AN ONLINE DENSITY-BASED CLUSTERING ALGORITHM FOR DATA STREAM BASED ON LOCAL OPTIMAL RADIUS AND CLUSTER PRUNING

MD KAMRUL ISLAM

MASTER OF SCIENCE

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : MD KAMRUL ISLAM

Date of Birth : 01 JULY 1985

Title : AN ONLINE DENSITY-BASED CLUSTERING ALGORITHM FOR DATA STREAM BASED ON LOCAL OPTIMAL RADIUS AND CLUSTER PRUNING

Academic Session : SEM 1 2019/2020

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_____	_____
(Student's Signature)	(Supervisor's Signature)
BY0934870	KAMAL Z. ZAMLI
_____	_____
New IC/Passport Number	Name of Supervisor
Date:	Date:

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

MAKLUMAT PANEL PEMERIKSA PEPERIKSAAN LISAN

Tesis ini telah diperiksa dan diakui oleh
This thesis has been checked and verified by

Nama dan Alamat Pemeriksa Dalam : Dr. Tuty Asmawaty Binti Abdul Kadir
Name and Address Internal Examiner Faculty of Computing
Universiti Malaysia Pahang
Pahang, Malaysia

Nama dan Alamat Pemeriksa Luar : Assoc. Prof. Dr. Junita Mohamed Saleh
Name and Address External Examiner School of Electrical & Electronic
Engineering, Universiti Sains Malaysia
Penang, Malaysia.

Disahkan oleh Timbalan Pendaftar di IPS
Verified by Deputy Registrar IPS

Tandatangan :
Signature

Tarikh:
Date

Nama :
Name

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Master of Science.

_____

(Supervisor's Signature)

Full Name : KAMAL ZUHAIRI ZAMLI

Position : PROFESSOR

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : MD KAMRUL ISLAM

ID Number : MCC17012

Date :

AN ONLINE DENSITY-BASED CLUSTERING ALGORITHM FOR DATA STREAM BASED ON LOCAL OPTIMAL RADIUS AND CLUSTERING PRUNING

MD KAMRUL ISLAM

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Master of Science

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

OCTOBER 2019

ACKNOWLEDGEMENTS

I am highly grateful to my supervisor Professor Dr. Kamal Z. Zamli for his emerging ideas, proper guidance and continuous support in forms of encouragement to carry on this research on right track. He has always impressed me with his outstanding professional conduct, his strong conviction for science, and his belief that a master program is only a start of a life-long learning experience. I appreciate his consistent support from the first day I applied to graduate program to these concluding moments. I am truly grateful for his progressive vision about my training in science, his tolerance of my naive mistakes, and his commitment to my future career. I would like to express very special thanks to my previous supervisor Dr. Md. Manjur Ahmed for his suggestions and co-operation throughout the study. I also sincerely thank for the time spent for proofreading and correcting my many mistakes.

My sincere thanks go to all faculty members of Faculty of Computing, College of Computing and Applied Sciences who helped me in many ways to make my life easy and pleasant at Universiti Malaysia Pahang. I am also grateful to all students and staff from SPINT lab for their inspiration and co-operation during the study.

ABSTRAK

Pengklasteran aliran data memainkan peranan penting dalam perlombongan data aliran untuk pengekstrakan pengetahuan. Dalam beberapa tahun kebelakangan ini, banyak penyelidik telah mengkaji teknik clustering berasaskan ketumpatan dalam talian kerana kemampuannya untuk menghasilkan kluster berbentuk bebas. Teknik ini meringkaskan aliran data dalam klaster mikro dengan mikro klaster tersebut membentuk kelompok. Walau bagaimanapun, sebahagian besar kluster ini sama ada tidak sepenuhnya dalam talian, atau tidak dapat mengendalikan sifat aliran data dengan betul. Selain itu, algoritma ini memerlukan penetapan awal radius optimum mikro kluster, yang merupakan tugas yang sukar, dan pilihan yang salah memburukkan kualiti kluster. Di samping itu, algoritma ini juga mengabaikan kehadiran kluster mikro sementara yang tidak relevan, walhal mungkin menjadi relevan pada masa akan datang. Hal ini menyebabkan kemerosotan kualiti kluster dan peningkatan masa pemprosesan kerana kelompok mikro dihapuskan dan dibuat kerap disebabkan oleh aliran data yang berubah-ubah. Dalam kajian ini, algoritma klaster berasaskan ketumpatan dalam talian yang dipanggil Penimbal Pengklasteran Dalam Talian untuk Aliran Data Berubah-ubah (BOCEDS) dibentangkan. BOCEDS mengelompokkan aliran data dalam satu peringkat. Algoritma meringkaskan data daripada aliran data dalam cluster mikro. Algoritma ini mengekalkan radius optimum tempatan mikro kluster optimum tempatan berbanding radius global dan malar. Selain itu, ia memperkenalkan penimbal untuk menyimpan kluster mikro yang tidak relevan serta proses pemangkasan sepenuhnya dalam talian untuk mengeluarkan kluster mikro yang tidak relevan dari penimbal. Proses pemangkasan ini dapat mengurangkan masa pemprosesan. Di samping itu, BOCEDS mencadangkan fungsi mengemaskini tenaga mikro-klaster dalam talian berdasarkan maklumat spatial aliran data. Geraf gumpalan kelompok klaster akan dihasilkan berdasarkan sambungan antara kluster mikro. Kemudian, klaster dihasilkan daripada graf gumpalan kelompok kluster tersebut. Untuk menilai prestasi, algoritma, BOCEDS dilaksanakan pada dua aliran data sintaktik dan satu data praktikal. Hasil eksperimen menunjukkan BOCEDS dapat menghasilkan kelompok baru dan menghapus kelompok lapuk dengan waktu seiring dengan perubahan kandungan data. Eksperimen dalam aliran data yang bising menunjukkan bahawa algoritma BOCEDS dapat mengesan kebisingan dengan ketepatan kira-kira 100%. Kejituan dan kesucian keseluruhan adalah lebih daripada 99%. Hasil eksperimen dibandingkan dengan algoritma kluster alternatif berasaskan ketumpatan hibrid dalam talian / luar talian. Masa pemprosesan purata untuk titik data dalam aliran data adalah kira-kira 2 milisaat yang jauh lebih rendah daripada algoritma kluster yang sejajar dalam literatur. Algoritma ini juga lebih berskala untuk aliran data dimensi yang tinggi daripada algoritma yang sedia ada. Kepekaan parameter clustering dalam BOCEDS juga diukur. Hasilnya menunjukkan bahawa perubahan nilai parameter kualiti kluster hanya menyimpang kualiti klaster dengan jumlah yang sangat kecil (<1%). Hasil ini membuktikan keunggulan algoritma BOCEDS berbanding algoritma kluster yang sedia ada.

ABSTRACT

Data stream clustering plays an important role in data stream mining for knowledge extraction. In recent years, numerous researchers have studied the online density-based clustering technique due to its capability to generate arbitrarily shaped clusters. The technique summarizes the data stream in micro-clusters and the micro-clusters form the clusters. However, most of the clusters are either not fully online, or cannot handle the properties of data stream properly. Moreover, the algorithms require predefining the global optimal radius of micro-clusters, which is a difficult task, and an erroneous choice deteriorates the cluster quality. In addition, the algorithms ignore the presence of temporarily irrelevant micro-clusters, which may be relevant in the future. This ignorance causes the degradation of clustering quality and the increase of the processing time as micro-clusters are deleted and created frequently due to evolving nature of data stream. In this study, a fully online density-based clustering algorithm called Buffer-based Online Clustering for Evolving Data Stream (BOCEDS) is presented. BOCEDS clusters the data stream in a single stage. The algorithm summarizes the data from data stream in micro-clusters. This algorithm maintains the local optimal radius of micro-clusters rather than a global and constant radius. Moreover, it introduces a buffer for storing irrelevant micro-clusters and a fully online pruning process for extracting the temporarily irrelevant micro-cluster from the buffer. The pruning process improves processing time. In addition, BOCEDS proposes an online micro-cluster energy updating function based on the spatial information of the data stream. Then, clustering graphs are generated based on the connectivity among micro-clusters. The clusters are generated from the clustering graphs. To evaluate the performance, BOCEDS algorithm is executed on two syntactic and one practical data streams. The experimental result shows BOCEDS is able to generate new clusters and remove outdated clusters with time as data stream contents change. The experiment on noisy data stream shows that BOCEDS algorithm can detect noise with an accuracy of approximately 100%. The overall clustering accuracy and purity are more than 99%. Experimental results are compared with other alternative online/offline hybrid density-based clustering algorithms. The average processing time for data point in the data stream is about 2 milliseconds which is much lower than the aligned clustering algorithms in literature. The algorithm is also more scalable to high dimensional data stream than the existing algorithms. The sensitivity of clustering parameters in BOCEDS is also measured. The result shows that in case of changing the values of parameters the cluster quality deviates by a very small amount (<1%). These results prove the superiority of BOCEDS algorithm over the existing clustering algorithms. The BOCEDS algorithm is then applied to real-world weather data streams to demonstrate its capability to detect the drifts in the data stream and discover arbitrarily shaped clusters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

^{o}F	Fahrenheit
$\cup$	Union
$\cap$	Intersection
ϕ	Empty set
$\in$	Element of a set
$\vee$	Logical OR
$\wedge$	Logical AND
mbar	Millibar
ms	Milliseconds
sec	Seconds

LIST OF ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
CF	Clustering Feature
CFMM	Copula-based Finite Mixture Models
ClustMD	Clustering for Mixed Data
EHCF	Exponential Histogram of Cluster Feature
EM	Expectation Maximization
GCHL	Grid-Clustering for High-dimensional Large databases
GDILC	Grid-based Density Isoline Clustering
gHHC	Gradient-based Hyperbolic Hierarchical Clustering
GIS	Geographical Information System
GMM	Gaussian Mixture Model
IT	Information Technology
IVHFAH	Interval Valued Hesitant Fuzzy Agglomerative Hierarchical
KnA	K-means and Agglomerative
MC	Micro-cluster
PB	Peta Bytes
SGC	Statistical Grid-based Clustering
ZB	Zeta Bytes

CHAPTER 1

INTRODUCTION

1.1 Introduction to Big Data and Data Stream

The fast development of information technology (IT) leads to generation of numerous amounts of data-generating applications ranges from machine condition monitoring and atmospheric science to social media analysis. Such large numbers of application usually generate a massive amount of data sets at every moment, often known as "big data" (Esposito et al., 2015). The term 'Big Data' was introduced by John in a Silicon Graphics (SGI) slide deck with the title "Big Data and the Next Wave of InfraStress" in 1988 (Fan & Bifet, 2013). TechAmerica Foundation defines 'big data' as-

> *"Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information."*

Unlike traditional data, big data is defined as very large amounts of structured, semi-structured or unstructured data (Losee, 2006). Structured data refers to the data with high degree of organization, such as in tables and relations; whereas unstructured data is essentially the opposite. On the other hand, semi-structured data is interpreted with structural information supplied as tags such as name='Kamrul', city='Gambang', gender='Male' instead of having regular structures. Big data are daily generated from heterogeneous sources at an unprecedented rate. The example of such big data sources includes sensor networks, anomaly detection, financial transactions, call records, social data, multimedia data, advertising, etc. The amount of data has grown exponentially

over the past decade. According to the September 2017 statistics, the customers of Wal-Mart provided approximately 2.5 petabytes (PB) or 10^{15} bytes of data per hour (Can & Alatas, 2017). The same kinds of trends were seen in case of other IT applications like Facebook, Twitter, and YouTube etc. The recent trend of big data growth has been analysed recently and the future trends of growth has been assumed (Statista, 2018). The trend analysis is shown in Figure 1.1.

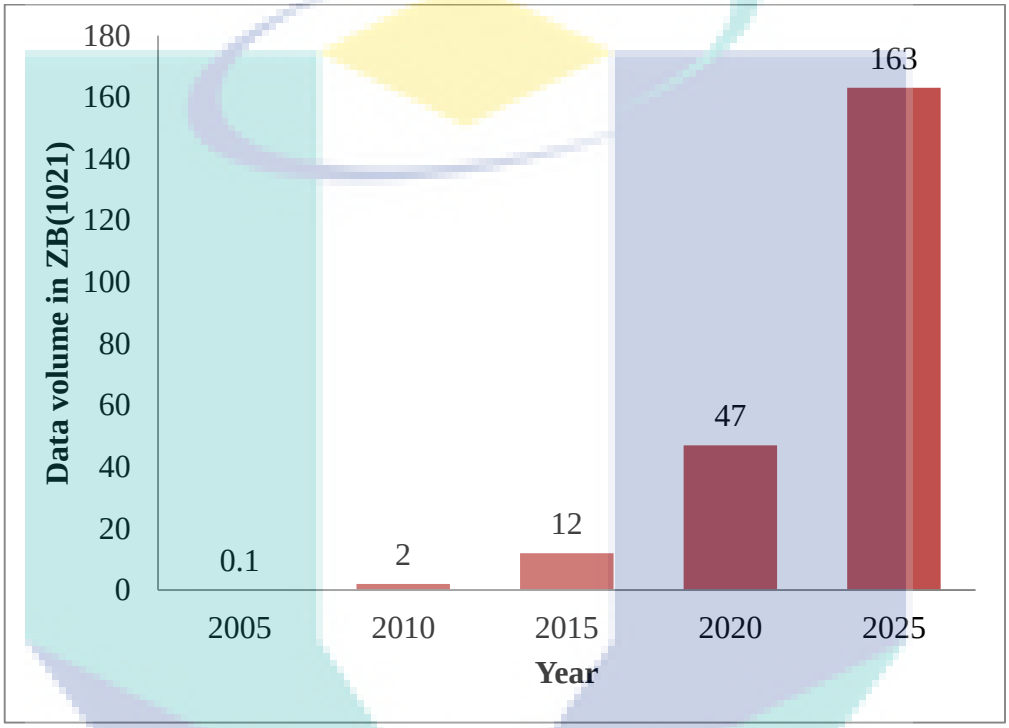


Figure 1.1 Growth of big data from 2005 to 2025

Source: Statista (2018)

According to the statistics, the world generated 0.1 zetabytes (ZB) or 100000 PB data in 2005. With growing the IT-based applications, the volume of data grew continuously and reaches to 12 ZB in 2015. It is estimated that by 2025, the data volume will be 163 ZB which 1630 time comparing to data volume 2005. The huge and unbounded series of data points that arrive continuously is referred to as a data stream (Krawczyk et al., 2017). They are enormous, rapid changing, and potentially limitless. Comparing to traditional static datasets, data stream poses three additional and special constraints (Krawczyk et al., 2017; Oussous et al., 2018; Silva et al., 2013) as in Figure 1.2. These three special constraints (volume, velocity and variety) are commonly known as the 3V properties of data stream.

