	HAC $C = 7$	K-mean $C = 7$
NMI	0.2625	0.2632	0.2542	0.2565	0.20961	0.2062
AR	0.1120	0.1143	0.1031	0.1065	0.1089	0.0595
RI	0.5541	0.5573	0.5415	0.5456	0.5090	0.5077
HI	0.1082	0.1146	0.083	0.0912	0.1120	0.0632
number of clusters	47.3	48	54	62	7	7
number of micro-clusters	48.5	49	211	238	-	-

Table 7 Forest cover type data set varying np.

Performance indices	10	20	30	40	50	60	70	80	90	100
NMI	0.2656	0.2648	0.2620	0.2619	0.2585	0.2539	0.2534	0.2533	0.2496	0.2461
AR	0.1153	0.1143	0.1120	0.1114	0.1088	0.1058	0.1051	0.1043	0.1018	0.1000
RI	0.5586	0.5571	0.5543	0.5531	0.5501	0.5464	0.5457	0.5445	0.5410	0.5393
HI	0.1172	0.1142	0.1086	0.1062	0.1003	0.0929	0.0914	0.0890	0.0819	0.0785
number of micro-clusters	65	57	53	48	43	40	38	40	33	32
number of clusters	60	56	51	48	43	39	38	39	32	32

Table 8 Hopkins 155 data set

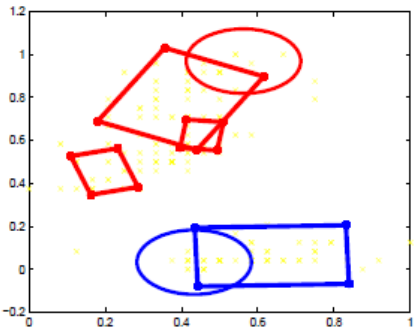
Performance indices	HCMstream $[r, np] = [0.25, 5]$	DenStream $[r, np] = [0.3, 5]$	D-Stream $[r, np] = [0.35, 5]$	ExCC $r = 0.35$	HAC $C = 3$	K-mean $C = 3$
NMI	0.7472	0.5571	0.6335	0.6219	0.5321	0.5284
AR	0.5925	0.2899	0.4979	0.4862	0.2591	0.2501
RI	0.8155	0.7384	0.7690	0.7648	0.7198	0.7164
HI	0.6309	0.4767	0.5381	0.5295	0.4395	0.4328
number of clusters	7	16	4	7	3	3
number of micro-clusters	14	20	4	7	-	-

3.5.6 Experiments on Removing Destructors

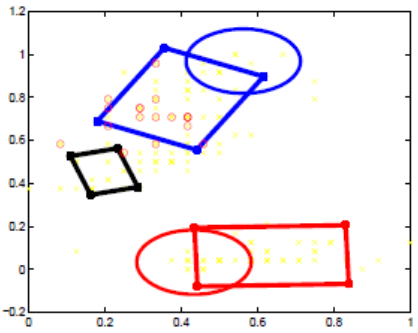
Although Iris data set contains three classes, two of which, namely Iris Versicolour and Iris Virginica, are connected to each other, usually leading to incorrect clustered points between these two classes. In this experiment, full data set was clustered first to determine how many clusters HCMstream could find. Then, we removed data points located between these two classes that caused difficulty in clustering and determined if the algorithm could discover three clusters correctly.

In removing points between the two classes, we used K-mean algorithm with $k = 3$. Mistaken points between these two classes, which were points belonging to class 2, but being mistaken as belonging to class 3 and vice versa, were marked as destructors. As a result, our data set consisted of 150 original data, with order randomly reshuffled, being set as constructors and 18 destructors obtained from this process, totally 168 points.

After clustering 150 constructors, HCMstream discovered two clusters as shown in Figure 27a which shows the resulting clusters of 150 original data points as two dimensional plot projected on the second and the fourth attributes of Iris data. The algorithm yielded two clusters shown in blue and red. When the destructors were processed, the original blue cluster was split into two clusters depicted in blue and black as shown in b.



(a) HCMstream found two clusters from original data set.



(b) Clustering result when destructors depicted in red circles were removed. The number of resulting clusters became three.

Figure 27 Testing removal of data with Iris data set.

When orders of constructors and destructors were randomly rearranged, at which constructors appeared before destructors, the average performance indices of ten repetitions are shown in Table 9 in comparison with those of D-Stream and K-means. Note that the order of incoming data can affect the clustering results in HCMstream which uses micro-clusters to capture the local statistics of data. When the size of data set is small, with different orders of incoming data, the pattern of micro-clusters can vary, leading to different clustering results. The effect of different orders of data set is mitigated when the volume of data set is large because local statistics of data set become more stable due to the law of large numbers, leading to similar pattern of micro-clusters from the same stream in different orders.

In this case, since the size of data is relatively low (150), the clustering results varied with different orders of the incoming data. Even though the algorithm could not correctly find 3 clusters in all repetitions, the performance indices of HCMstream were slightly higher than those of K-means algorithm. Note that due to small data size, K-means algorithm could not yield consistent clustering results as well, as seen from the fact that despite removing the incorrectly clustered points based on K-means, the algorithm still could not yield one hundred percent correct results. For D-Stream, although the performance indices were highest, it incorrectly found 4

clusters due to the fragments of the cluster representing data points in class Iris Virginica.

Table 9 Comparison of performance indices from the three algorithms on Iris data set after removing destructors.

	nmi	AR	RI	HI	number of clusters
HCMstream	0.9046	0.9181	0.9585	0.9171	3.3
D-Stream	0.9420	0.9590	0.9815	0.9630	4
K-means	0.8541	0.8168	0.9129	0.8259	3

Chapter 4
Proposed algorithm 2: LLDstream

4.1 Unsupervised localized linear discriminant analysis (ULLDA)

The process of ULLDA is summarized in Algorithm 3. ULLDA requires three inputs, namely a reference point $\mathbf{x}$, a set of the centers of all clusters $\mathfrak{C}$, and the number of nearest clusters nc. It returns the projection matrix $\mathbf{V}$.
At Line 1, we initialize an empty set $\mathcal{NC}$ and let $\mathbf{V}$ be an identity matrix. As a result, for the first iteration, at Line 3 we calculate the distances of $\mathbf{x}$ and the centers of the clusters in the full-dimensional space.

The set of nearest clusters $\mathcal{N}n(nc)$ is determined in Line 4. As initially $\mathcal{NC}$ is an empty set, the algorithm would proceeds to Lines 6 and 7, resulting in the center matrix $\widetilde{\mathbf{M}}$ and a nonempty set $\mathcal{NC}$ for the second iteration. Next, the algorithm starts at Line 2 finding the projection matrix $\mathbf{V}$. From the second iteration onward, the distances calculated in Line 3 would be the projected distances on $\mathbf{V}$. The steps from Lines 2 to 8 are repeated until $\mathcal{N}n(nc)^{new}$ is the subset of $\mathcal{NC}$.

Algorithm 3: ULLDA	
1	Initialize an empty set $\mathcal{NC} = \emptyset$. Let $\mathbf{V}$ be an identity matrix. Go to Line 3.
2	Perform the SVD of $\widetilde{\mathbf{M}}$ as $\widetilde{\mathbf{M}} = \mathbf{VSU}^T$.
3	Compute the projected distance between $\mathbf{x}$ and all centers of clusters $\mathbf{c}_i$ as $\mathbf{x}'=\mathbf{V}^T\mathbf{x}$ and $\mathbf{c}_i'=\mathbf{V}^T\mathbf{c}_i$.
4	Create $\mathcal{N}n(nc)^{new} = \{id(1), \dots, id(nc)\}$, where $id(i)$ is the index of the i^{th} nearest cluster of $\mathbf{x}$.
5	If $\mathcal{N}n(nc)^{new}$ is not a subset of $\mathcal{NC}$
6	$\mathcal{NC} = \mathcal{NC} \cup \mathcal{N}n(nc)^{new}$
7	Obtain $\widetilde{\mathbf{M}}$ from $\mathcal{NC}$.
8	Go back to Line 2
9	EndIf
10	Return

ULLDA chooses relevant dimensions that maximally separate nearby clusters with respect to reference point $\mathbf{x}$. However, in a new projected subspace, the set of nc nearest clusters does not necessarily remain the same. As a result, in each iteration, the new nearest clusters are added into $\mathcal{NC}$, leading to new center vectors being added into $\widetilde{\mathbf{M}}$. In the last iteration, when no new neighboring clusters are added into $\mathcal{NC}$, the center vectors of all relevant nearby clusters are included in $\widetilde{\mathbf{M}}$. The subspace spanned by the columns of $\widetilde{\mathbf{M}}$ is the resulting projected subspace.

4.2 The complexity of ULLDA

ULLDA consists of the iteration of two steps, namely finding nc nearest clusters in Line 4 and performing SVD of $\widetilde{\mathbf{M}}$ in Line 2. Let $|\mathfrak{C}|$ be the number of clusters, finding distance between $\mathbf{x}$ and all clusters and choose the nc nearest clusters take $\mathbf{O}(nc|\mathfrak{C}|)$. The size of $\widetilde{\mathbf{M}}$ is at most $d \times |\mathfrak{C}|$ where d is the number of

features. Usually for high dimensional data $d > |C|$, leading to $\mathbf{O}(|\mathbb{C}|^2 d)$ for calculating SVD of $\widetilde{\mathbf{M}}$ [28]. As a result, in each iteration, the algorithm takes

$$\mathbf{O}(|\mathbb{C}|^2 d + nc|\mathbb{C}|)$$

Since the term $nc|\mathbb{C}| \ll |\mathbb{C}|^2 d$, equation $\mathbf{O}(|\mathbb{C}|^2 d + nc|\mathbb{C}|)$ becomes $\mathbf{O}(|\mathbb{C}|^2 d)$ With i iterations (which is usually unknown), the computation complexity of ULLDA for one point is

$$\mathbf{O}(i|\mathbb{C}|^2 d)$$

4.3 LLDstream

In this study, we use a spherical micro-cluster defined by two attributes, namely the center of a micro-cluster and the number of points assigned to that micro-cluster as described in **Definition 1** as follow.

Definition 1: A micro-cluster denoted by $\mathcal{M} = (\mathbf{c}, N)$ is a cluster formed by a sphere with fixed radius r whose center is at $\mathbf{c}$. N is the number of data points assigned to the micro-cluster.

The radius of micro-clusters, r, is provided as an input parameter by the user. LLDstream uses a fixed radius sphere as a local model for grouping data points. Note that our objective is to assign an incoming datum to a micro-cluster defined by a sphere with fixed boundary, not to a cluster of data points whose boundary cannot be defined clearly. This allows us to disregard the actual distribution of data points inside the micro-clusters and treat each micro-cluster as a spherical cluster. In performing ULLDA, we can neglect S_w and determine the projection matrix $\mathbf{V}$ directly from $\widetilde{\mathbf{M}}$.

The algorithm of LLDstream [29] in online phase is shown in Algorithm 4. When a new datum $\mathbf{x}$ arrives, ULLDA is performed at the location of $\mathbf{x}$ with micro-clusters being regarded as local clusters. Since ULLDA requires the convergence of $\mathcal{N}\mathbf{n}$, the obtained projection vectors do not depend much on the initial value of

nearest neighbors nc. As a result, nc is set constant at 10 in this study. If the total number of clusters is less than 10, no dimension reduction is performed. The datum will be assigned to a micro-cluster in the original feature space.

Let $\mathfrak{C} = \{\mathcal{M}_1, \ldots \mathcal{M}_{|\mathfrak{C}|}\}$ be a set of micro-cluster where $\mathcal{M}_i$ denotes the i^{th} micro-cluster defined by $(\mathbf{c}_i, N_i)$ The projection matrix $\mathbf{V}$ is obtained from ULLDA. Then, the projected distance between $\mathbf{x}$ and $\mathcal{M}_i$ on $\mathbf{V}$, denoted by $d_{\mathbf{V}}(\mathbf{x}, \mathcal{M}_i)$, is calculated by

$$d_{\mathbf{V}}(\mathbf{x}, \mathcal{M}_i) = ||\mathbf{V}^T(\mathbf{c}_i - \mathbf{x})||$$

We determine whether point $\mathbf{x}$ is assigned to an existing micro-cluster from the following condition. Let $K = \underset{i}{\mathbf{argmin}}\ d_{\mathbf{V}}(\mathbf{x}, \mathcal{M}_i)$, then the closest micro-cluster of point $\mathbf{x}$ is denoted by $\mathcal{M}_K$. Point $\mathbf{x}$ is assigned to $\mathcal{M}_K$ when the projected distance between
point $\mathbf{x}$ and the center of $\mathcal{M}_K$ is less than r, as

$$d_{\mathbf{V}}(\mathbf{x}, \mathcal{M}_K) \leq r$$

where r is the radius of the micro-cluster.