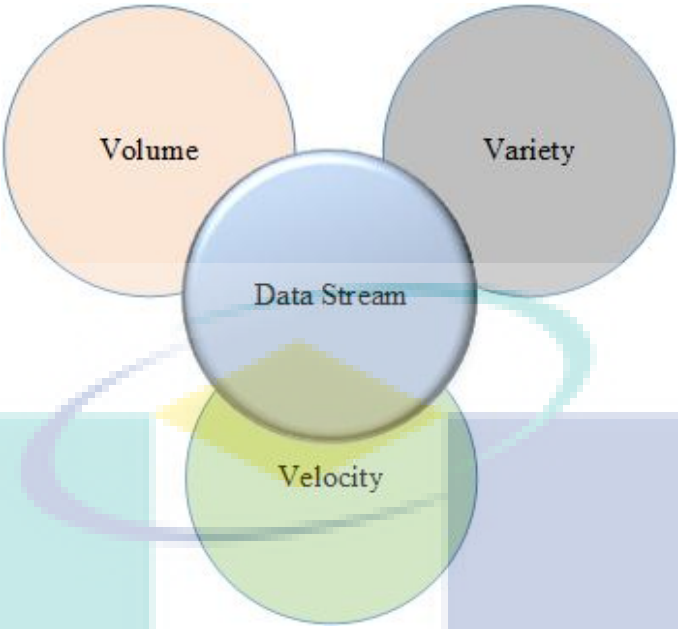


Figure 1.2 Properties of data stream

The properties are described as follows:

i. *Volume:* Volume refers to the size of the data stream. The data points are continuously coming from diverse sources. The volume of data stream is large and will grow to extremely large in future. This fact makes it impossible to store the data stream in memory to analyze.

ii. *Velocity:* Velocity is the rate of data generation in the data stream. The arrival speed of data in data stream is quite high. To deal with this high-speed data stream, fast processing of data points is required to enable real time processing of data stream.

iii. *Variety:* Variety refers to the structural heterogeneity in a data stream. In other words, it is the gradual change of data stream as time progress. The term to describe the fact is the evolving data stream. In such a data stream, an old data point from the stream may be irrelevant (or even harmful) for the current model. The detection of such change of data stream is desired.

1.2 Data Stream Clustering

Clustering of the data stream is one of the vital techniques in the field of stream mining and has a wide range of applications such as gene expression profiling, anomaly detection in bank transaction, image segmentation, text clustering, environmental trend analysis and so on (Z. Wang et al., 2018). The technique for partitioning the data stream into clusters based on the similarity among data points is known as data stream clustering (Bryant & Cios, 2018). Traditional data clustering algorithms are best equipped to run one-time on the concept of persistent data sets that are stored reliably in storage (Garofalakis et al., 2016). However, several modern applications generate data stream on a continuous basis. Due to volume characteristics of data stream, it is quite impossible and impractical to store the entire data stream in memory for analysis. The data points from data stream pass only once and so multiple scans are not feasible. Low processing time is another requirement to enable real time processing (Amini et al., 2014).

Given the unprecedented amount of data that will be produced, collected and stored in the coming years, one of the technology industry's great challenges is how to benefit from it (Al-Jarrah et al., 2015). Data analyst always looks for the technique which can extract the hidden knowledge in these data stream. The extracted knowledge has been used by researchers to solve social problems towards a comfortable life and a better world for humanity. Some examples of such efforts include social unrest prediction based on social media data (Ansah et al., 2018), people demand analysis for new product development (Zhan et al., 2019), building groups of genes with related expression patterns for genome annotation (Gudenas et al., 2019), analysing patterns of antibiotic resistance for new antibiotic development (Mohana et al., 2018), identifying areas where there are greater incidences of a specific type of crime (Win et al., 2019), finding weather regimes (Hannachi & Trendafilov, 2017). Mining data streams is one of the knowledge extraction technique that has attracted the researchers and clustering is a significant part of mining data streams (Ackermann et al., 2012). The technique for partitioning the data in data stream into clusters is known as data stream clustering where the similar data are placed in the same cluster, and dissimilar data are placed in different cluster (Nguyen et al., 2015). Increasingly, it has become a useful, ubiquitous and essential tool in data stream analysis.

Considering the above constraints, a good data stream clustering algorithm is one which tries to achieve the following properties (Han et al., 2011; Kranen et al., 2011).

i. The clustering result is generated with minimum time to deal with this high speedy property of data stream towards real time processing of data stream.

ii. The model evolves to detect and learn the changes of concept in data stream rapidly. The current model always reflects the recent nature of data stream.

iii. The number of clusters is not constant and varies with time. The number of total cluster in the model completely depends on the data points those are arriving continuously from the data stream.

iv. Noise in the data stream is detected in forms of the outlier and actions are taken accordingly.

v. Be able to process heterogeneous data stream.

vi. Be able to process high-dimensional data stream as the data stream may contain a high number of attributes or dimensions in their nature such as genomics data stream.

The clustering approaches are categorized into five categories and they are model-based, partitioning, grid-based, hierarchical, and density-based clustering. This categorization can be drawn as Figure 1.3.

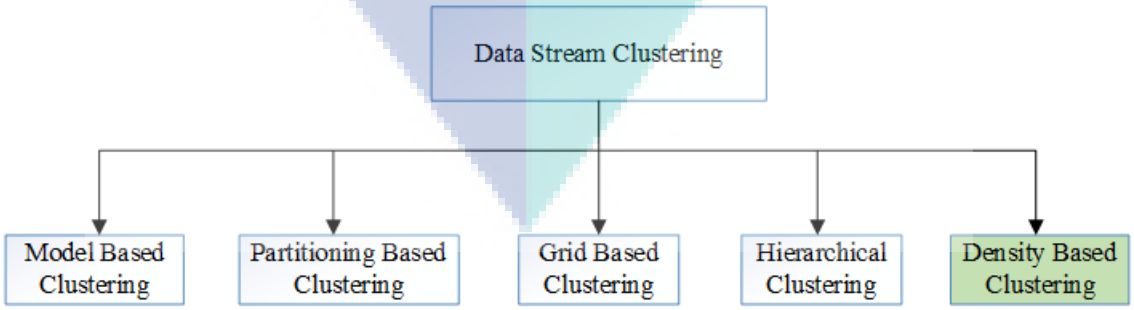


Figure 1.3 Hierarchy of data stream clustering approaches

1.2.1 Model-based Clustering

The model-based clustering technique aims to find the best fit the data points to a cluster based on the mathematical model like EM (Expectation Maximization) algorithm (D. Xu & Tian, 2015). K-means algorithm is adapted to design model-based clustering. EM maps the data point to an existing cluster based on a weight which represents the probability to a spherical-shaped cluster membership. Some popular model-based clustering algorithms include GMM (Rasmussen, 2000), RJMCMC (Malsiner et al., 2016), CFMM (Kosmidis & Karlis, 2016), ClusMD (McParland & Gormley, 2016). The model-based data stream clustering techniques suffer from the well-known 'curse of dimensionality' problem (Donoho, 2000) where memory space and time requirement grow at a faster than linear rate with the growing dimension size. Moreover, the model may ignore the clusters which are small but significant.

1.2.2 Partitioning Based Clustering

The partitioning based clustering technique divides the data space into some partitions and each partition forms a single cluster (Sardar & Ansari, 2018). This clustering technique is also designed based on the k-means algorithm to form spherically shaped clusters. Most data clustering algorithm from this category maps the data point to a partition based on the distance among data points. K-Means (MacQueen, 1967), K-Medoids (H. S. Park & Jun, 2009), CLARA (Kaufman & Rousseeuw, 2009), CLARANS (Ng & Han, 2002) are example of some popular partitioning based clustering algorithms. Though the partitioning based clustering techniques are simple, scalable and require low processing time, they are limited to spherically shaped clusters only and cannot extract clusters of arbitrary shape. Moreover, the clustering results are usually influenced by noise.

1.2.3 Grid-based Clustering

A grid-based clustering technique for data stream creates a number of cells called grids in data space to form grid structure and then form the clusters from the cells in the grid structure (Cheng et al., 2018). Unlike the partitioning technique, the partitioning process does not depend on the distribution of data points. This technique utilizes a multi-resolution grid structure. The grids which have more density than its neighbour grids form the clusters. SGC (N. H. Park & Lee, 2004), GDILC (Zhao &

Song, 2001), GCHL (Pilevar & Sukumar, 2005), SGC (W.M. Ma & Chow, 2004) are well known algorithms in the field of grid-based clustering of data stream. Since these clustering techniques consider the density of grids to form cluster, so they are mostly considered as density-based clustering. The advantages of this category of clustering technique include low processing time, but they suffer from 'curse of dimensionality' problem. Moreover, the optimal value of grid size should be predefined by user.

1.2.4 Hierarchical Based Clustering

The hierarchical clustering utilizes the concept of CF-tree (Clustering Feature), where the summarization of the data stream is represented by a node in a balanced tree (Bouguettaya et al., 2015). The nodes are created and balanced based on predefined threshold number of data points. The non-leaf node of the tree aggregates the statistics in its descendant nodes which are used to generate the clusters. Once an operation is finished to merge or split the node, it cannot be reversed. KnA (Bouguettaya et al., 2015), IVHFAH(Xiaolu Zhang & Xu, 2015), gHHC (Monath et al., 2019) are popular hierarchical-based clustering algorithms. Hierarchical clustering is simple and appropriate for data stream with well-separated spherical clusters. Moreover, these techniques avoid the necessity of defining the number of target clusters in advance. However, the technique is expensive in high dimensional spaces due to the curse of dimensionality phenomenon.

1.2.5 Density-based Clustering

The final category, density-based clustering technique has been developed based on the concept of density. The total number of data points in a region referred to as the density of the region. The region is defined by a center and a radius which contributes to form either cluster or outlier. A cluster is a set of density-connected data points with maximum density reachability. Thus, the denser regions in the data space form cluster and they are separated by the sparse regions. The data points which do not belong to any of the current clusters are considered to be outlier or noise (Han, 2005). Density-based clustering has been found as a natural and attractive clustering technique as it has the ability to generate arbitrarily shaped clusters, to handle the evolving nature of data stream and to detect noises and act accordingly in noisy environment. Therefore, they have become the most appropriate clustering method for data stream (Amini et al.,

2014). In recent years a lot of researchers have proposed density-based data stream clustering techniques. But most of them are not fully online methods, or unable to handle evolving or noisy characteristics of data stream or suffers from low-performance problem like high memory requirement, low processing rate, low cluster quality, low data coverage or curse of dimensionality (Hyde et al., 2017)

1.3 Problem Statement

Data stream clustering is an unsupervised learning technique in the field of data stream mining (Reddy & Bindu, 2017). Among the five categories of clustering (

Figure 1.3), density-based technique has gained remarkable popularity due to its ability to extract arbitrary shaped clusters, to identify noise and avoiding the requirement of predefining the number clusters (Amini et al., 2014). There are many density-based clustering algorithms exist today. A majority of such algorithms are either hybrid online/offline methods, windowed offline methods, or unable to handle evolving nature of data stream (Hyde et al., 2017). The windowed offline method requires a portion of data to be stored in memory for analysis and hybrid online/offline method requires generating clusters by an offline process. However, they are not ideal for data stream clustering as the clustering results are not always available and online data cannot be stored or postponed due to the unknown size and order of data points (Sharma & Sharma, 2017). A few density-based clustering algorithms are fully online. However, in the field of online density-based clustering, two major problems still exist:

1st problem: Density-based micro-clustering algorithms for data stream require setting the optimal value of global and constant micro-cluster radius prior to the execution of algorithms.

Description of 1st problem: The performance of the density-based micro-clustering algorithm for data stream is highly dependent on setting the optimal value of global and constant micro-cluster radius prior to the execution of the algorithm. A global radius is used for all micro-clusters and remains constant during the execution of the algorithm. However, a unique radius for all micro-clusters may provide inefficient clustering result. For illustrating the fact, consider the data distribution in Figure 1.4.

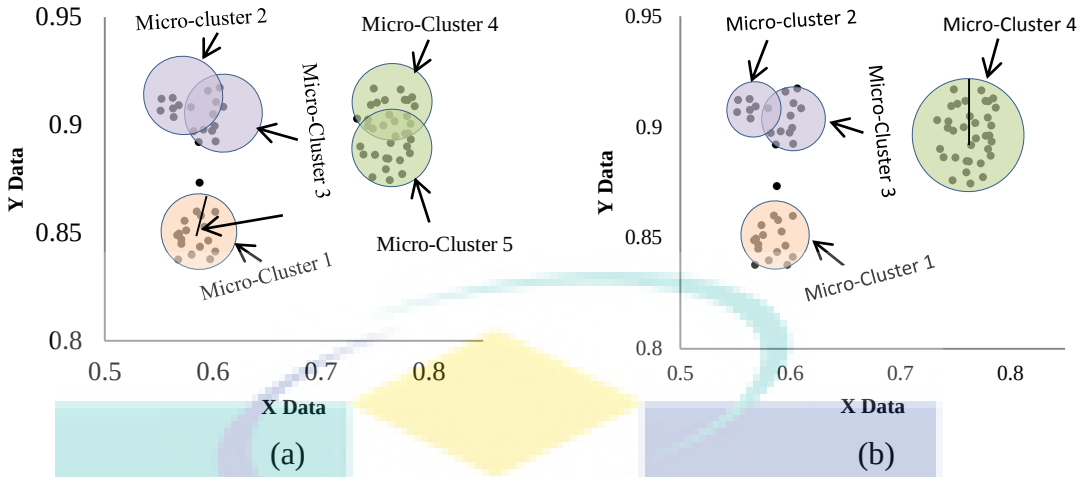


Figure 1.4 Micro-cluster formation (a) Micro-clustering by unique radius (b) Micro-clustering by variable radius.

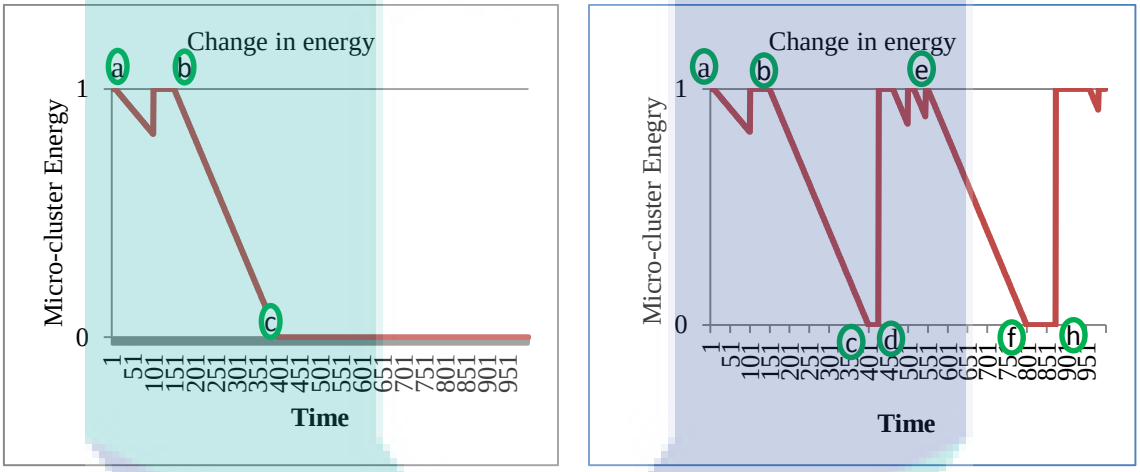
Figure 1.4(a) shows the formation of micro-clusters by micro-clustering technique with a unique radius for all micro-clusters. This radius is input from user and remains constant. Figure 1.4(b) shows the formation of micro-clusters by human with eye and hand where every micro-cluster has its own optimal value of clustering radius. Micro-clusters are formed in dense regions separated by sparse regions. In Figure 1.4(a), micro-cluster 2 and micro-cluster-3 formation includes sparse regions with dense regions. Moreover, though there is not sparse region between micro-cluster 4 and micro-cluster 5, two micro-clusters are formed by violating the rule of separateness between micro-clusters. Thus, comparing Figure 1.4(a) and Figure 1.4(b), it can be said that Figure 1.4(b) is more efficient clustering than clustering in Figure 1.4(a).

This problem generates two performance issues. The first issue is that there are many IT applications where it is really difficult to set the optimal micro-cluster radius. An erroneous choice of the radius degrades the cluster quality remarkably. Secondly, the concept of global radius further affects the cluster quality by generating the clusters in sparse region.

2nd problem: The existing density-based micro-clustering algorithms cannot identify the temporarily irrelevant from irrelevant clusters.