If $\mathbf{x}$ is assigned to $\mathcal{M}_K$, the parameters of $\mathcal{M}_K$ are updated in Line 8 as

$$\mathbf{c}_K^{new} = \frac{\mathbf{c}_K N_K + \mathbf{x}}{N_K + 1}$$
$$N_K^{new} = N_K + 1.$$

If $\mathbf{x}$ cannot be assigned to $\mathcal{M}_K$, a new micro-cluster $\mathcal{M}_{|\mathfrak{C}|+1}$ is created with

$$\mathbf{c}_{|\mathfrak{C}|+1} = \mathbf{x}$$
$$N_{|\mathfrak{C}|+1} = 1$$

Initially, there is no micro-cluster in the feature space; in other words, the set of micro-clusters $\mathfrak{C}$ is empty. When a new data point $\mathbf{x}$ arrives, there is no nearest micro-clusters, hence, the algorithm performs the assignment in the original space by setting $\mathbf{V}$ as an identity matrix. A new micro-cluster is generated in Line 10.
The algorithm would keep generating new micro-clusters until there are sufficient micro-clusters to perform ULLDA. Then, the algorithm would start performing ULLDA at Line 2 on the next incoming datum. On the other hand, the user can choose to pre-generate initial micro-clusters using DenStream by setting aside some data points for generating the initial micro-clusters.

Algorithm 4: LLDstream Algorithm: online phase

1 If $|\mathfrak{C}| \geq 10$
2 Perform ULLDA at $\mathbf{x}$ as $\mathbf{V}$ =ULLDA($\mathbf{x}$, $\mathfrak{C}$, 10)
3 Else
4 Assign $\mathbf{x}$ to a micro-cluster in the full dimensional space by setting $\mathbf{V}$ as identity matrix.
5 EndIf
6 Calculate the projected distance between $\mathbf{x}$ and all micro-clusters in the subspace spanned by $\mathbf{V}$ using Equation $d_{\mathbf{V}}(\mathbf{x},\mathcal{M}_i) = \left|\left|\mathbf{V}^T(\mathbf{c}_i - \mathbf{x})\right|\right|$.
7 If $d_{\mathbf{V}}(\mathbf{x},\mathcal{M}_K) \leq r$
8 Assign $\mathbf{x}$ to $\mathcal{M}_K$ and update the parameters of $\mathcal{M}_K$ according to Equations $\mathbf{c}_K^{new} = \frac{\mathbf{c}_K N_K + \mathbf{x}}{N_K + 1}$ and $N_K^{new} = N_K + 1$.
9 Else
10 Create a new micro-cluster $\mathcal{M}_{|\mathfrak{C}|+1}$ by $\mathbf{c}_{|\mathfrak{C}|+1} = \mathbf{x}$, and $N_{|\mathfrak{C}|+1} =$ N
11 $\mathfrak{C} = \mathfrak{C} \cup \mathcal{M}_{|\mathfrak{C}|+1}$
12 End

When the request of final clusters from the user arrives, the algorithm operates in the offline phase. The density threshold np is provided by the user. Micro-clusters with the number of assigned data greater than np are taken into consideration in the offline phase while those with the number of data inside less than np are regarded as outliers.

LLDstream generates a cluster from a set of connected micro-clusters. Two micro-clusters $\mathcal{M}_i$ and $\mathcal{M}_j$ are connected if

$$||\mathbf{c}_i - \mathbf{c}_j|| \leq 2r$$

4.4 The complexity of LLDstream

In the online phase, the bottleneck of LLDstream is in ULLDA which requires $\mathbf{O}(i|\mathfrak{C}|^2 d)$ for one data point. In the offline phase, determining pairwise distances among micro-clusters takes $\mathbf{O}(|\mathfrak{C}|^2)$. For large data set with the numbers of other features much less than the number of incoming data n, the complexity of LLDstream becomes $\mathbf{O}(n)$.

4.5 Experimental results

All data sets used in this study were obtained from UCI Machine Learning Repository [26]. Two sets of experiments were conducted to evaluate the performance of LLDstream in comparison with other algorithms.

The first set consisted of experiments with benchmark data sets for streaming data, including KDD cup 99, NSL-KDD, and Forest cover type data sets which have been widely used for evaluating stream data clustering algorithms in many previous studies such as [9], [30], and [14]. For KDD cup 99 data set, we used data from the file *kddcup.data_10_.gz*, which consists of 494,021 data instances with 41 features. Among 7 symbolic features out of all 41 features, we kept three binary-value symbolic features while the other four symbolic features which cannot be represented by binary data were removed. As a result, there are 38 features in this data set. NSL-KDD data set [31], which can be obtained from [31], is the refined subset of KDD cup 99, in which redundant records are removed. Moreover, the number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set, resulting in a more proportionately distributed data set. NSL-KDD consists of 125,973 instances with the same features as those in KDD data set. Each instance is labeled as either normal or anomaly. The preprocessing of NSL-KDD is performed in the same way as that of KDD cup 99. 38 out of 41 features were used in the experiments. For Forest cover type data set, out of 54 features, 10 quantitative features were used in the clustering while other 44 symbolic features were removed from the data set. In this set of experiments, state-of-the-art algorithms for clustering stream data, including

DenStream [9], HDDStream [30] and HPStream [14], were used to compare their clustering performances with the proposed algorithm.

In the second set of experiments, we evaluated the performance of LLDstream in comparison with other clustering algorithms for non-streaming data, which used the whole data sets in clustering process, in contrast to the one-pass-and-throw-away clustering of LLDstream.

4.5.1 Comparison with algorithms for streaming data

In this set of experiments, to compare the clustering performance of the algorithms without the effect of fading data, the fading function of the micro-clusters was excluded from the algorithms. To determine the dynamic performance in clustering evolving stream, the sliding window model was adopted as in [7]. In this set of experiments, the parameters of LLDstream were set at r=0.1, np=15 for all data sets.

In KDD cup 99 data set, the parameters of DenStream were set by trial-and-error, then we chose the parameters yielding the best indices which were $\varepsilon = 0.5$ and $\beta\mu = 20$; those of HDDStream were set at $(\varepsilon, \beta, \mu, \pi) = (0.2,5,30)$ according to [30], while HPStream used *k*=23, equal to the number of classes of the data set. In NSL-KDD data set, all parameters were set at the same values as those in KDD cup 99 data set, excepting the number of clusters in HPStream which was set at 2 according to the number of classes in the data set. In the experiments with Forest cover type data set, the parameters of HDDStream were $(\varepsilon, \beta, \mu, \pi) = (0.2,5,8)$ according to [30] and the number of clusters in HPStream was 7. The parameters of DenStream were set at $\varepsilon = 0.1$ and $\beta\mu = 10$.

Figure 28, Figure 29, and Figure 30 show the plots of four performance indices, namely NMI, AR, RI and HI for KDD cup 99, NSL-KDD, and Forest cover type data sets respectively. The time horizon for KDD cup 99 and NSL-KDD data sets was at 10,000 data instances, while that of Forest cover type data set was at 2,000. The reason we used the smaller window size in Forest cover type is that the pattern of clusters in this data set change over time more rapidly than that of KDD cup 99. If

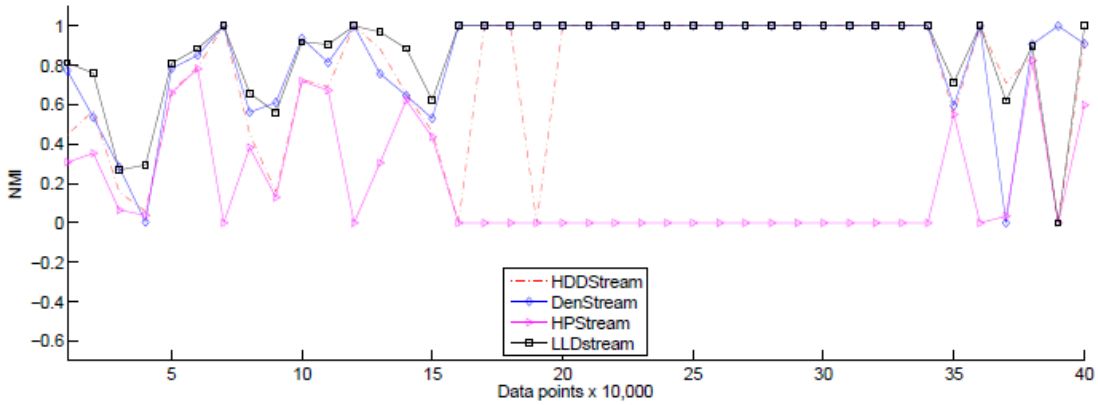
the window size is too large, the performance indices of the algorithms would be very low. The performance indices were computed when the last datum of each window arrived. Table 10, Table 11 and Table 12 show the average clustering performances over the whole length of KDD cup 99, NSL-KDD, and Forest cover type data sets respectively.

KDD cup 99 data set The comparison plots of performance indices are shown in Figure 28. The window size was set at 10,000 points. LLDstream outperforms other algorithms in all indices. Data during the 200,000 th to 350,000 instances belong to a single class. To avoid division by zero in NMI and AR, we set the indices to be 1 when the clustering result is identical to the class label. LLDstream and other algorithms which adopt density-based clustering can detect the number of cluster correctly as 1, leading to all indices reaching the value of 1. The number of clusters in HPStream was predefined at the number of classes, leading to the relatively low clustering performance during this period of time.

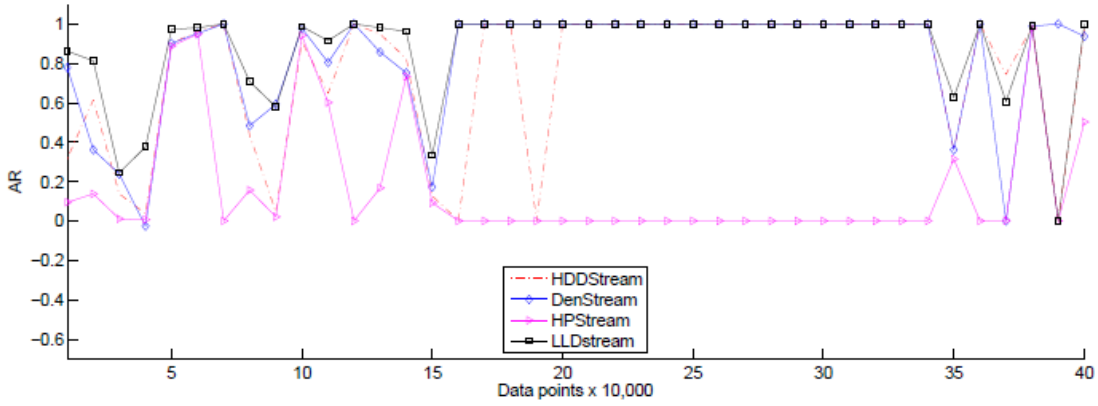
For the comparative average performance indices shown in Table 10, LLDstream outperforms the comparing algorithms. Moreover, the performance indices of the density-based clustering algorithms, namely, LLDstream, DenStream, and HDDStream, are better than those of HPStream due to the fact that the number of clusters in HPStream was set at constant for the whole clustering process, while the actual number of clusters can vary during each time horizon. Since the density-based clustering algorithms do not assume a fixed number of clusters, they can respond to the varying number of clusters in different intervals more accurately.

Table 10 Comparative average performance indices with KDD cup 99 data set.

Algorithm	NMI	AR	RI	HI
LLDstream	0.8648	0.8736	0.9715	0.9431
DenStream	0.8377	0.8284	0.9463	0.8926
HDDStream	0.8188	0.7497	0.8992	0.7984
HPStream	0.1856	0.1553	0.5986	0.1972



(a) Normalized mutual information(NMI).



(b) Adjusted Rand index(AR).