Description of 2nd problem: In most of the real world IT-applications, the concepts in data stream change over time. This property of data stream can invalidate the current learned model (Gomes et al., 2017). The micro-clusters in the learned model those do not represent the current data stream contents are called irrelevant micro-clusters. Most

of the evolving clustering algorithms for data stream use micro-cluster energy to detect and remove the irrelevant micro-cluster in handling the evolving nature of the data stream. However, they fail to identify the temporary irrelevant but relevant in near future cluster. For illustration the problem, consider the change in energy due to data point arrival to an evolving micro-clustering environment in Figure 1.5. Figure 1.5(a) shows at point 'a' a micro-cluster is created. Then, at point 'b' the energy of the micro-cluster begin to fall due to no data point falls into this micro-cluster. Finally, at point 'c' the energy falls down to zero and at this point the micro-clustering techniques mark this micro-cluster as irrelevant to current data stream content and delete from the system. Figure 1.5(b) shows the actual situation where micro-clusters are not deleted.



(a) (b)

Figure 1.5 Evolving of micro-clusters (a) Removing micro-cluster, (b) Without removing micro-cluster

Likewise Figure 1.5 (a), at point 'c' the micro-cluster energy goes to zero. But at point 'd' again data point resides in the micro-cluster and the micro-cluster is alive again. From point 'd' to 'e' the micro-cluster receives data points and at 'f' again the energy to zero and again alive at point 'f'. Hence, the micro-cluster is said to be temporarily irrelevant but relevant at near future. The failure to identify the temporary irrelevant micro-clusters let the model to create and delete the micro-clusters frequently. This operation increases the computational time.

In order to mitigate the aforementioned limitations, a new density-based clustering strategy needs to be designed which will work in a fully online manner. The clustering parameter micro-cluster radius needs to be self-adapted based on data stream content to reduce the dependency on user to avoid erroneous setting. Micro-clusters have to maintain their own radius independently. The micro-clusters those are

temporarily irrelevant but relevant at near future, need to be identified and pruned. The pruning operation prevents the memory to grow beyond the limit and also to conform that the out-dated clusters are removed from the system.

1.4 Research Objectives

The main concern of this research is to design a fully online density-based clustering algorithm that handles the challenges of data stream efficiently. To achieve the goal, the following four objectives have been set in this research as follows.

i. To design an online clustering algorithm based on the concept of local optimal radius and irrelevant micro-cluster buffering.

ii. To implement the algorithm by adapting a non-linear procedure for updating the micro-cluster energy and pruning the micro-clusters.

iii. To evaluate the performance of the developed algorithm against selected benchmark functions as case studies.

1.5 Research Scopes

This research focuses on only density-based clustering among the five categories of data stream clustering technique. The research has some scopes but not limited to as follows.

i. The focus of this study is on the development of a new online density-based clustering algorithm, called Buffer-based Online Clustering for Data Stream (BOCEDS) for evolving data stream.

ii. The cluster information in BOCEDS algorithm is updated using recursive procedure to ensure the fully online behaviour of the algorithm.

iii. There are different kinds of attribute in the data stream including numerical, categorical and uncertain data. The current study considers only numerical attribute in the data stream to generate clusters.

iv. The criteria to evaluate the developed algorithm include visualization of cluster, noise sensitivity, accuracy, purity, data processing time, scalability, memory efficiency and parameter sensitivity. The performance is measured by executing

the algorithm on two benchmark syntactical data streams (Mackey-glass and helical data stream) and one practical data stream (KDDCUP'99 data stream).

v. The designed BOCEDS algorithm is implemented in MATLAB programming environment.

1.6 Thesis Outlines

This chapter briefly introduced the research background and some preliminary knowledge regarding density-based clustering algorithms. The problem statement, research objectives, and research scopes were included in this chapter. The rest of this thesis is organized as follows.

Chapter 2 introduces the existing algorithms in the field of density-based clustering. The algorithms are categorized according to their data processing and working principle in Section 2.3. Each algorithm is discussed shortly with their working steps in Section 2.4. The advantages and limitations of each algorithm also identified and summarized in Section 2.5.

Chapter 3 describes the methodology of the developed algorithm in the field of density-based clustering for data stream namely Buffer-based Online Clustering for Evolving Data Stream (BOCEDS). The flowchart and design concept can be found in Section 3.2. The data structures used in the developed algorithm is explained in Section 3.2.1. The detailed discussions and algorithms of BOCEDS could be found in Section 3.2.2. Finally, Section 3.3 summarizes the whole methodology.

Chapter 4 illustrates the evaluation of the characteristics of the developed BOCEDS algorithm. In-depth discussions on several characteristics such as cluster quality, scalability, memory requirement, noise sensitivity could be found in this chapter. The performance of developed algorithm is compared to other popular algorithms in literature. The sensitivity of algorithmic parameters is also measured and described in details. The developed algorithm is applied to a real time data stream to show its effectiveness in real world.

Finally, Chapter 5 covers the conclusions of this research including the major contributions and summary of findings. Also, the future research recommendations could be found in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

Modern IT-based applications produce data stream of huge size and high degree of complexity. The analysis of such data streams and knowledge mining is becoming vital for the success of organizations (Esposito et al., 2015). Researchers introduce several data mining techniques for extracting hidden knowledge in the data stream (Ramírez et al., 2017). Data stream clustering is a preliminary stage of data stream mining where the data stream is partitioned into several partitions called clusters (Jacques & Preda, 2014; Nguyen et al., 2015; Puschmann et al., 2017). It is the most commonly used and essential unsupervised learning tool in data stream analysis (Lv et al., 2016). There is a lot of research on clustering algorithms for static datasets, but they cannot be applied on data stream due to three special characteristics (volume, velocity, variety) (Chenaghlou et al., 2018) (discussed in Section 1.1, Chapter 1). Considering these three characteristics, the requirements of a good data stream clustering technique tries to achieve minimum processing delay, to detect noise and evolving nature of data stream without predefining the number of clusters and able to generate arbitrarily shaped cluster (discussed in Section 1.2, Chapter 1). Over last decade, researchers have introduced numerous clustering algorithms for clustering of data stream which is broadly categorized into five categories (discussed in Section 1.2, Chapter 1). Among the categories of clustering (Figure 1.3, in Chapter 1), density-based clustering has been found a natural and attractive clustering technique. It has the ability to generate arbitrarily shaped clusters in dense areas, to handle the evolving nature of data stream and to detect noises and act accordingly in noisy environment making the most appropriate clustering method for data stream (Amini et al., 2014). The derivation of

density-based clustering method from mining for data stream is illustrated in the following Figure 2.1.

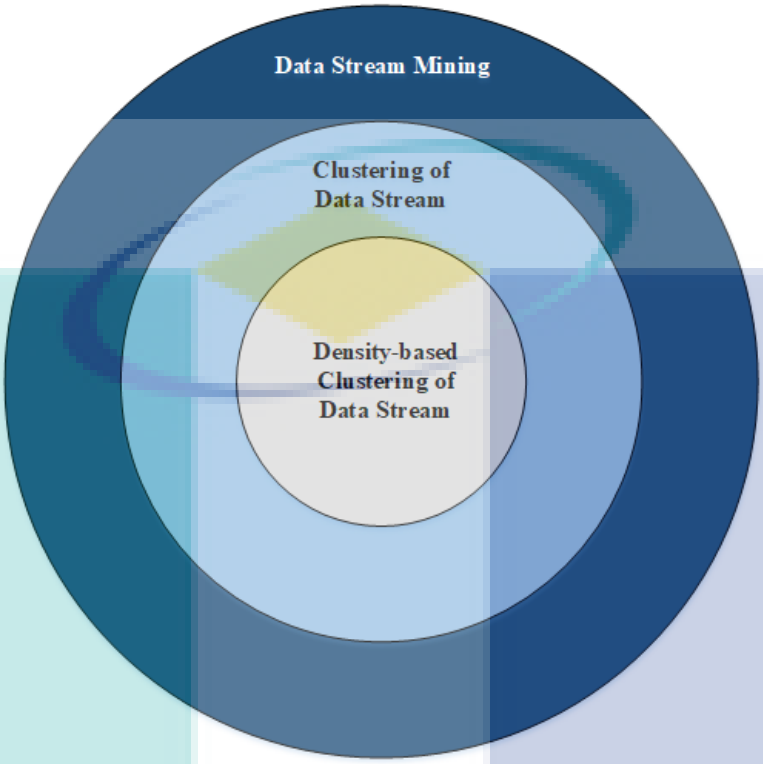


Figure 2.1 Drive towards density-based clustering

From Figure 2.1, knowledge extraction from the data stream includes various tasks like feature transformation, classification, clustering, association, and so on. Clustering is one of the vital tasks where data groups are extracted and density-based clustering is a popular method of clustering.

In this chapter, an extensive literature review has been done on density-based clustering. The concept of data stream mining has been discussed in Section 2.2. In Section 2.3, clustering algorithms are described with their category. The most popular algorithms of density-based clustering on data stream have been reviewed. The pros and cons of every algorithm have been analysed which are described in the following Section 2.4.

2.2 Data Stream Mining

Due to the rapid development of IT, big data applications have risen tremendously. Numerous such applications are generating huge data collection

continuously (known as data stream) and now commonly used tools and techniques fail to capture, manage, and process within acceptable processing time. The most fundamental task of these data stream processing is to extract useful knowledge for further actions (X. Wu et al., 2014). The process of discovering interesting patterns and knowledge from data stream is referred to as data stream mining (L. Xu et al., 2014). Traditional data mining techniques cannot process the data stream efficiently as data stream possess three special properties (discussed in Section 1.1, Chapter 1). It is evitable to use special mining technique for knowledge extraction from data stream, called data stream mining. Data stream mining faces the challenges of data stream accessing and computing process as the data volumes may rise continuously. An efficient data stream platform needs to consider large-scale memory for mining task. The comparison between traditional data mining and data stream mining technique is illustrated in the following Table 2.1.

Table 2.1 Traditional data mining VS data stream mining

Characteristics	Traditional data mining	Data stream mining
Memory	Unlimited	Bounded
Number of passes	Multiple	Single
Time	Unlimited	Real-time
Concept	Fixed	Evolve

From Table 2.1, Traditional data mining algorithms require the whole data set to be loaded into the memory (Fong et al., 2016). The data set can be scanned several times to generate an improved result. Unlike them, data stream mining algorithms face the technical barrier because of uncertainty in volume of data stream. Thus, storing data stream may not be feasible. The data points come and require immediate processing for once and removed within short period of time from the system. While traditional data mining techniques has much time to generate results from data points, data stream mining requires real time processing. Traditional data mining processes within a single concept environment where the concept is constant. However, data streams are dynamic and the concepts evolve with time leading to produce multiple concepts. For example,

in a network monitoring application, the attackers (the person who tries to steal data from a network) always try to introduce new attacking methods (e.g. anomaly data injection, worms spreading) to breakdown the system by inserting anomaly data in the application. In this system, the number of class of network attack may rise over time. Data stream mining contains several tasks feature transformation, data summarization, classification, clustering, association and trend analysis (Sumathi & Sivanandam, 2006)

2.3 Clustering of Data Stream

Clustering of data stream is one of the major techniques in the area of data stream mining. Clustering of data stream refers to the task of grouping the data points from data stream in such a way that data points in the same group (called a cluster) are more similar to each other than to those in other groups (Bryant & Cios, 2018). This technique is used in a lot of fields like image analysis, remote sensing, bioinformatics, and text analysis. Researchers have done a lot of work in the field of clustering and introduced several algorithms for clustering of data stream. The clustering algorithms are classified based on their working principle and data point processing technique

2.3.1 Based on Working Principle

Based on working principle, the clustering algorithms are broadly classified into five categories and they are model-based, partitioning based, grid-based, hierarchical, and density-based clustering approach (discussed in Section 1.2, Chapter 1). The comparison among the five categories of data stream clustering is given in the following Table 2.2. From Table 2.2, data stream partitioning is the simplest clustering approach that shows the very good clustering performance in terms of clustering accuracy, purity and time complexity. However, it suffers from the requirement of pre-defining the number of clusters in the system. Additionally, this clustering approach only generates spherical shaped clusters and fails to discover arbitrary shaped clusters. Hierarchical clustering produces the clusters from the data stream in a natural way. But the time complexity is higher compared to other clustering approaches. Moreover, the clustering performance of clustering highly depends on the sequence of data points from the data stream.

In terms of processing time, grid-based clustering is much more efficient that is able to find arbitrary shaped clusters and detect outliers (noise clusters) in a noisy

environment. The weaknesses of this approach are the clustering result highly depends on the grid defining and the approach is not efficient for high-dimensional data stream. Model based approach depends on predefining the models based on domain knowledge.

Table 2.2 Comparison of different clustering approaches

Approaches	**Advantages**	**Limitations**
Partitioning	Simple and relatively efficient	Need to specify the number of clusters and unable to discover non-spherical clusters
Hierarchical	Derive more meaningful cluster structures	High complexity and sensitive to the order of the data records
Grid-based	Fast and can discover arbitrary-shape clusters in noisy environment	The clustering quality depends on the grid granularity and unsuitable to high-dimensional data
Model-based	Simple and include domain knowledge	Depends strongly on the assumed models
Density-based	Find arbitrary shaped clusters, robust to noises and high cluster quality	Need many parameters to be predefined.

The final approach is density-based clustering of data stream. It is the most popular approach that produces arbitrary shaped clusters and can detect the noises in the data stream efficiently with high cluster quality. However, the only downside of density-based clustering approach is that it requires defining two clustering parameters in advance and they are micro-cluster radius and density threshold.

2.3.2 Based on Data Stream Processing Method

In the field of clustering of data stream, the most important challenge is to process the continuously generated data points which change over time. There are some clustering methods for processing data streams. They are broadly classified into the two categories and they are online-offline and online clustering (Nguyen et al., 2015).

2.3.2.1 Online-offline Clustering

Online-offline data stream clustering is useful when the system requires to investigate the clustering result over different parts of the stream (Amini et al., 2014).

Though, different windowing models are available for tracking the evolving characteristics of data stream, they do not perform dynamic clustering over all possible time horizons. It is two-phase algorithm where the summary information of data stream is maintained in the online phase and clusters are formed based on data summaries in the offline phase (Mansalis et al., 2018). This type of algorithm contains the online and the offline phases as below:

- *Online phase*: The online phase faces the data points from the data stream and maps the data points to current clustering system. Finally, the summary information about data points is produced and maintained.
- *Offline phase*: The summary information is used towards generating the clusters in offline mode in demand. The shape of clusters can be ellipsoidal, spherical, box or arbitrary

2.3.2.2 Online Clustering

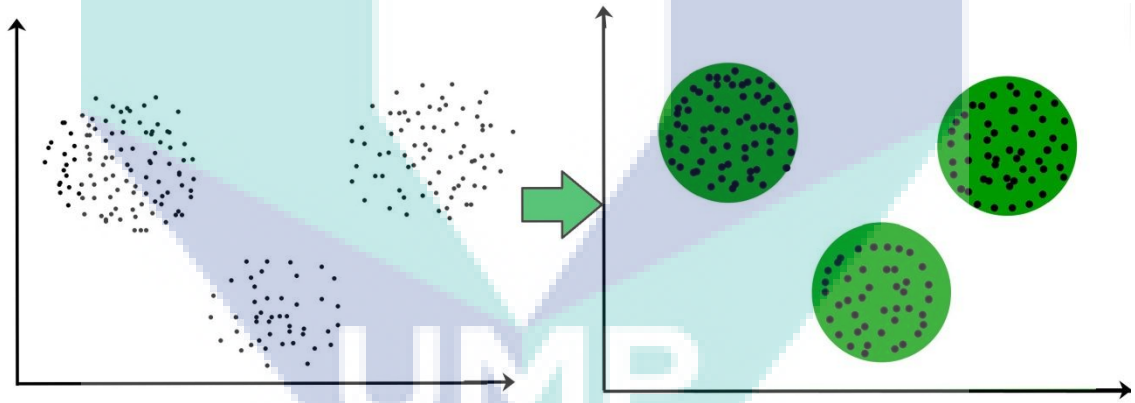
When a clustering technique generates clusters in a single online phase, then the clustering is referred to as online clustering. There are two ways of implementing online clustering and they are as follows.

i. *Chunk-based:* In many applications, the data streams are generated in a way that they seem to be in a packet. For example, 500 data points per minute. These 500 data points collectively form a chunk. Assuming the data points are coming in chunk, clusters are generated in a single pass by scanning the data stream in chunk. Every chunk is processed and the clustering result is updated. Different density based clustering algorithms like STREAM, DUCstream are designed based on this method.

ii. *Evolving:* In the chunk based approach, the clusters are computed in every chunk. However, the data points from data streams arrive continuously and the concepts change as time progresses. As a result, the clusters evolve over time. In the evolving approaches, the data points are processed individually and the clustering results are computed at every data points. Clustering algorithms such as DEC, CODAS, CEDAS were developed based on this approach.