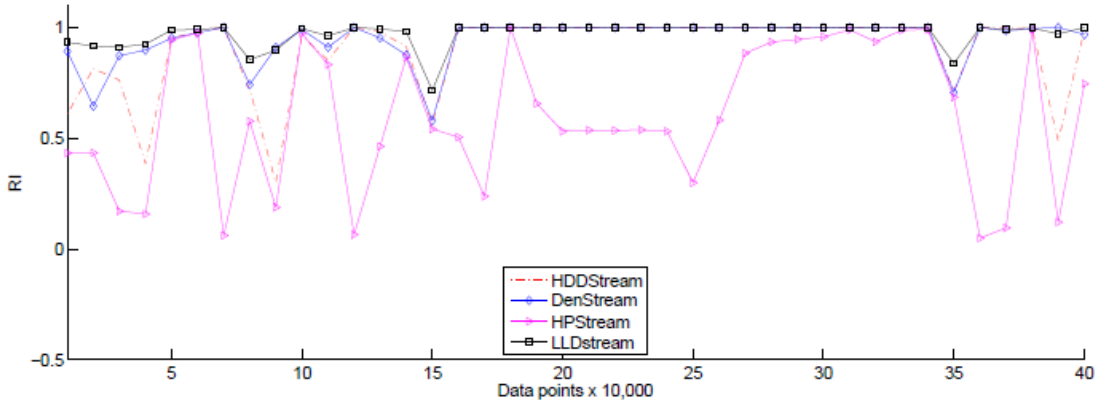


Figure 28 Comparison of clustering performance indices for KDD cup 99 data set with the time horizon of 10,000.

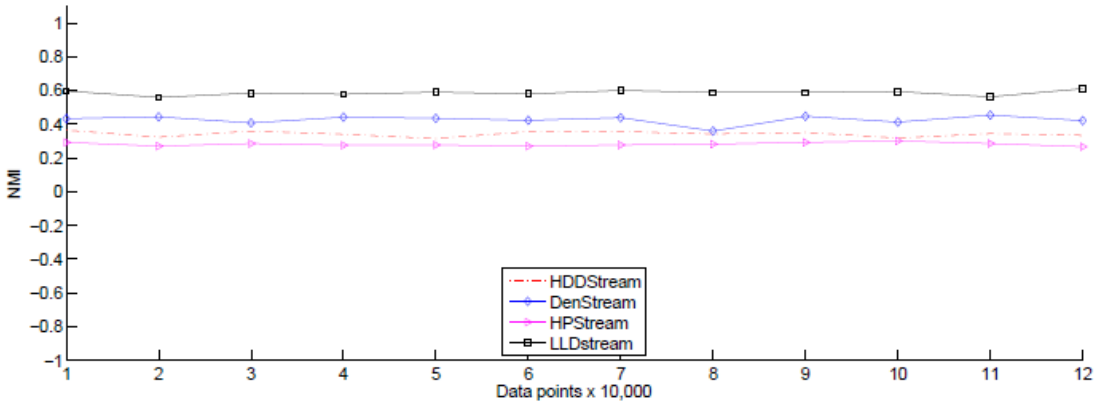
NSL-KDD data set For NSL-KDD data set, LLDstream clearly outperforms the compared algorithms. Without the over-representative of some classes as discussed in [31], the average performance indices of NSL-KDD are lower than those of the

original KDD cup 99. This results from the fact that the performance indices of all algorithms in NSL-KDD were relatively consistent for the whole time horizons, unlike the performance indices from KDD cup 99 which become 1 during the 200,000 th to 350,000 th instances. Compared to Figure 28, the plots in Figure 29 show more uniform performance indices over the whole stream due to the removal of the redundant records and the re-arrangement of the point order.

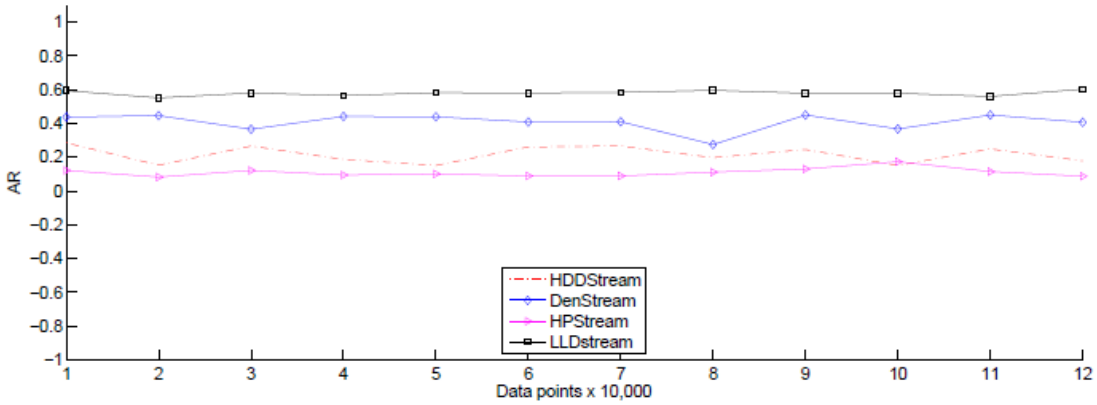
Note that in Table 10, LLDstream slightly outperforms DenStream. However, in Table 11, LLDstream outperforms DenStream by more than 10 percent improvement on RI, while on NMI, AR, and HI, LLDstream makes almost 30 percent improvement. This results from the fact that in KDD cup 99 data set, during the 200,000 th to 350,000 th instances, both DenStream and LLDstream can correctly detect one cluster, leading to the same value of all performance indices as 1 for this interval. Since this interval covers almost half of the clustering, the improvement of LLDstream becomes less distinct in the average performance indices. On the other hand, in NSL-KDD data set, there is no such an interval. As a result, the average performance indices can more clearly reflect the improvement that LLDstream made.

Table 11 Comparative average performance indices with NSL-KDD data set.

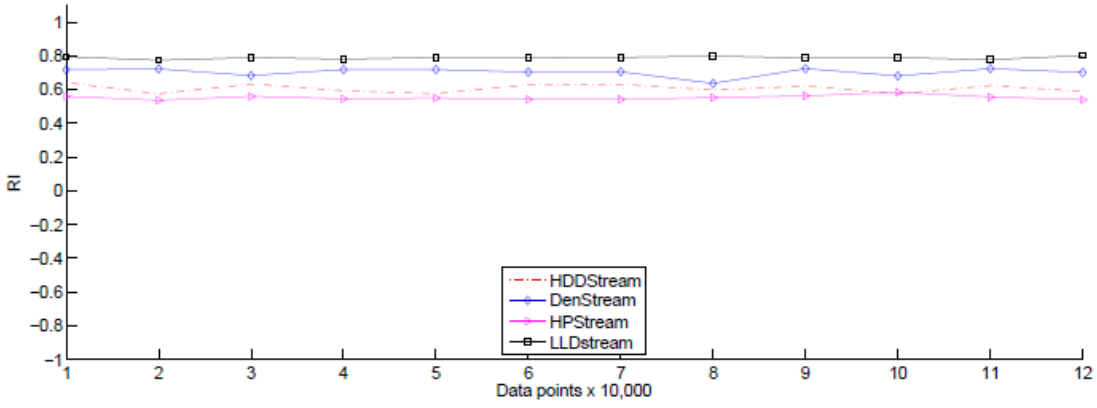
Algorithm	NMI	AR	RI	HI
LLDstream	0.5865	0.5793	0.7889	0.5778
DenStream	0.4274	0.4077	0.7033	0.4067
HDDStream	0.3172	0.1588	0.5790	0.1579
HPStream	0.3044	0.2102	0.5990	0.1981



(a) Normalized mutual information(NMI).



(b) Adjusted Rand index(AR).



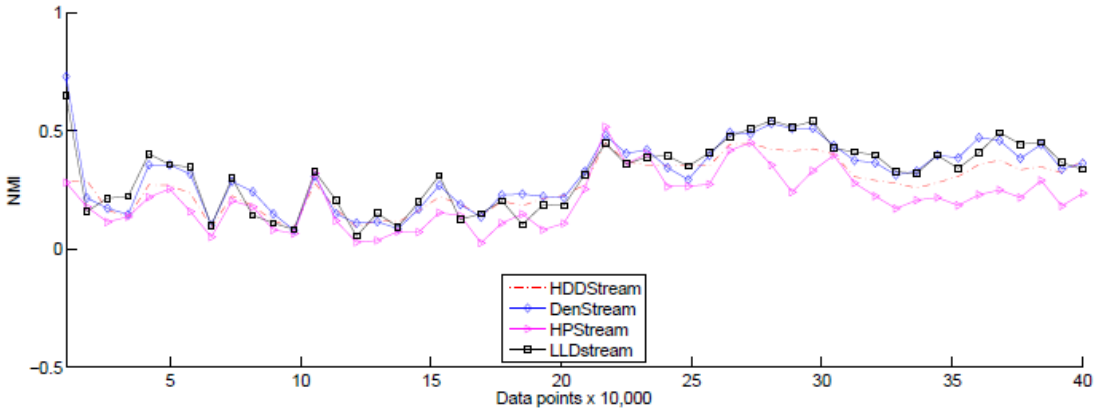
(c) Rand index(RI).

Figure 29 Comparison of clustering performance indices for NSL-KDD data set with the time horizon of 10,000.

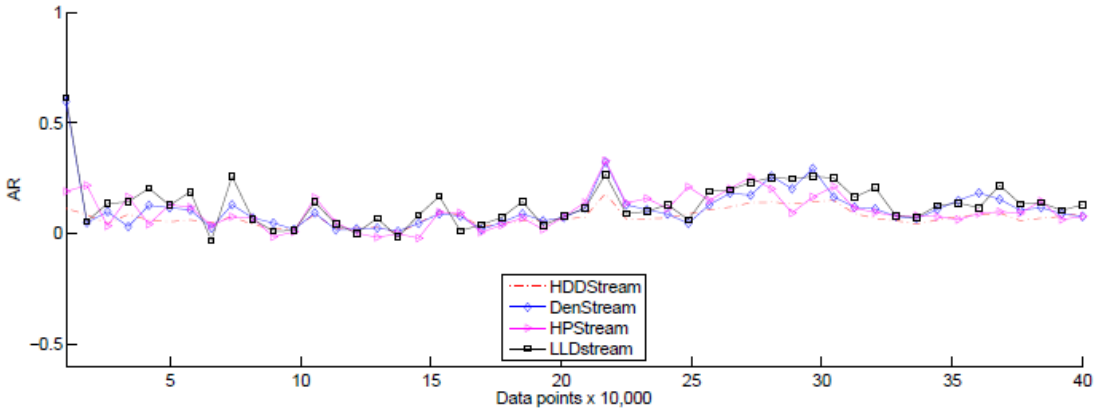
Forest cover type data set LLDstream did not outperform the compared algorithms much. The relatively low performance indices of all algorithms indicate that data in the same class of this data set do not form very clean clusters. Rather, each cluster consists of data from several class, leading to the low performance indices of all algorithms. Moreover, since the number of attributes in this data set is not very high, the benefit of dimension reduction in this data set might not be very obvious, as seen from the fact that the performance indices of LLDstream and DenStream are not significantly different.

Table 12 Comparative average performance indices with Forest cover type data set.

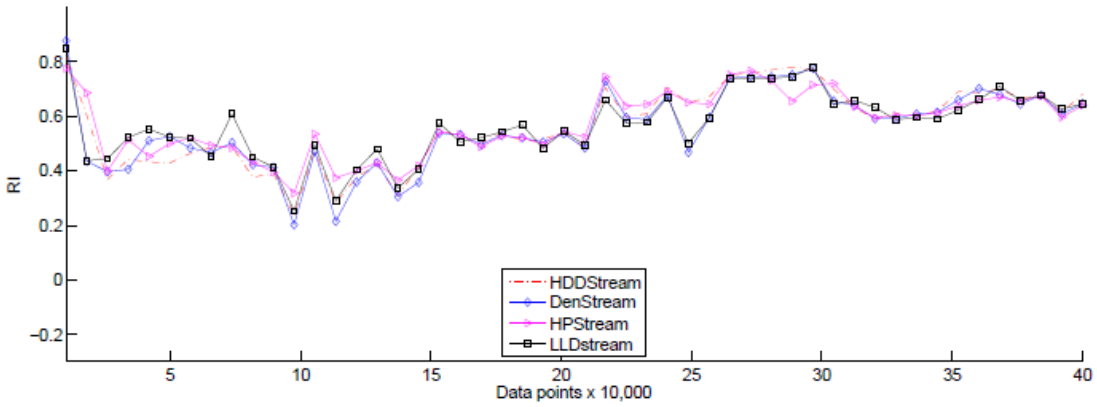
Algorithm	NMI	AR	RI	HI
LLDstream	0.3064	0.1284	0.5667	0.1333
DenStream	0.3052	0.1021	0.5531	0.1063
HDDStream	0.2690	0.0746	0.5610	0.1220
HPStream	0.2100	0.0979	0.5667	0.1294



(a) Normalized mutual information(NMI).



(b) Adjusted Rand index(AR).



(c) Rand index(RI).