2.4 Density-based Clustering

Density-based clustering techniques are developed based on the density of data points from the data stream. In density-based algorithms, a cluster is defined as a connected dense points and grows in the direction driven by the data density (Lv et al., 2016). The clusters are built in dense regions of data space which are isolated by the sparse regions. The basic principle is to generate a cluster as soon as the density in an area above the density threshold. The density threshold confirms that the background noisy data points or outliers are detected and filtered out. Density-based clustering of data stream is done in two major steps (Amini et al., 2014). At first step, the procedure of density estimation is formulated for each data point and applied to identify the data points those stays within dense regions (core data points). In the final step, a region formation method is defined that detects the group of data points those are reachable from core data points. The method should work only in the dense regions and no two data points in a low density region should be reachable. Cluster generation using density-based clustering method is illustrated in Figure 2.2.



(a) Data point distribution (b) Clusters

Figure 2.2 Density based clustering

Figure 2.2(a) shows the data point distribution in a 2D data space where clearly the data point are distributed in three regions. The generated clusters are shown by the green colour region in Figure 2.2(b). The figure shows the clusters are formed in dense region by core data points which are separated by the non-core data points in sparse region.

In the field of density-based clustering of data stream, DBSCAN (Ester et al., 1996) is considered to be the primitive algorithm that generates arbitrarily shaped clusters in an incremental manner. It was developed for large spatial dataset but later adopted by various clustering algorithm for data stream. In each iteration, the DBSCAN scans the unvisited data points and forms new cluster until all the data points are visited. The algorithm has the ability to find arbitrarily shaped clusters and it is robust to outliers. But it is not entirely deterministic as any points those are reachable from more than one cluster, can be part of either cluster. Moreover, it is not preferable for high dimensional data set as it suffers from the so-called "curse of dimensionality" difficulty and it takes much memory space for loading the whole dataset in memory. Several researchers have proposed clustering algorithms based on the concept derived from DBSCAN. The algorithms are categorized in two broad classes called density micro-clustering and density grid-based clustering algorithms (Amini et al., 2014). Several algorithms are developed in both subcategories. Figure 2.3 shows the taxonomy of literatures.

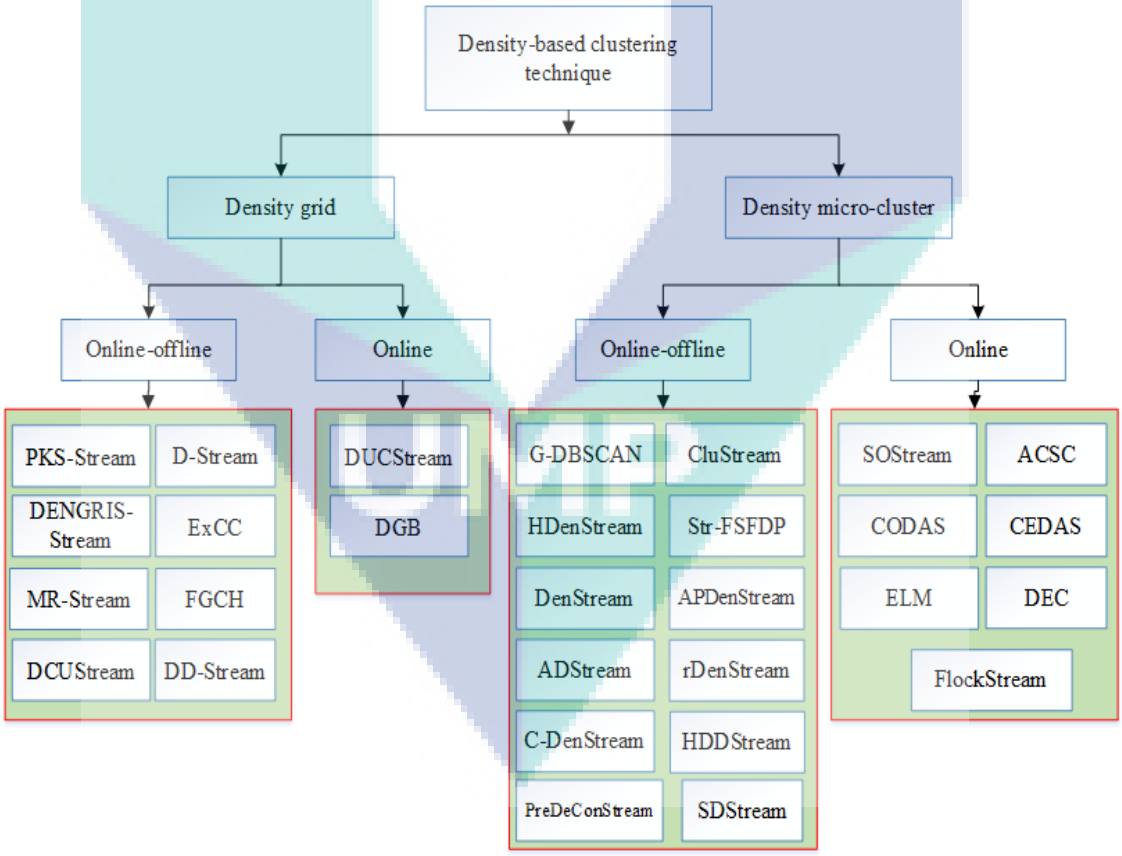


Figure 2.3 Taxonomy of reviewed density-based clustering algorithms

Some of the algorithms work in an online basis where the clustering results are available at every moment. On the other hand, some of the algorithm are hybridization of online and offline phases. The popular literatures in the field of density-based clustering algorithm for data stream have been reviewed. The articles are collected from high impact journals. The contributions and applications are identified with their advantages and limitations.

From Figure 2.3, the algorithms are divided into density grid-based and density micro-cluster-based clustering algorithms. Each category of algorithms is further grouped into online-offline and online clustering group. The algorithms are described with basic ideas, their applicability, advantages and limitations. Section 2.4.1 describes the algorithms from density grid-based category whereas algorithms from density micro-clustering algorithms are explained in Section 2.4.2.

2.4.1 Density Grid-based Algorithms

Applying the concept of grid based method on density based clustering method; researchers have developed several hybrid clustering algorithms for data streams. They are referred to as density grid-based clustering algorithm. The general framework for this type of algorithms is illustrated in Figure 2.4. From Figure 2.4

Figure 2.4, density grid-based clustering algorithms divide the total data space into grids and the following data points are mapped to grids in the first step. In the final step, the clusters are generated based on the density of grid. Density grid-based algorithms are popular for forming arbitrary shape cluster and detecting the noise with low processing time.

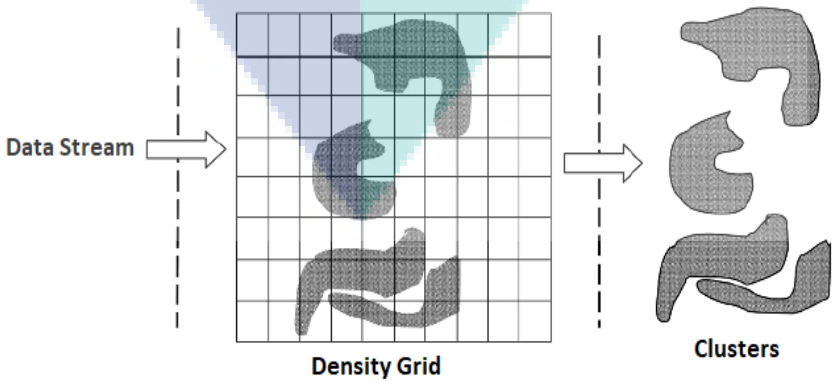


Figure 2.4 Framework for density grid-based clustering

To cope with the evolving nature of data stream, a parameter called 'density coefficient' is used for every data point from the data stream. This type of algorithm maintains the summary information of data points in characteristic vector. For detecting the noisy data points, density grid-based algorithms use sporadic grids.

Researchers have proposed several density grid based clustering algorithms which can discover arbitrary shaped clusters in dense areas. The algorithms process the data points in either online-offline or online mode. In the following section (Section 2.4.1.1), the evolution of online-offline density-grid clustering algorithms and online density grid algorithms of clustering are described in Section 2.4.1.2.

2.4.1.1 Online-offline Density Grid-based Clustering

Density grid-based clustering algorithm is a mixture of density based clustering and grid based clustering. The online-offline mode of this category is refers to the fact that the algorithms consist of two phase; the online phase and the offline phase. Aggarwal et al. (Aggarwal et al., 2003) introduced the online-offline clustering of data stream algorithm to enable real time stream processing and to meet the storage constraint. The clustering algorithm contains an online phase where the summary information of data stream is computed and an offline phase where clusters are generated. Based on this concept, researchers have designed many clustering algorithms in the following years. In the field of density grid-based clustering of data stream, DCUStream (Dynamic Density Based Clustering of Uncertain Data Stream) (Y. Yang et al., 2012) is considered to be an excellent algorithm. The algorithm is specially designed for uncertain data stream. It is a two steps algorithm where the grids are computed from the whole data space in the first step. The arriving data point is mapped to an existing grid based on the uncertain tense weight of data point. The grids are either dense grid (density more than the dynamic density threshold) or spares grid (density less than the dynamic density threshold). In the final step, DCUStream generate clusters using dense grids and outlier using sparse grids. DCUStream improves cluster quality and able to handle noise in uncertain evolving data stream. However, it suffers from the depth first search time consuming process to find the core dense grids.

For handling the evolving behaviour of data stream efficiently, three popular algorithms are D-Stream (Y. Chen & Tu, 2007), DD-Stream (Jia et al., 2008), and DENGRIS-Stream (Amini & Wah, 2012). Based on density decaying function, Chen

and Tu proposed an algorithm called D-Stream to detect the evolving nature of data stream. In online part, the newly arrived data point is mapped to a density grid based on its density coefficients and updates the characteristic vector of the mapped grid only. Based on the decay, a grid turns into dense grid, transitional grid or sparse grid. One type of grid can promotes or demotes to another type of grid. Considering the fact, the offline part inspects each grid's density and derives the clusters in each time interval gap. D-Stream defines another term sporadic grid which is one kind of sparse grid with very few data points and need to remove from memory as outlier. D-Stream finds the clusters of arbitrary shapes. It can handle the evolving behavior and detect the outliers in data stream. The authors claim the improvement of space and time efficiency. But the technique is not scalable as it relies on emptiness of large majority grids in high-dimensional data streams. To enhance the quality of generated cluster, D-Stream was improved in DD-Stream. The algorithm detects the data points in the border of grids using a proposed DCQ-means algorithm. The online phase is quite similar to the online phase of D-Stream. Additionally, the data points are placed on borders based on their distances from neighbouring grids and grid density. The offline extracts the boundary points from the grids at each inspection period, identify the sparse grid and dense grid based on their density and threshold and apply the same density-based methods as in D-Stream on dense grids to generate the clusters. DD-Stream is scalable to high dimensional data stream and generates high quality of the cluster. However, it suffers from a time-consuming process namely the border points mapping grids. Moreover, it does not describe clearly the removal process of sporadic grids. DENGRIS-Stream is another density grid-based clustering algorithm for data stream that works based on sliding window concept. The algorithm handles the evolving behaviour of data stream efficiently. In the online phase, each data point from data stream is mapped to a grid in the model and the grid summarization is updated. The clusters are generated based on the grid summary within time window units. The algorithm detects the expired grid using time stamp strategy and removes them. The algorithm shows excellent performance in terms of memory requirement and data processing time. However, DENGRIS-Stream requires the evaluation over various data stream and comparison with other state-of-the-art algorithms.

Based on multiple resolution of grid, Wan et al designed a density grid-based algorithm called MR-Stream (Wan et al., 2009) that computes the summarization information and computes the time interval to extract the cluster. In addition, the

algorithm determines the density threshold based on fading model. It divides the data space to create grid in a tree-like structure. In online phase, when a new data point is arrived, the related grid cell or node is searched. If there is no such node found, then a new node is created, and the information of its parent, grandparent is also updated recursively and the updating operation continues up to the root of the tree. A tree pruning operation is added to identify the sparse grids that become outlier. The offline phase searches the reachable dense grids based on user-defined height and forms cluster. MR-Stream is able to detect the high quality arbitrary shaped clusters in evolving data streams. However, MR-Stream is not scalable to high-dimensional data stream.

For clustering of heterogeneous data stream, ExCC (Bhatnagar et al., 2014) and FGCH (J. Chen et al., 2018) are two excellent density-grid based algorithms. ExCC (Exclusive and Complete Clustering) is designed based on the speed of data stream. The online phase of ExCC keeps synopsis in the grids and offline phase forms the final clusters on demand. The algorithm maps the numerical and categorical attributes of data points to the grids and domain sets respectively. ExCC utilizes the wait and watch policy to detect noise in the offline phase. The algorithm determines the density threshold using granularity of grid and data dimensionality. It generates the clusters from the pool of dense and recent grids. The major advantage of ExCC includes the capability of handling heterogeneous attributes (numeric and categorical). However, the algorithm requires more memory as it uses the hold queue and pool strategy and more processing time as every attribute is handled differently. On the other hand, FGCH (Fast and Grid based Clustering for Hybrid data stream), a fast data stream clustering that uses the non-uniform attenuation model to initialize the grids. The online phase of this algorithm computes the grid tuple information based on the attenuation coefficients. In offline phase, a distance matrix for data points is computed based on the frequency and inter-dimensional correlations. The algorithm shows good clustering accuracy and purity with low data processing time. However, the algorithm cannot handle the drifting properties of data stream efficiently.

For clustering of high-dimensional data stream, PKS-Stream is a popular density grid-based algorithm that is designed based on an assumption that there exist many empty grid-cells for high dimensional data stream. The algorithm maintains the non-empty grid using PKS-tree and k-cover concept. A grid is said to be k-cover, if it meets the density threshold requirement. The online phase of PKS-Stream algorithm maps the

data points to a grid cell in the PKS-tree and the offline phase forms the clusters based on the dense neighbouring grids in PKS-tree. In each time interval, the sparse grids are identified as outliers and deleted from PKS-tree. PKS-Stream shows good performance over high dimensional data stream. However, the algorithm suffers from the inefficient pruning on the tree after adding a new data point to any of the cells of the tree.