Figure 30 Comparison of clustering performance indices for Forest cover type data set with the time horizon of 2,000.

4.5.2 Comparison with algorithms for non-streaming data

The performance of LLDstream was compared with state-of-the-art algorithms for non-streaming data including DBSCAN [19], SNN [32], and PreDeCon [33]. In this set of experiments, we compared both clustering indices as well as computation time among these algorithms.

As all algorithms are based on DBSCAN, DBSCAN was chosen as a based-line algorithm. Two parameters of DBSCAN are required from the user, including the neighborhood radius ε and the minimum number of points in ε -neighborhood $MinPts$.

SNN is designed to handle high dimensional data by using share nearest neighbor for measuring similarity rather than some primary distance, such as Euclidean distance. Using secondary similarity measures based on share nearest neighbor improves the robustness of the algorithm, thus reducing the effect of irrelevant attributes [34]. The algorithm requires three parameters, neighborhood list size k, SNN radius Eps, and SNN density $MinPts$.

PreDeCon is a subspace clustering algorithm designed to cope with high dimensional data by using weighted similarity measure. In calculating distance, weighting coefficient of each feature is determined by the variance of ε -neighborhood. Four parameters are required from the user including the number of preference dimension λ, the variance threshold δ, as well as the two parameters of DBSCAN, ε and $MinPts$.

The parameters in these algorithms were set by trial-and-error process, then the best results were reported. All features of the data sets were normalized to have value between 0 to 1. For LLDstream, the orders of each data set were randomly shuffled and tested with the algorithm. The average results of 10 repetitions were reported. For other compared algorithms, since their clustering is not one-pass-and-throw-away method, the clustering results would remain the same regardless of the order of the incoming data. Table 13,

Table 14, and Table 15 show the comparison of clustering performance indices from the compared algorithms. The parameters of LLDstream, SNN, DBSCAN, and PreDeCon are represented as (r, np), $(k, Eps, MinPts)$, $(\varepsilon, MinPts)$, and

$(\lambda, \delta, \varepsilon, MinPts)$ respectively. We used the modules of SNN, DBSCAN, and PreDecon implemented in ELKI platform [35].

Table 13 Comparative performance indices with Image segmentation data set.

Algorithm	Parameters	NMI	AR	RI	HI
LLDstream	(0.15,20)	0.8050	0.6580	0.8951	0.7901
SNN	(170,140,20)	0.6939	0.5520	0.8887	0.7774
DBSCAN	(0.15,5)	0.6393	0.4386	0.8730	0.7461
PreDeCon	(0.4,6,0.01,15)	0.5926	0.3754	0.8606	0.7212

Table 14 Comparative performance indices with Multiple features data set.

Algorithm	Parameters	NMI	AR	RI	HI
LLDstream	(2.75,2)	0.7956	0.5802	0.8965	0.7930
SNN	(190,150,30)	0.6589	0.3205	0.8124	0.6248
DBSCAN	(5.7,4)	0.7013	0.4473	0.8739	0.7478
PreDeCon	(25,7,12,649)	0.6977	0.4526	0.8639	0.7278

Table 15 Comparative performance indices with Pen digits data set.

Algorithm	Parameters	NMI	AR	RI	HI
LLDstream	(0.25,10)	0.9061	0.8571	0.9682	0.9364
SNN	(150,100,30)	0.8156	0.7658	0.9598	0.9196
DBSCAN	(0.15,5)	0.7112	0.5408	0.9113	0.8226
PreDeCon	(1.6,20,1.0,16)	0.6941	0.4949	0.8779	0.7557

The comparative tables show that LLDstream outperformed other algorithms in term of clustering performance indices, despite its one-pass-and-throw-away clustering scheme. These results confirm that LLDstream can also perform well in the non-streaming environment. In Landsat satellite data set, LLDstream, PreDeCon

and SNN achieved relatively higher performance than DBSCAN. This indicates that for this data set, algorithms that can exclude irrelevant features can operate more efficiently. Although PreDeCon and LLDstream both perform clustering in reduced dimension subspace, LLDstream outperformed PreDeCon in all data sets, supporting the argument that LDA subspace used in LLDstream is more efficient than the axis-parallel subspace employed in PreDeCon.

Note that although the parameters ε and $MinPts$ of DBSCAN might seem similar to the parameters r and np of LLDstream, they are quite different. ε in DBSCAN indicates the distance in the original feature space while r in LLDstream refers to the distance in the projected subspace. If we use the same values of parameters, the clustering results from the two algorithms can be totally different. Consequently, as seen from the comparison tables, the values of the parameters for the best performance indices of the two algorithms are different for all data sets.

Chapter 5
CONCLUSION

This dissertation presents two algorithms for clustering streaming data, namely HCMstream and LLDstream. The two algorithms adopt density-based clustering as the method in clustering incoming data while including incremental clustering to allow the algorithms to process the incoming data in one-pass fashion. Basically, one-pass clustering collect some statistics of the incoming data while discard the raw data in order to save memory space. The proposed two algorithms address differing constraints in order to deal with some different situations found in real applications.

HCMstream is designed to optimize memory storage by merging micro-clusters together to form larger cylindrical micro-clusters. With cylindrical shape micro-cluster, the algorithm can maintain the compactness of the small micro-cluster while reduce the number of the micro-cluster, thus reducing the memory storage and the computation effort.

Moreover, HCMstream can process unscheduled record removal, the feature of which most existing clustering algorithms for streaming data do not have. Processing unscheduled record removals allows the user to deal with more variety of data sets, such as the bank accounts which older data records are similarly important as the new data records.

From the experimental results, we find that the algorithm generates less micro-clusters compared with other clustering algorithm for stream data. However, due to merging process, the algorithm requires longer computation time than DenStream.

LLDstream is designed to deal with high-dimensional data sets. It incorporates dimension reduction process into clustering framework. The clustering process is performed in LDA subspace instead of the original feature space. Clustering data sets in LDA space, instead of the original space, allows LLDstream to process data with

more efficiency in the sense that irrelevant features which are not parallel to axis can be excluded from the computation.

We performed two sets of experiments. In the first set, the clustering performance of LLDstream was compared with other clustering algorithms for streaming data. The comparative experimental results show that the algorithm outperformed other existing algorithms. In the second set of experiments, LLDstream was compared with other traditional clustering algorithms that use conventional data sets rather than streaming data sets. The results also confirm that LLDstream yields good clustering results with less computation time.

However, in some data sets, LLDstream might not perform as efficient as traditional clustering algorithms. For example, LLDstream is not very efficient when process data sets having small data instances with large features. In this situation, LLDstream would take longer computation time due to its bottle neck in performing SVD.

REFERENCES

1. Yogita, Y. and D. Toshniwal, *Clustering techniques for streaming data- a survey*, in *IACC*. 2013. p. 951-–956.
2. Amini, A., Y.W. Teh, and H. Saboohi, *On density-based data streams clustering algorithms: a survey.* Journal of Computer Science and Technology, 2013. **29**: p. 116-141.
3. Guha, S., et al., *Clustering Data Streams*, in *Proceedings of the 41st IEEE FOCS Conference*. 2000, IEEE Computer Society: Washington, DC, USA. p. 359-366.
4. Wattanakitrungroj, N. and C. Lursinsap, *Memory-less unsupervised clustering for data streaming by versatile ellipsoidal function*, in *CIKM'11*. 2011. p. 967-–971.
5. Tu, Q., et al., *Density-based Hierarchical Clustering for Streaming Data.* Pattern Recognition Letters, 2012. **33**: p. 641-645.
6. Aggarwal, C.C., et al., *A framework for clustering evolving data streams*, in *VLDB*. 2003. p. 81-–92.
7. Rehman, M.Z., et al., *Hyper-ellipsoidal clustering technique for evolving data stream.* Knowledge-Based Systems, 2014. **70**: p. 3-14.
8. Lughofer, E. and M. Sayed-Mouchaweh, *Autonomous data stream clustering implementing split-and-merge concepts- Towards a plug-and-play approach.* Information Science, 2015. **304**: p. 54-79.
9. Cao, F., et al., *Density-based clustering over an evolving data stream with noise*, in *Proceedings of the 6th SIAM International Conference on Data Mining (SDM)*. 2006. p. 328-–339.
10. Ester, M., et al., *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, in *KDDM*. 1996. p. 226-–231.
11. Zhou, A.Y., et al., *Tracking cluster in evolving data streams over sliding windows.* Knowledge and Information Systems, 2008. **15**: p. 181-214.
12. Tu, L. and Y. Chen, *Stream Data Clustering Based on Grid Density and Attraction.* ACM Trans. Knowl. Discov. Data, 2009. **3**(3): p. 12:1-12:27.

13. Bhatnagar, V., S. Kaur, and S. Chakravarthy, *Clustering Data Streams Using Grid-Based Synopsis.* Knowledge Information System, 2013. **15**: p. 1-–26.
14. Aggarwal, C.C., et al., *A framework for projected clustering of high dimensional data streams*, in *VLDB*. 2004. p. 852-–863.
15. Forestiero, A., C. Pizzuti, and G. Spezzano, *A single pass algorithm for clustering evolving data streams based on swarm intelligence.* Data Mining and Knowledge Discovery, 2013. **26**: p. 1-26.
16. Li, W., et al., *Density-Based Clustering of Data Streams at Multiple Resolutions.* ACM Transaction on Knowledge discovery from Data, 2009. **3**: p. 1-28.
17. Wang, C., et al., *SVStream: A Support Vector-Based Algorithm for Clustering Data Streams.* IEEE Transaction on Knowledge and Data Engineering, 2013. **25**: p. 1410-1423.
18. Luhr, S. and M. Lasarescu, *Incremental clustering of dynamic data streams using connectivity based representative points.* Data and Knowldege Engineering, 2009. **68**: p. 1-27.
19. Ester, M., et al., *Incremental Clustering for Mining in a Data Warehousing Environment*, in *VLDB*. 1998. p. 967-–971.
20. Tang, H., T. Fang, and P.-F. Shi, *Rapid and Brief Communication: Laplacian Linear Discriminant Analysis.* Pattern Recognition, 2006. **39**(1): p. 136-139.
21. Manning, C.D., P. Raghavan, and H. Schutze, *Introduction to Information Retrieval (1st Ed.)*. 2008, New York: Cambridge University Press, Inc.
22. Rand, W., *Objective Criteria for the Evaluation of Clustering Methods.* Journal of the American Statistical Association, 1971. **66**: p. 846-850.
23. Hubert, L. and P. Arabie, *Comparing Partitions.* Journal of Classification, 1985. **2**: p. 193-218.
24. Laohakiat, S., S. Phimoltares, and C. Lursinsap, *Hyper-cylindrical micro-clustering for streaming data with unscheduled data removals.* Knowledge-Based Systems, 2016. **99**: p. 183-200.
25. Udommanetanakit, K., T. Rakthanmanon, and K. Waiyamai, *E-stream: evolution-based technique for stream clustering*, in *ADMA'07*. 2007. p. 605-–615.
26. *UCI Machine Learning Repository*. 1999.

27. Tron, R. and R. Vidal, *A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms*, in *CVPR*. 2007. p. 951-–956.

28. Trefethen, L.N. and D. Bau, *Numerical linear algebra*. 1997, Philadelphia: Society for Industrial and Applied Mathematics.

29. Laohakiat, S., S. Phimoltares, and C. Lursinsap, *A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction.* Information Sciences, 2017. **381**: p. 104-123.

30. Ntoutsi, I., et al., *Density-based projected clustering over high dimensional data streams*, in *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*. 2012. p. 987-998.

31. *Information Security Centre of Excellence (ISCX)*. 2015.

32. Ertoz, L., M. Steinbach, and V. Kumar, *Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data*, in *Proceedings of the 3rd SIAM International Conference on Data Mining*. 2003. p. 47-58.

33. Bohm, C., et al., *Density connected clustering with local subspace preferences*, in *Proceedings of the Fourth IEEE International Conference on Data Mining*. 2004, IEEE Computer Society: Washington, DC, USA. p. 27-34.

34. Houle, M.E., et al., *Can Shared-neighbor Distances Defeat the Curse of Dimensionality?*, in *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management*. 2010, Springer-Verlag: Heidelberg, Germany. p. 482-500.

35. Schubert, E., et al., *A Framework for Clustering Uncertain Data.* PVLDB, 2015. **8**(12): p. 1976-1987.

APPENDIX

VITA

Sirisup Laohakiat was born in Bangkok. He received B.Eng and M.Eng from Faculty of Engineering, Chulalongkorn unversity in 1996 and 1999 respectively. He received a grant from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. program in 2015.