2.4.1.2 Online Density Grid-based Clustering

Density grid-based clustering algorithm is a hybridization of density based clustering and grid based clustering. A fully online mode of this category is refers to the fact that the algorithms completes the clustering task in a single task. A graph-based single pass clustering algorithm called DUCstream (Data Stream Clustering Based on Dense Units) (Gao et al., 2005) was proposed by Gao et al. for data stream and considered as the primitive grid-based clustering algorithm for data stream. DUCstream assumes the arrival of data in chunks and generates dense unit with some data points. This algorithm also introduces the concept of local dense unit which is a candidate for dense unit. Each dense unit is considered as a vertex of a connected graph. The vertices of this graph indicate the neighborhood between two dense units. When a new dense unit is formed, it is either create a disjoint graph or participate to the existing graph. In case of disjoint graph, the dense unit creates a new cluster; otherwise, it is assigned to an existing cluster. The clustering statistics are reflected by clustering bit string represents the number of dense units where 1 and 0 states for dense and non-dense unit. The clustering result is updated by an incremental process. DUCstream takes low processing time and the memory space by adapting the bitwise clustering. However, the success of this algorithm heavily depends on size of chunks of data which depends on the user to determine.

Recently, another density grid-based clustering algorithm called DGB (Density grid-based) is proposed (B. Wu & Wilamowski, 2017). The algorithm automatically computes the number of clusters and detects noise. In addition, a new method of finding mountain ridges (Vallim et al., 2014) of a cluster is introduced. Instead of simply counting the data points, a soft decision strategy is proposed to compute nodes density. DGB detects the outliers efficiently and produce arbitrary shaped cluster. In addition, it decreases the processing time. However, the algorithm suffers low cluster quality (accuracy and purity) as the algorithm cannot handle the case of the non-uniform density distribution of data in data space. Thus the cluster quality needs to be improved.

2.4.2 Density Micro-clustering Algorithms

Micro-clustering is a popular clustering approach in the field of data stream to summarize the temporal locality of data points. The concept of micro-cluster was first introduced in (T. Zhang et al., 1996) for large data base and successfully applied to data stream in (Aggarwal et al., 2003).The basic idea of density micro-clustering algorithms is to store and update the synopsis information about the data stream in a metadata called micro-cluster. Micro-cluster is a method to keep statistical information about the data locality. It can adjust well with evolution of the underlying data streams. Such a method can be used to filter out noise or outliers and to discover clusters of arbitrary shape. Figure 2.5 illustrates the micro-clusters and clusters. Micro-cluster extend the concept of cluster feature (CF) (T. Zhang et al., 1996) that maintains the triple vector to summarize the clustering information. Researchers have proposed many density micro-clustering algorithms which can discover arbitrary shaped clusters efficiently.

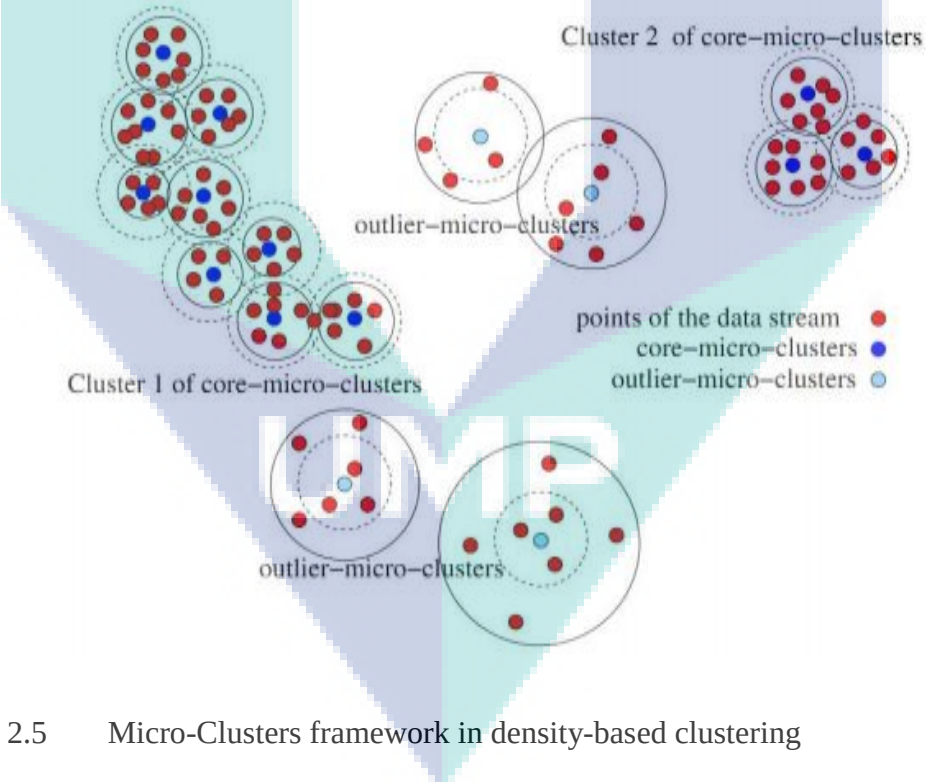


Figure 2.5 Micro-Clusters framework in density-based clustering

Some of the algorithms are online-offline and others are online clustering. The basic concept of online-offline approach and online approach is explained in details in Section 2.3.2.1 and Section 2.3.2.2. In the following section (Section 2.4.2.1), the evolution of popular online-offline micro-clustering algorithms and online micro-clustering algorithms are described in Section 2.4.2.2.

2.4.2.1 Online-offline Density Micro-clustering

Micro-clustering is a popular method in clustering of data stream to summarize the data stream effectively and to maintain the temporal locality of data points. Micro-clusters is the temporal extension of cluster feature (CF) (T. Zhang et al., 1996) for data stream. In the field of density micro-clustering, DBSCAN (Ester et al., 1996) is considered to be a primitive algorithm which was developed for large spatial dataset but also adopted for data stream. The clustering algorithm creates micro-clusters and then clusters in high-density regions based on density neighborhood in a two-phase process. A micro-cluster represents the summarization of data points in it. In online phase, DBSCAN recursively selects a data point randomly and create micro-clusters by searching its neighborhood. In offline phase, the micro-clusters are used to generate final clusters. DBSCAN has the ability to find arbitrarily shaped clusters and it is robust to outliers. However, the algorithm is not entirely deterministic as any points those are reachable from more than one cluster, can be part of either cluster. Moreover, it is not preferable for high dimensional data set as it suffers from the so-called "curse of dimensionality" difficulty and it takes much memory space for loading the whole dataset in memory. The performance of DBSCAN is improved in G-DBSCAN (Kumar & Reddy, 2016) based on a graph theory (Zahn, 1970). The algorithm utilizes a graph-based index structure of groups to decrease the neighbour searching time. Rather than searching the entire patterns of data, G-DBSCAN uses the group method of searching pattern where patterns are grouped using graph-based structure. Similar to DBSCAN, the algorithm runs in two phases; the online and the offline phase. G-DBSCAN improved the processing time and can detect the outliers efficiently. However, G-DBSCAN is not evolving and not scalable to high-dimensional data stream.

Based on DBSCAN, several online-offline clustering algorithms are introduced in the literature. DenStream(Density Based Data Stream Clustering) (Cao et al., 2006) and CluStream (Aggarwal et al., 2003) are two most popular among them. They work as the basic algorithm for many new micro-clustering algorithms. DenStream is able to discover arbitrary shaped clusters for evolving data stream with noise. The clusters are then created based on these micro-clusters. DenStream defines three types of micro-clusters and they are core micro-cluster, potential micro-cluster, and outlier micro-cluster. The core micro-clusters are used to create the clusters with arbitrary shape. Any

potential micro-cluster with weight above the threshold weight is considered as the core micro-cluster whereas the micro-cluster which has the weight less than the threshold weight is defined as the outlier micro-cluster. To handle the evolving nature, the weight of each micro-cluster is reduced exponentially with time using a fading function. It is a hybridization of online and offline framework. In online phase, it uses the DBSCAN algorithm to build the initial potential micro-clusters based on neighbourhood of data points. In the offline phase, DenStream adopts DBSCAN to find the final cluster from the current potential micro-clusters. It also uses a pruning to identify the real outlier-micro-cluster in outlier buffer based on the weight of the outlier micro-cluster. The algorithm also defines a density threshold function where the density threshold is measured. Any micro-cluster with density below the density threshold is considered to be the real outlier micro-cluster and removed from the outlier buffer. Though DenStream has the ability to handle the evolving data stream effectively but it does not release any memory space by either removing or merging micro-clusters until the pruning phase. Furthermore, the pruning of outlier micro-cluster is a time-consuming task. On the other hand, CluStream uses k-means algorithm for clustering evolving data streams. The algorithm starts by an offline process where the initial micro-clusters are created using a standard k-means algorithm from a predefined number of data points from data stream. In the online phase, these initial micro-clusters are used to cluster the later data stream. When a data point arrives, it is mapped to a micro-cluster based on the distance from data point to micro-cluster center. The data point lies in the closest micro-cluster and the micro-cluster information is updated. CluStream frees up the memory space by either merging two micro-clusters or deleting an old micro-cluster as outlier. The offline phase generates the macro-cluster or simply cluster from the current micro-clusters in memory to summarize the statistics of the micro-clusters. CluStream is effective for both evolving and core streams. It shows high data point processing rate and linear scalability with data dimensionality. The major downside of this algorithm is that it is unable to generate cluster of arbitrary shape as the k-means focuses more on detecting spherical clusters even. Moreover, it is inefficient to apply on high-dimensional data stream.

The performance of DenStream is further improved in C-DenStream (Ruiz et al., 2009) and rDenStream (Liu et al., 2009). C-DenStream (DenStream with Constraints) creates arbitrary shape clusters with constraint. The algorithm extends the instance-level

constraints from static data to data stream. The constraints consist of domain-related knowledge. In C-DenStream, the constraint instructs the data or instances towards the clustering i.e. whether the data belong to the same micro-cluster (Must-Link constraints) or to a different micro-cluster (Cannot-Link constraints). Finally, based on the generated micro-cluster, the clusters are formed using C-DBSCAN algorithm (Ruiz et al., 2007). The algorithm takes advantage when the domain expert has prior knowledge about the group membership and thus it is very useful for those applications. However, C-DenStream cannot handle high dimensional data stream, requires an expert of the application to define the constraints and cannot solve the limited memory space issue. On the other hand, rDenStream was developed specially for very noisy data stream applications. rDenStream executes in three steps micro-clustering, macro-clustering and retrospect learning. In the first step, potential micro-clusters and outlier micro-clusters are formed in online mode using the same approach as in DenStream. Only the potential micro-clusters are forwarded as input to the next step, while outlier micro-clusters are placed in historical outlier buffer. The next step uses DBSCAN approach where all the generated potential micro-clusters are used to produce macro-clusters or simply clusters based on the density threshold. The final step is called the retrospect step. In this step, the misinterpreted outlier micro-clusters are learned again to increase the robustness of the clustering. The clusters form a classifier which is used to re-learn the outlier micro-cluster in the historical outlier buffer. rDenStream extracts knowledge pattern from the initially arriving data stream and improves the clustering accuracy through a re-learning process. The downside of this algorithm is that it requires high processing time in re-learn step and extra memory space to store outlier micro-clusters.

For clustering of heterogeneous data stream, the two popular online-offline algorithms are HDenStream(Density based Clustering over Heterogeneous Data Stream) (Lin & Lin, 2009) and Str-FSFDP (J. Y. Chen & He, 2016). In HDenStream, the data points are defined by two kinds of attributes continuous attributes and categorical attributes which may be important in distinguishing the clusters. Beside deriving the concept of core micro-cluster, potential micro-cluster, outlier micro-cluster from DenStream, the separate distance measures between data point to data point, data point to micro-cluster or micro-cluster to micro-cluster are adopted from HCluStream (C. Yang & Zhou, 2006). This algorithm maintains a two-dimensional (2D) array to store the frequency of categorical attributes. This algorithm is quite similar to

DenStream with online and offline phases and the pruning phase. Like DenStream, HDenStream also can identify arbitrary shaped clusters and provides high cluster quality. But this algorithm does not describe the idea to store the categorical attributes in an efficient way. On the other hand, Str-FSFDP computes the cluster centers automatically. A new micro-cluster vector is introduced for storing and updating the summarization information of mixed data dynamically. A new micro-cluster decay function and deletion mechanism has been introduced to handle the evolving behaviour of data stream. In the offline stage, Str-FSFDP determines the micro-cluster centres based on field intensity, linear regression and residuals analysis. In the online stage, the data points are mapped to the micro-cluster based on its field intensity and micro-cluster centers. The micro-cluster decay function and micro-cluster removing strategy are executed on the micro-clusters to detect the drift in data stream. Str-FSFDP generates arbitrary shaped clusters and can detect outliers efficiently for heterogeneous data stream. However, the algorithm needs further effort to improve the cluster quality.

Two sliding window based clustering algorithms called SDStream (Density-based Clustering over Sliding Windows) (Ren & Ma, 2009) and CC_TRS (Riyadh et al., 2017) were proposed. SDStream (Density-based Clustering over Sliding Windows) was proposed based on the idea of analyzing the most recent data stream and the data points are removed those are not in current window. In the online part, the new data points are added to the either any potential micro-cluster or to an outlier micro-cluster in main memory. The micro-clusters are stored in the form of Exponential Histogram of Cluster Feature (EHCF) in main memory. A set of data points with time stamp form a temporal cluster features (TCF) and a set of ordered TCF form an EHCF. The memory is freed up by merging micro-clusters or deleting an outdated outlier micro-cluster based on timestamp. In the offline part, DBSCAN algorithm is executed on the potential micro-cluster in memory to generate the clusters of arbitrary shape. SDStream is concerned about the user's interest in the distribution of most recent data stream. Processes the most recent data and summarizes the old data by using sliding window model. It can handle noisy environment and evolving nature of data stream. But SDStream cannot handle high dimensional data stream. Moreover, it does not explain properly the main usage of exponential histogram. On the other hand, CC_TRS (Continuous Clustering of Trajectory Stream) was specially designed for clustering of trajectory data streams based on micro-cluster life. The online phase summarizes the

spatiotemporal data stream into temporal micro clusters and the offline phase generates clusters based on micro-clusters in a response to user request. Similar temporal micro-clusters are merged when the size of occupied memory exceeds a given memory space. CC_TRS provides high quality clusters with low data processing time. However, the algorithm takes high memory space the data structure of temporal micro cluster has extra temporal fields.

By combining the advantages of density clustering and affinity propagation clustering (Frey & Dueck, 2007), two online-offline micro-clustering algorithms called APDenStream (Affinity Propagation and Density Based Clustering) (J. P. Zhang et al., 2013)and ADStream(Adaptive Density Based Clustering) (Ding et al., 2016). APDenStream uses the decay density to handle the evolving features of data stream. To generate and maintain micro-cluster information, it uses the online dynamic delete mechanism. The algorithm also adapts the WAP (Xiangliang Zhang et al., 2009) algorithm to detect new class patterns which is absorbed in the clustering model. The online phase finds the micro-cluster for a newly arrived data point or passes it to the Reservoir memory and updates the micro-cluster metadata. The offline phase is invoked by the user to generate clusters in the reservoir and merge this result with the model result within every time stamp. APDenStream generates good quality clusters, particularly in noisy environment. The downside of this algorithm is that it takes high memory space to define the Reservoir and not applicable for high dimensional data stream. On the other hand, ADStream detects the initial cluster automatically by passing messages to data points in data stream. The online-phase generates the micro-clusters by analyzing the dynamic data stream in a sliding window and applying affinity propagation method. The offline phase forms macro-cluster using the micro-clusters at different time granularities. ADStream shows impressive performance in detecting clusters in complex hybrid data streams. But this algorithm suffers from the negative effect of noise on performance in complex data streams.

For clustering of high-dimensional data stream, DenStream is extended in HDDStream(Clustering over High Dimensional Data Stream) (Ntoutsi et al., 2012) and PreDeConStream (Hassani et al., 2012). While the data points and dimensions are summarize in form of micro-cluster in the online phase, the offline phase use PreDeCon(Bohm et al., 2004), a projected clustering algorithm, to produce the final

clusters. HDDStream introduces the concept of prefer vector to maintain micro-clusters. The prefer vector is computed based on variance where data points are denser along this dimension in a cluster. A micro-cluster with prefer vector is refer to as projected micro-cluster. Initially, an initial set of potential projected micro-clusters are created by applying PreDeCon on the predefined amount of data points from the data stream. In the online phase, the data points are assigned to a potential projected micro-cluster by updating the prefer vector; then finding the closest potential projected micro-cluster to the data points and finalizing the operation without affecting the natural boundary. In offline phase, from the generated potential projected micro-cluster, the final clusters are generated. HDDStream is able to able to handle high-dimensional data stream and create cluster with high quality. The major disadvantage of this algorithm is that it cannot handle the evolving nature of data stream properly as it does not check the prefer vector during pruning. Similar to HDDStream, PreDeConStream also uses the concept of prefer vectors for subspace using the variance of micro-clusters and their neighbours. A weight of data points is used to recognize changes in the data stream quickly which is calculated using a fading function. Three types micro-cluster are maintained in this algorithm and they are core micro-cluster, potential micro-cluster and outlier micro-cluster. The improvement is done in the pruning time where the pruning is done both on newly added or deleted potential micro-cluster. The neighbours of these both types of micro-clusters are checked for updating the subspace prefer vectors and are kept in a separate list called affected micro-clusters. This generated list is used to expand the clusters. Though PreDeConStream improves the efficiency but the pruning phase suffers from the time penalty for searching the affected neighbouring clusters.

2.4.2.2 Online Density Micro-clustering

Density micro-clustering technique has drawn remarkable attention of researcher due to its exceptional performance over data stream. However, most of the algorithms are online-offline clustering and the field has the scarcity of online algorithms. In the field of density micro-clustering, SOStream (Isaksson et al., 2012) is an excellent online algorithm that adapts the density threshold to form the clusters. SOStream (Self-organizing Density-based Clustering) uses the online competitive learning concept (Kohonen, 1982) where the winner cluster influences its neighbour micro-clusters. The cluster creation, merging and deleting processes are online in SOStream. When a data

point appears, it is mapped to a micro-cluster based on the distance between the data point and all existing clusters. The micro-cluster with minimum distance is said to be winner cluster and the data point belongs to the winner cluster. The micro-cluster information and the density threshold is recursively updated. The neighbourhood of micro-clusters is defined and they are merged if neighbourhood distance is less than the merge-threshold distance. SOStream achieves better clustering quality with occupying less memory. Though SOStream can adapt the threshold but the competitive learning part suffers from time penalty which makes SOStream unsuitable for data stream clustering. Moreover, the algorithm is not fully evolving.

ELM (Baruah & Angelov, 2012) and DEC (Baruah & Angelov, 2014) are two online evolving clustering techniques. ELM(Evolving Local Means) is designed based on the mean-shift algorithm for data stream. In this algorithm, clusters are summary of data points in data stream which consists of two elements; a cluster centre and a distance parameter. When a data point from the data stream arrives, ELM learns from either the scratch or existing clusters. The cluster information is updated recursively by shifting the mean and distance parameter. After shifting the mean, if the cluster overlaps with another cluster then they are merged based on the neighbourhood distance. ELM is an online algorithm of stream clustering which provides high cluster purity but does not describe the strategy to remove outdated cluster which is required for evolving clustering. On the other hand, DEC (Dynamically Evolving Clustering) was designed based on concept from computational geometry. DEC defines the cluster as a group of data points which are bounded by a hypersphere with a centre and a radius. The cluster summarizes the data points in forms of with a feature vector where the weight of a cluster decreases with time. DEC defines a threshold of weight to distinguish the core cluster (weight above the threshold) and noncore cluster (weight below the threshold). When a data point from data stream arrives, the nearest existing core cluster is searched. The weight of every cluster is updated at every timestamp called the inspection time. After every timestamp, the core and non-core clusters are checked to change their status. To distinguish the outlier from non-core cluster, DEC also defines a lower limit of weight threshold. Any non-core cluster, having the weight below this limit, is identified as outlier and deleted from memory. The evolving nature of cluster in DEC algorithm save the memory space and improve the processing time. But the technique raised an issue to select the optimal value of cluster radius and adaption.

Inspired from bio-nature, two well-known density micro-clustering algorithms namely FlockStream (Forestiero et al., 2013) and ACSC (Fahy et al., 2018) was introduced. FlockStream was intrucued based on flocking model (Eberhart et al., 2001) that defines agent in form of new data point and two types of micro-cluster (potential micro-cluster and outlier-micro-cluster). When a new data point arrives, FlockStream search for similar micro-cluster by checking whether the sub-space of other micro-cluster overlaps with the visibility distance of the data point. The micro-clusters can shift in the virtual space up to a predefined threshold for a specific time according to some rules such as cohesion, separation, and alignment (Forestiero et al., 2013). The overlapping micro-clusters form the micro-cluster representative (simply cluster). Though the data point processing rate of FlockStream is high and forms outlier micro-cluster to handle noise, it does not clarify when and how to remove the outliers micro-cluster from memory. Based on ant colony optimization (ACO) (Dorigo et al., 1996), ACSC(Ant Colony Stream Clustering) is proposed. It offers a single pass tumbling window model(Li et al., 2005) to form clusters incrementally. Like other micro-cluster based clustering, it also summarizes the clustering using micro-clusters. A stochastic method was introduced to find the rough clusters and they are refined by a method which was designed based on the observed sorting behavior of ants. ACSC is scalable, robust to noise and generate high quality cluster. It requires less computational time. But the algorithm is not able to find the clusters of similar density only.

Based on representing the micro-cluster with graph, two recent micro-clustering techniques are CODAS (Hyde & Angelov, 2015) and CEDAS(Hyde et al., 2017). CODAS (Clustering Online Data-streams into Arbitrary Shape) is a data-driven algorithm which generates the micro-cluster to summarize the data points in it. The micro-cluster consists of a centre, radius, and density. The micro-cluster consists of core region covered by inner half of the radius and non-core region covered by outer half. When a data point from data stream arrives, it falls into either empty region or a micro-cluster region. In case of empty region, the data point creates a new micro-cluster itself. Otherwise, the data point is assigned to the micro-cluster and it's information is updated recursively. The micro-clusters are presented using the clustering graph and the clusters are generated from the graph. A micro-cluster with local density below the threshold is referred to as outlier. CODAS generates high cluster quality and it is scalable to multi-dimensional data stream. However, the generated cluster does not evolve. CODAS is

improved in CEDAS (Clustering of Evolving Data-streams into Arbitrary Shapes) for detecting the evolving behaviour by introducing a simple aging process. The micro-clusters also include the energy. The energy of micro-cluster is maintained by a simple aging process. The aging process confirms the removal of old micro-cluster. Similar to CODAS, the clusters are mapped, updated and presented using the clustering graph. In addition, the energy of all other micro-cluster is decreased by an amount depending on decay of application. Micro-cluster with local density below density threshold and negative energy is marked as outlier and removed from the memory. Also, micro-clusters with negative energy but local density above the density threshold are considered as old micro-cluster and removed from the clustering graph and from memory. Every time a micro-cluster is modified, the clustering graph is updated and clusters are re-generated. This task confirms the immediate access to clustering result. CEDAS is a fully online clustering algorithm for evolving data stream. But the algorithm suffers from two major problems. It is difficult and erroneous task to select the optimal value of cluster radius. The linear aging process and immediate removal of micro-cluster affects the clustering quality as some deleted micro-clusters are significant.

2.5 Summary of Literature Review

The above discussed density based clustering algorithm are summarized and compared in Table 2.3. The major features for comparing the algorithms are the clustering summarization method, data processing method and generated cluster shape, the ability of handling the evolving nature and the noisy behaviour of data stream and whether the technique is applicable for high dimensional data stream. The techniques are also compared in terms of its advantages and disadvantages. The advantages and disadvantages are analyzed with respect to cluster quality, memory requirement and completeness of the technique.

According to the Table 2.3, the density based clustering algorithms are either density grid based or density micro-clustering types. The clustering algorithms only maintain the summary of data points instead of storing all data points from the data stream. The density grid clustering algorithms store the synopsis information in the grids whereas density micro-clustering algorithms store the data stream summary information in micro-clusters. It is desired that a clustering algorithm is able to

generated arbitrary shaped clusters. Most of the density based clustering algorithms generate arbitrary shaped clusters except CluStream, HDDStream, ELM, DEC and PKS-Stream. They generate either spherical or hyper-ellipsoidal shaped clusters. It is desired to have fully online method for clustering of data stream. However, major algorithms in Table 2.3, are hybridization of online and offline phases in their execution. The micro-cluster mapping is done in an online manner. However, the clustering results are generated in an offline phase using the generated micro-clusters. Some algorithms like rDenStream, FlockStream, ELM, DEC, CODAS and CEDAS are online density based clustering algorithms. They generate cluster in an online manner where the clustering results are immediately available. Evolving is one of the vital properties which indicate that the current content of data stream may not be relevant always and this property should be handled carefully.

In Table 2.3, in most of the density based clustering algorithms handle this property where the generated micro-clusters evolve with time. This evolving is provided either by maintaining energy of micro-clusters or by maintaining a timestamp. After expiring the timestamp or energy, the micro-clusters are marked as unusable and immediately removed from the result. However, two algorithms namely DenStream and CODAS are not evolving in nature. The generated micro-clusters are not removed from these algorithms. An ideal clustering algorithm should be should be scalable. This property describes that the algorithm is applicable to low to high dimensional data streams. About 50% of the discussed algorithms like CEDAS, DD-Stream, DCU-stream etc. are scalable and half of them are not. The non-scalable algorithms cannot be applied on high dimensional data stream applications like genetic data stream, satellite data stream. Most of the natural data streams are not cleaned and noisy samples are present in the stream. Thus the clustering algorithms should consider the presence of noise in data stream. Most of algorithms can detect the noisy data points except DEC. The noisy samples are removed and they don't participate in cluster summarization updating. This property keeps the cluster summary accurate and free from noise.

Table 2.3 Summary of reviewed density-based algorithms for data stream

Clustering Algorithms	Working Principle	Cluster Shape	Data Processing	Evolving	Scalable	Noise detection	Major Advantage	Major Limitations
ACSC (Fahy et al., 2018)	Micro-clustering	Arbitrary	Online	√	√	√	Less computational time.	Limited to find the clusters of similar density only.
ADStream (Ding et al., 2016)	Micro-clustering	Arbitrary	Online-offline	√	√	√	Process complex hybrid data streams.	Negative impact of noise on performance.
APDenStream (J. P. Zhang et al., 2013)	Micro-clustering	Arbitrary	Online-offline	√	×	√	Good performance in noisy environment	Takes high memory space to define the reservoir.
CC_TRS (Riyadh et al., 2017)	Micro-clustering	Arbitrary	Online-offline	√	×	×	Very effective for trajectory data stream.	Requires high memory space.
C-DenStream (Ruiz et al., 2009)	Micro-clustering	Arbitrary	Online-offline	√	×	√	Very useful in the applications with knowledge on the group membership.	Needs an expert to define its constraints.
CEDAS (Hyde et al., 2017)	Micro-clustering	Arbitrary	Online	√	√	√	Efficient draft handling.	Heavily depends on the user-defined parameter.
CluStream (Aggarwal et al., 2003)	Micro-clustering	Spherical	Online-offline	√	×	√	High data processing rate.	Inefficient to apply on high-dimensional data stream.
CODAS (Hyde & Angelov, 2015)	Micro-clustering	Arbitrary	Online	×	√	√	High cluster quality.	The unused old micro-clusters are not removed.
D-Stream (Y. Chen & Tu, 2007)	Density Grid	Arbitrary	Online-offline	√	×	√	Low time complexity and high cluster quality.	Inefficient way to define time gap.
DCUStream (Y. Yang et al., 2012)	Density Grid	Arbitrary	Online-offline	√	√	√	Clustering uncertain data	High time complexity

Table 2.3 Continued

Clustering Algorithms	Working Principle	Cluster Shape	Data Processing	Evolving	Scalable	Noise detection	Major Advantage	Major Limitations
DD-Stream (Jia et al., 2008)	Density Grid	Arbitrary	Online-offline	√	√	√	High cluster quality.	High time complexity
DEC (Baruah & Angelov, 2014)	Micro-clustering	Hyper-ellipsoidal	Online	√	√	×	Require low the memory space and processing delay	Heavily depends on the user-defined parameter.
DENGRIS-Stream (Amini & Wah, 2012)	Density Grid	Arbitrary	Online-offline	√	×	√	First density clustering algorithm for evolving data streams over sliding window model.	No evaluation to show the algorithm effectiveness compared.
DenStream (Cao et al., 2006)	Micro-clustering	Arbitrary	Online-offline	√	×	√	High cluster accuracy	Suffers from the time-consuming pruning operation.
DGB (B. Wu & Wilamowski, 2017)	Density-Grid	Arbitrary	online	×	√	√	Low time complexity.	Not effective for high density clusters.
DUCStream (Gao et al., 2005)	Density Grid	Arbitrary	Online	√	×	√	Low time and space complexity.	Heavily depends on the user-defined parameter.
ELM (Baruah & Angelov, 2012)	Micro-clustering	Hyper-ellipsoidal	Online	√	√	√	High cluster purity	Does not describe the strategy to remove outdated cluster
ExCC (Bhatnagar et al., 2014)	Density Grid	Arbitrary	Online-offline	√	×	√	Clustering heterogeneous data streams	The hold queue strategy needs more memory and processing time.
FGCH (J. Chen et al., 2018)	Density Grid	Arbitrary	Online-offline	√	√	√	High quality cluster and high processing speed.	Require initial density from user and data points are feed with fixed speed

Table 2.3 Continued

Clustering Algorithms	**Working Principle**	**Cluster Shape**	**Data Processing**	**Evolving**	**Scalable**	**Noise detection**	**Major Advantage**	**Major Limitations**
FlockStream (Forestiero et al., 2013)	Micro-clustering	Arbitrary	Online	√	×	√	High processing speed.	No clear strategy to remove the outliers.
G-DBSCAN (Kumar & Reddy, 2016)	Micro-clustering	Arbitrary	Online-offline	√	×	√	Effective in noisy data stream.	Cluster quality is not high.
HDDStream (Ntoutsi et al., 2012)	Micro-clustering	Spherical	Online-offline	√	√	√	High cluster accuracy.	Inefficient pruning and It cannot handle the data in a limited time.
HDenStream (Lin & Lin, 2009)	Micro-clustering	Arbitrary	Online-offline	√	×	√	Ability to work on heterogeneous data stream.	Lack of details about saving categorical features in an efficient way.
MR-Stream (Wan et al., 2009)	Density Grid	Arbitrary	Online-offline	√	×	√	High cluster quality.	Not effective in highly noisy stream.
PKS-Stream (Ren et al., 2011)	Density Grid	Hyper-ellipsoidal	Online-offline	√	√	√	Low time and space complexity.	Does not have any pruning on the tree after adding a new data point.
PreDeConStream (Hassani et al., 2012)	Micro-clustering	Arbitrary	Online-offline	√	√	√	It improves the efficiency of the HDDStream.	Suffer from a time-consuming process for searching the affected clusters.
rDenStream (Liu et al., 2009)	Micro-clustering	Arbitrary	Online	√	×	√	High accuracy.	Memory usage and the time complexity are high.
SDStream (Ren & Ma, 2009)	Micro-clustering	Arbitrary	Online-offline	√	×	√	High accuracy in noisy environment	Lack of clarification about the purpose of using exponential histogram to store micro-clusters.
SOStream (Isaksson et al., 2012)	Micro-clustering	Arbitrary	Online	√	×	√	Good clustering quality occupying less memory.	Suffers from the time-consuming method, SOM (Self Organizing Maps).
Str-FSFDP (J. Y. Chen & He, 2016)	Micro-clustering	Arbitrary	Online-offline	√	×	√	Process mixed data with lower time complexity.	Lower clustering quality.

It can be seen from the Table 2.3, ACSC, HDDStream, FGCH, CODAS, SDStream, APDenStream, and ExCC show excellent performance in terms of cluster purity. The purities of these algorithms are more than 96%. Some algorithms like CEDAS, PreDeConStream, SOStream, DenStream, FlockStream, G-DBSCAN, MR-Stream, rDenStream are able to generate good clusters with 91-95% cluster purity. On the other hand, the rest clustering algorithms in Table 2.3 provide moderate quality clusters. The purities of these clusters are less than or equal 90% like the purity of ADStream, DStream, DCUStream, ELM, H-DenStream, Str-FSFDP are 88%, 90%, 81%, 87%, 90% and 89% respectively. The accuracy measures for some of these algorithms are available. The accuracies of these algorithms are close to the purity values of these algorithms. Like the accuracies of ACSC, CODAS, CEDAS, ADStream, APDenStream, DCUStream and DDStream are 98%, 98.9%, 96.5%, 86%, 98.5%, 82% and 93.5% respectively. CC_TRS, FSFDP, ADStream, SOStream, DD-Stream take very less processing time per data, whereas PreDeConStream, CODAS, SDStream, HDDStream, FGCH, MR-Stream, rDenStream is slow algorithms. The rest algorithms like ELM, DEC, CEDAS, D-Stream, C-DenStream, ExCC require moderate data processing time to complete the clustering task. The scalability results show that CEDAS, DenStream, CC_TRS, DStream, ELM, H-DenStream and MR-Stream is scalable to high dimensional data stream.

To summarize, every algorithm has its own advantages and limitations. Most of these algorithms are not fully online algorithm and suffers from the consuming excessive memory space. Few algorithms like ELM, DEC, CODAS, CEDAS are the fully online algorithm in Table 2.3. However, some of them take more processing time or excess memory or unable to generate arbitrary shaped clusters. All of the density-based clustering algorithms maintain the global and constant value of micro-cluster radius. This fact contributes to leave some sparse regions in the micro-cluster and the cluster quality is degraded as a result. Moreover, the micro-clusters are created and deleted frequently that increases the processing time of clustering. The issues have been described in details (Section 1.3, Chapter 1). Adapting the concept of local optimal radius and preventing the frequent creation and removal of micro-clusters can solve these issues. Thus it is still an open research issue to provide a fully online clustering algorithm makes a trade-off among clustering quality, processing time and memory requirement.

CHAPTER 3

METHODOLOGY

3.1 Introduction

Extracting the knowledge or information from data stream is becoming more and more useful in real time applications. Clustering is a method of extracting summary information of data stream that helps the companies towards real-time decision-making (Yu et al., 2013). For illustration, clustering trajectory data stream helps the drivers to know the congestion on road at any time in traffic management system. Continuous clustering of patient movements can help the doctors to predict the condition of patient in hospital. In Chapter 2, already several density based data stream clustering algorithms have been discussed with their basic working principle. The flaws of each of them are also identified and discussed. Most of the density-based clustering algorithms are not fully online. Though, few algorithms are online but they suffer from several problems which are tabulated in Table 2.3, Chapter 2. In the field of density-based clustering, the research gaps are discussed in details in Section 1.3, Chapter 1. Several objectives are set in Section 1.4, Chapter 1 to mitigate the problems.

In this chapter, the developed BOCEDS algorithm is described in details. The data structure of the developed BOCEDS algorithm is visualized and explained in Section 3.2.1. The flowchart of the algorithm has been drawn as Figure 3.2 in Section 3.2.2. The steps of the algorithm are also described in details by formulating the solution and providing the algorithmic presentation. Finally, Section 3.3 concludes the chapter by summarizing the algorithm, the way of mitigating the problems by the algorithm.

3.2 Developed Algorithm

The goal of the developed BOCEDS is to provide high cluster quality and low memory requirement and processing delay and detect the noise and evolving characteristics of data points in a data stream. BOCEDS is a single phase clustering algorithm that generates clusters from data stream in a fully online manner. The algorithm stores the data summary information in a data structure called micro-clusters. In case, a data doesn't fall into any micro-clusters in the current model, it creates a new micro-cluster itself. The data structures are updated every time a data arrives from the data stream in a fully online manner. Along with summary information, the micro-clusters also maintain an energy level to bear the timing information about the last change. The energy of a micro-cluster is increased every time it receives a new data and otherwise decreased. The micro-cluster with non-positive energy is considered as irrelevant micro-cluster and move from main memory to a special memory, called buffer. This operation confirms the correct functionality of BOCEDS to evolving data stream. In case, a new data maps to a micro-cluster in buffer then it is marked as relevant micro-cluster and move to main memory again. The micro-clusters in memory generate micro-clustering graphs based on their connectivity. A single graph forms a cluster.

Section 3.2.1 describes the data structures used in the developed BOCEDS algorithm in details. The algorithmic parameters and the steps are discussed in Section 3.2.2. The sub-algorithms are also presented in this section.

3.2.1 Data Structures in BOCEDS

The developed BOCEDS is an online micro-clustering density-based clustering algorithm. Similar to other micro-clustering density techniques, it summarizes the clustering information in the form of micro-clusters. The macro-clusters are generated based on membership among the micro-clusters in a clustering graph. Two connected micro-clusters belong to the same macro-cluster. BOCEDS defines the "decay" parameter to detect the evolving property of data and the "minimum density threshold" to differentiate the outliers from the micro-clusters. Decay is defined as the total data points from the data stream that arrive in a period at a specific sampling rate or the number of data points that arrive per unit time (Hyde et al., 2017). The minimum

number of data points required to form a micro-cluster is known as the minimum density threshold ($Th_{density}$). A micro-cluster with a local density less than the threshold is an outlier micro-cluster. Figure 3.1(a) and Figure 3.1(b) show the structure of a micro-cluster and the neighbourhood of micro-clusters, respectively. In Figure 3.1(b), a total of 8 micro-clusters are denoted by $V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8$. Figure 3.1 (c) derives the clustering graph from the micro-clusters intersections (Figure 3.1 (b)). In Figure 3.1 (c), the macro-clusters or simply clusters are denoted by M_1, M_2 and M_3.

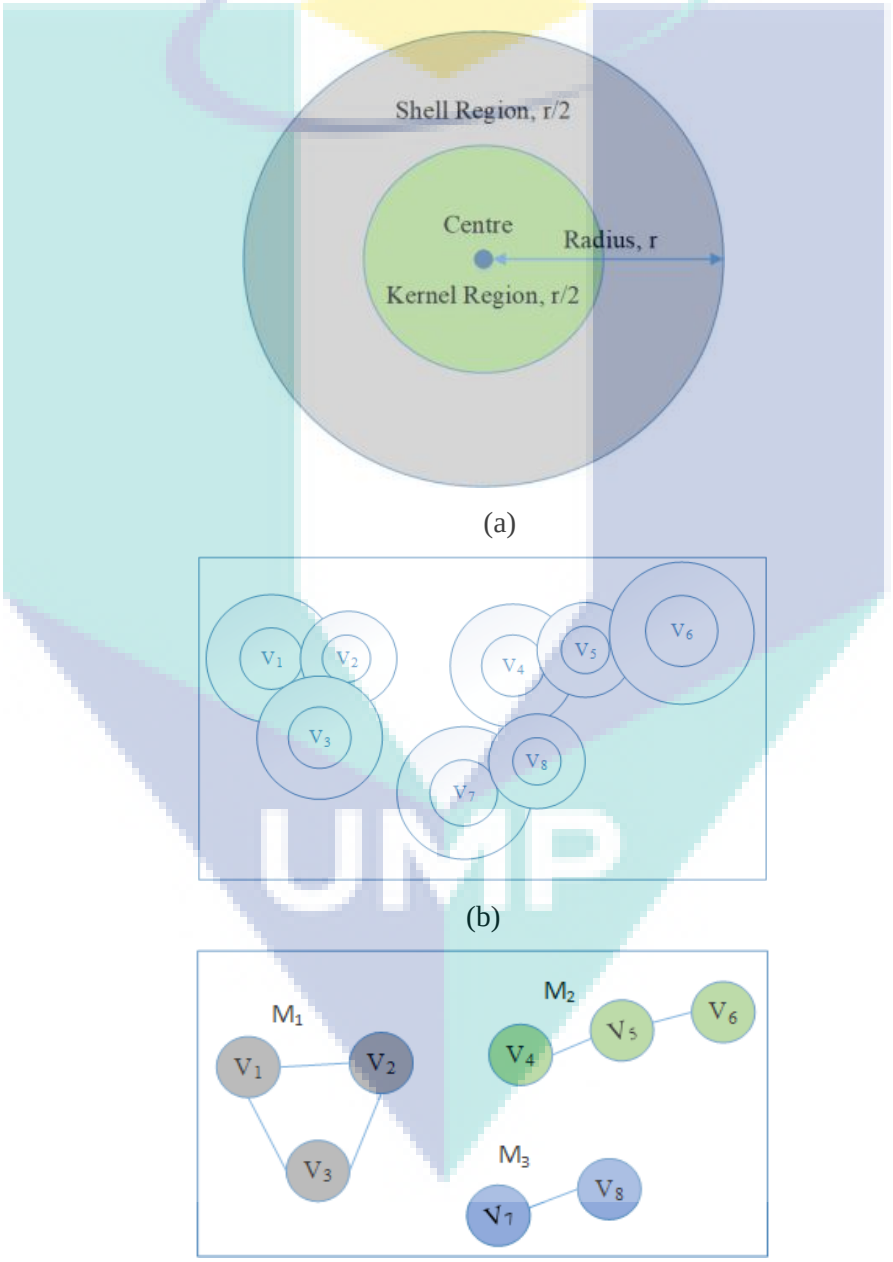


Figure 3.1 The data structure in BOCEDS algorithm (a) Micro-cluster structure (b) Insections of micro-cluster (c) The formation of clustering graph and macro-cluster

A micro-cluster (*MC*) is defined as the tuple $MC(N, N', C, R, E, EL, M)$, where

i. Center (*C*) is the center of the micro-cluster that defines the location of the micro-cluster in the data space. It is computed as the mean of the data points in the micro-cluster (Figure 3.1(a));

ii. Radius (*R*) is the radius of the micro-cluster that describes the spread of the micro-cluster from the center. The inner region covered by half of the radius is known as the kernel region (the grey region in Figure 3.1(a)), whereas the outer half part is called the shell region (the greenish region in Figure 3.1(a)).

iii. Local density (*N*) is the local density of a micro-cluster that describes the number of data points within the micro-cluster radius. N' is defined as the number of data points in the shell region of the micro-cluster.

iv. Energy (*E*) is the energy of a micro-cluster that is defined as the potential of the micro-cluster. It is used to determine the length of time since a micro-cluster receives last data. The energy of every micro-cluster is updated (the details of which is in Section 3.2.2.3) after clustering every data point. A micro-cluster dies or lives on the basis of energy. A micro-cluster with zero or negative energy $(E \leq 0)$ is killed and does not participate in the clustering graph.

v. Edge list (EL) in each micro-cluster shows the connected or edged micro-clusters. Two micro-clusters with radius R_1 and R_2 are considered intersected if the distance (d) between their centers is less than the intersecting distance $d' = (R_1 + \frac{R_2}{2})$. In other words, two micro-clusters are considered edged if the kernel region of a micro-cluster intersects with the shell or kernel region of another micro-cluster. The intersecting micro-clusters collectively form the edge list of a micro-cluster. In Figure 3.1(b), the edge list of V_1 micro-cluster comprises V_2 and V_3 micro-clusters. Meanwhile, the edge list of V_7 micro-cluster consists only of V_8 micro-cluster. As the shell region, V_8 intersects with the shell region of V_4 micro-cluster; thus, they do not belong to the same macro-cluster.

vi. Macro-cluster (*M*): Intersecting micro-clusters form a single macro-cluster. A micro-cluster with local density more than the threshold but with no intersecting

micro-cluster forms a macro-cluster. Figure 3.1(c) illustrates the formation of three macro-clusters (i.e., M_1, M_2, and M_3).

Each time a new data point emerges; it either contributes to form a new micro-cluster or falls into an existing micro-cluster. BOCEDS maintains four types of micro-clusters, namely, core, potential, weak, and outlier micro-clusters. The developed algorithm defines the density threshold ($Th_{density}$; Section 3.2.1) and the micro-cluster energy to distinguish among different types of micro-clusters. The micro-clusters are defined as the following *Definitions 1 to 4*.

Definition 1 (Core Micro-cluster). A core micro-cluster at time t, where $MC_{core}(N_t, N_t', C_t, R_t, E_t, EL_t, M_t)$ is defined as the group of close points $X_1, X_2, \ldots\ldots\ldots X_{N_t}$ in a high-density area where

i. the local density(N_t) is equal or exceeds the density threshold, $N_t \geq Th_{density}$ where $Th_{density}$ is an application-dependent and user-defined parameter;

ii. the number of data points in the shell region (N_t') is less than or equal to the local density, $N_t' \leq N_t$;

iii. the radius (R_t) holds $R_{\min} \leq R_t \leq R_{\max}$, where $R_{\min}$ to $R_{\max}$ is the range of the micro-cluster radius defined by the user;

iv. the living energy is positive, $E_t > 0$;

v. the center ($\forall_{k \in D}, C_t^k = \frac{\sum_{t=1}^{N_t} X_t^k}{N_t'}$) is calculated as the mean of X_t D-dimensional data points in the shell region of the micro-cluster (N_t') ;

vi. the edge list is $EL_t = \{MC_1, MC_2, \ldots\ldots, MC_P\}$, where the micro-cluster is intersected with other P core micro-clusters; and

vii. the macro-cluster id M_t is a unique integer assigned to each intersected micro-cluster from edge list EL_t.

Definition 2 (Weak Micro-cluster). A weak micro-cluster, $MC_{weak}(N_t, N_t', C_t, R_t, E_t, EL_t, M_t)$, in the buffer is defined as the group of close data points $X_1, X_2, ..., X_{N_t}$ in the high-density area at time t, with the local density(N_t) equal or more than the density threshold($Th_{density}$), the number of data points in the shell region (N_t') is less than or equal to local density(N_t), positive energy($E_t > 0$), empty edge list($EL_t = \phi$), and no macro-cluster id ($M_t = 0$). The center(C_t) and radius(R_t) are calculated similar to **Definition 1**.

Definition 3 (Potential Micro-cluster). A potential micro-cluster $MC_{potential}(N_t, N_t', C_t, R_t, E_t, EL_t, M_t)$ is defined as the group of one or more close points $X_1, X_2, ..., X_{N_t}$ at time t, with the local density(N_t) below the density threshold ($Th_{density}$), the number of data points in the shell region (N_t') is less than or equal to local density(N_t), positive energy($E_t > 0$), radius(R_t) equal to minimum radius($R_{\min}$), empty edge list($EL_t = \phi$), and zero macro-cluster id($M_t = 0$). The center(C_t) is calculated similar to **Definition 1**.

Definition 4 (Outlier Micro-cluster). An outlier micro-cluster $MC_{outlier}(N_t, N_t', C_t, R_t, E_t, EL_t, M_t)$ is defined as the group of one or more data points $X_1, X_2, ...X_{N_t}$ in a low-density area at time t, with the local density (N_t) below the density threshold ($Th_{density}$), the number of data points in the shell region(N_t') is less than or equal to local density(N_t), non-positive energy($E_t > 0$), radius(R_t) equal to minimum radius($R_{\min}$), empty edge list($EL_t = \phi$), and zero macro-cluster id($M_t = 0$). The center(C_t) is calculated similar to **Definition 1**.

The algorithm requires a user defined parameter, namely density threshold to define the types of micro-cluster (Section 3.2.2). The parameter denotes the number of minimum data points to differentiate a micro-cluster from the background noise of data stream. A micro-cluster with the local density below the density threshold is considered

as noise or outlier micro-cluster. A density threshold equal to one defines that the data stream contains no noise. On the basis of the four definitions, a micro-cluster needs the local density to be above the user-defined density threshold to be a core (Definition 1) or weak (Definition 2) micro-cluster, whereas the condition is reversed for potential (Definition 3) and outlier (Definition 4) micro-clusters. The micro-cluster center is calculated as the mean of the data points in the shell region only because they prevent the micro-cluster from following the drift of the data stream endlessly by limiting its movement (Hyde et al., 2017). Although core and weak micro-clusters have a positive energy, a core micro-cluster is stored in the primary memory, whereas a weak micro-cluster is stored in a special buffer memory. The core micro-cluster actively participates in cluster graph and has a positive macro-cluster id, whereas a weak micro-cluster does not participate in the cluster graph and no macro-cluster id is assigned to it. Similar to weak micro-clusters, potential and outlier micro-clusters do not participate in the cluster graph and do not have a macro-cluster identification number. Outlier micro-clusters are identified as noise and removed immediately after identification.

3.2.2 Description of the Developed BOCEDS Algorithm

Prior to the execution of the developed BOCEDS algorithm, few application-dependent parameters are defined on the basis of the expert knowledge of the application similar to other micro-cluster density-based clustering techniques, such as DenStream, CluStream, DEC, CODAS, and CEDAS. The developed BOCEDS algorithm defines the following clustering parameters.

i. *Decay* (*Decay*)*:* Decay is the number of data points from the data stream that arrive per unit time. It is the data rate. A decay of 1000 implies that 1000 data points are sequentially coming on an average per unit time (e.g., second, minute) from the data stream. It is used to update the energy of micro-clusters. This value is set based on expert knowledge about the application.

ii. *Maximum* (R_{max}) *and Minimum* (R_{min}) *Radii:* The maximum and minimum radii of micro-clusters are set based on expert knowledge about the application. The maximum radius confirms the separation and smoothness of micro-clusters, whereas the minimum radius confirms the formation of micro-clusters with

sufficient data points. Recently, the authors in (Albertini & Mello, 2018) described an adaptive method for estimating clustering parameters.

iii. *Minimum Threshold* ($Th_{density}$)*:* The minimum threshold is the minimum number of data points required to form a core micro-cluster. This value separates the micro-clusters from the background noise.

After setting the application parameters, the developed BOCEDS algorithm is executed on data stream $X = \{X_0, X_1, X_2, X_3, ...\}$ by the following six distinct steps.

i. Initialize the micro-cluster (Section 3.2.2.1)

ii. Search the target micro-cluster (Section 3.2.2.2)

iii. Update the micro-clusters (Section 3.2.2.3)

iv. Move the weak micro-cluster to the buffer (Section 3.2.2.4)

v. Kill the weak micro-cluster in the buffer (Section 3.2.2.5)

vi. Update the cluster graph (Section 3.2.2.6)

Figure 3.2 shows the developed BOCEDS clustering procedure. The procedure begins by reading the application-dependent clustering parameters ($Th_{density}, R_{\min,} R_{\max}, Decay$). The clustering procedure then waits for the data points (X_i) from the data stream (X).

As show in Figure 3.2, when a data point (X_i) arrives, it searches for the target micro-cluster (*T*), where X_i resides based on the Euclidean distance between the data point and the hyper-spherical micro-clusters. BOCEDS emphasizes on maintaining a hyper-spherical micro-cluster because of its favourable computational characteristics over hyper-ellipsoidal or hyper-box-shaped micro-clusters in terms of dimensional stability and processing time (Hyde et al., 2017). The searching operation is executed on the core (MC_{core}), weak (MC_{weak}), and potential ($MC_{potential}$) micro-cluster sets (Section 3.2.2.2).

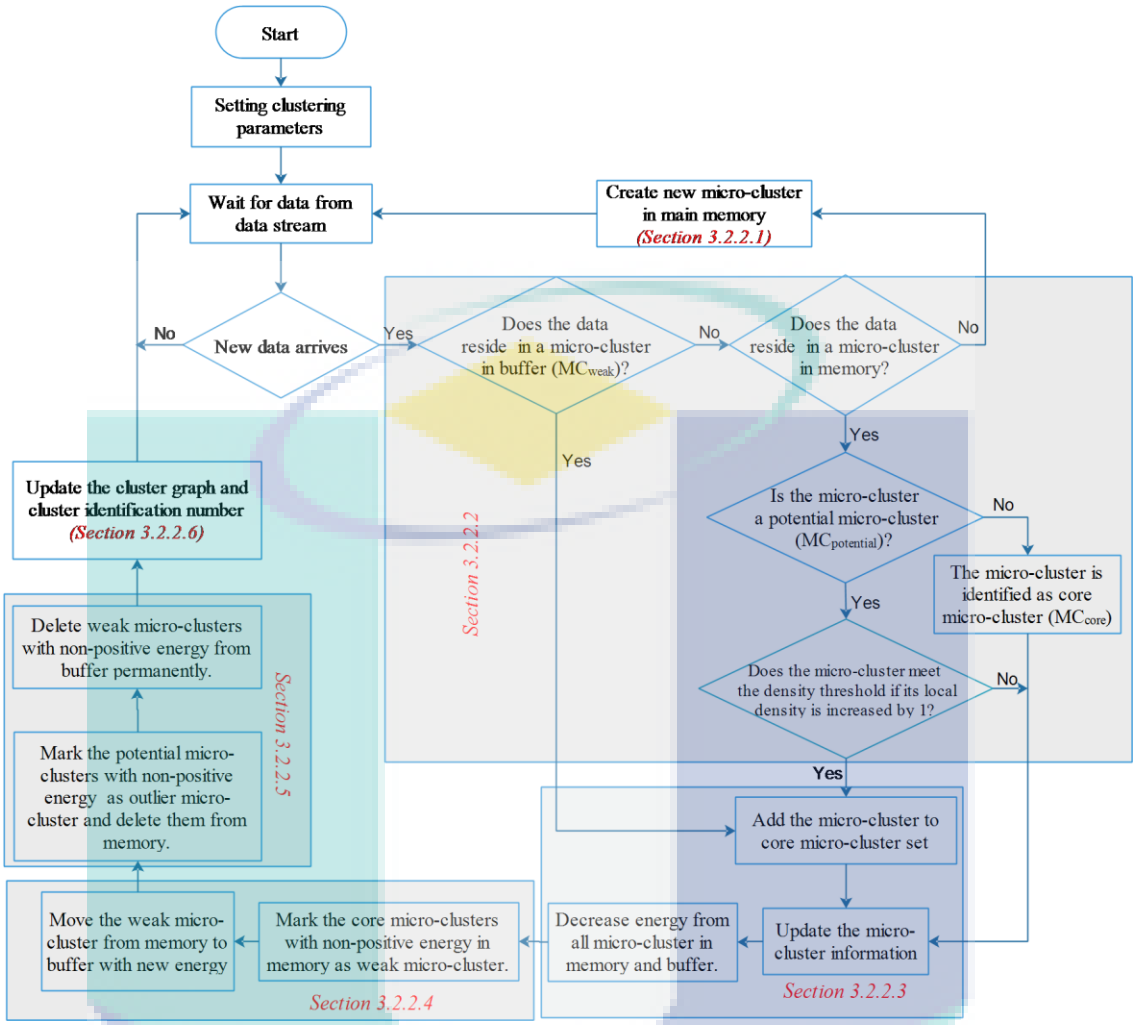


Figure 3.2 Developed BOCEDS clustering algorithm

If the data resides in a weak micro-cluster, the micro-cluster is immediately turns into core micro-cluster. If the data reside in a potential micro-cluster, then the micro-cluster is checked for core micro-cluster set membership and added to the set. Figure 3.2 indicates that in the case of successful searching, the information of the target micro-cluster (T) is extracted and updated (Section 3.2.2.3). Otherwise, a new potential micro-cluster is created (Section 3.2.2.1). The energy of every micro-cluster in the system is updated to find the weak micro-cluster candidate in the core micro-cluster set (Section 3.2.2.4), the outlier micro-cluster in the potential micro-cluster set, and the dying micro-clusters in the weak micro-cluster set (Section 3.2.2.5). Energy (E) represents the energy of micro-clusters (i.e., potential, core, and weak micro-clusters). $E \leq 0$ of a core micro-cluster indicates that the micro-cluster is weak (temporarily irrelevant) and is thus stored in the buffer. Meanwhile, $E \leq 0$ of a weak micro-cluster indicates that the micro-cluster is dying (entirely irrelevant) and is thus removed

completely from the buffer. Finally, E≤0 of a potential micro-cluster indicates that the micro-cluster is dying (outlier). The core micro-cluster with non-positive energy becomes a weak micro-cluster, and the micro-cluster is moved to the buffer memory. Spherical micro-clusters are then represented using the clustering graph, which is finally updated by updating the core micro-cluster set (Section 3.2.2.6). Arbitrarily macro-clusters (or simply clusters) of arbitrary shape are formed using the graph. Thus, although the micro-clusters are spherical, they generate arbitrarily shaped clusters.

Sections 3.2.2.1 to 3.2.2.6 describe the procedures of the developed BOCEDS clustering algorithm.

3.2.2.1 Initialize the Micro-cluster

In this step, new micro-cluster (MC_{new}) is created in case the data point does not fall in any micro-cluster. The micro-cluster creation begins by initializing the micro-cluster feature vector. The data point is set as the center of the micro-cluster $(C = X_i)$, and the initial radius is set to the minimum radius $(R = R_{\min})$. The local density and number of data points in the shell region are set to 1 $(N = N' = 1)$ because the micro-cluster contains only one data point. The edge list is an empty set of intersecting micro-clusters $(EL = \phi)$. The initial energy is set to 1 $(E = 1)$ to ensure that a micro-cluster is just created with full energy level. The newly created micro-cluster (MC_{new}) has a local density $(N = 1)$ less than the density threshold $(N < Th_{density})$ and has a positive energy $(E = 1)$. Thus, on the basis of *Definition 3*, MC_{new} is a potential micro-cluster and is thus added to the potential micro-cluster set $(MC_{potential})$ using the union operation in Eq. 3.1.

$$MC_{potential} = MC_{potential} \cup MC_{new} \qquad 3.1$$

However, the setting of the density threshold to 1 $(Th_{density} = 1)$ implies that the new micro-cluster immediately turns into a core micro-cluster (MC_{core}) and practically no potential micro-clusters exists. Potential micro-clusters do not participate in the clustering graph; thus, they do not have macro-cluster ids $(M = 0)$.

3.2.2.2 Search the Target Micro-cluster

Each time a new data point arrives from the data stream, the developed BOCEDS attempts to map it to an existing micro-cluster based on the Euclidean distance (d) between the micro-cluster center and the data point. A data point (X_i) belongs to a micro-cluster, $Q(N, N', C, R, E, EL, M)$, if the distance value (d) is less than the radius of the micro-cluster and is expressed by Eq. 3.2.

$$d(X_i, C) < R \quad 3.2$$

The mapped micro-cluster may be one of the following micro-cluster sets:

a. A weak micro-cluster from the core micro-cluster set (MC_{weak}) in the buffer;

b. A potential micro-cluster from the potential micro-cluster set $(MC_{potential})$; or

c. A core micro-cluster from the core micro-cluster set (MC_{core}).

To find the target micro-cluster for the newly arrived data point (X_i), the developed BOCEDS uses a three-step search operation, as illustrated in Algorithm 1. The first search operation is executed on a weak micro-cluster set in the buffer using Eq. 3.2. This search operation is a type of pruning operation that aims to find the relevant micro-cluster from the temporary irrelevant micro-clusters.

Algorithm 1: ***Micro-cluster Searching***

Input: *Data point X_i, core micro-cluster set MC_{core}, potential micro-cluster set $MC_{potential}$, weak micro-cluster set MC_{weak}.*

Output: *Micro-cluster, T that contains X_i.*

Step 1: *Initialize a target micro-cluster, $T = null$*

Step 2: *Find a weak micro-cluster, Q' $(i.e. Q' \in MC_{weak})$ that satisfies Eq. 3.2.*

Step 3: *If $Q' \neq null$, then*

Set $T = Q'$ and go to Step 8.

[End If]

Step 4: *Find a potential micro-cluster, Q'' $(i.e. Q'' \in MC_{potential})$ that satisfies Eq. 3.2.*

Step 5: *If* $Q'' \neq null$ *, then*

Set $T = Q''$ *and go to Step 8.*

[End If]

Step 6: *Find a core micro-cluster,* Q''' $(i.e. Q''' \in MC_{core})$ *that satisfies Eq. 3.2.*

Step 7: *If* $Q''' \neq null$ *, then*

Set, $T = Q'''$

[End If]

Step 8: *Return T.*

If no such weak micro-cluster is found, then the algorithm searches the potential micro-cluster set to find the target micro-cluster in a similar manner as the first search operation. In case these two search operations fail, a final search is executed on the core micro-cluster set to find the mapped core micro-cluster. In case two or more micro-clusters satisfy Eq. 3.2, the algorithm randomly selects the target micro-cluster.

3.2.2.3 Update the Micro-clusters

If any micro-cluster receives a new data point, then the metadata will be updated recursively. If at the t^{th} time instant, micro-cluster $T(N_t, N'_t, C_t, R_t, E_t, EL_t, M_t)$ exists and a new data point (X_{t+1}) has been mapped to that micro-cluster, then its summary information or metadata at $(t+1)^{th}$ time is updated online, as discussed in Algorithm 2. The local density (N_{t+1}) is simply incremented by Eq. 3.3. In the case where T is a weak micro-cluster $(T \in MC_{weak})$ or a potential micro-cluster $(T \in MC_{potential})$ with a local density greater than the density threshold $(N_t = Th_{density})$, T is added to the core micro-cluster set (MC_{core}). If T is already a core micro-cluster $(T \in MC_{core})$ or a newly added core micro-cluster, then its radius (R_{t+1}) is recursively updated by utilizing the forgetting mechanism (Khamassi et al., 2018; W. Wang & Vrbanek, 2008). The micro-cluster radius is updated if the data point stays in the shell region of the micro-cluster because the data points in kernel region have minimal impact on increasing the radius and the current radius is sufficiently large. The radius updating equation is formulated based on the statement that the farther away the data point expands, the more intensive the micro-cluster radius is than in the case of a closer data point. The closeness of the

data point to the outer edge of the micro-cluster is defined as $[\{2d(X_{t+1}, C_t)/R\} - 1]$, and the radius is increased by a forgetting factor of $(1/Decay)$ per unit closeness. Thus, the micro-cluster radius is updated using Eq. 3.4.

$$\text{Local density, } N_{t+1} = N_t + 1 \tag{3.3}$$

$$\text{Radius, } R_{t+1} = \min\left(\left[R_t + \left\{\frac{2 \times d(X_{t+1}, C_t)}{R_t} - 1\right\} \times \frac{1}{Decay}\right], R_{\max}\right) \tag{3.4}$$

The micro-cluster radius never exceeds the maximum radius $(R_{\max})$. The micro-cluster center (C_{t+1}) is updated only if the data point lies in the shell regions (d stays in the range of $[0.5 \times R_{t+1}, R_{t+1}]$). The motivation of this operation is that the participating points for center updating remain in the shell region because they prevent the micro-cluster from following the drifting of the data stream endlessly by limiting its movement (Hyde et al., 2017). If data point X_{t+1} resides in the shell region, then the number of data points in the shell region (N'_{t+1}) and the micro-cluster center (C_{t+1}) are updated using Eq. 3.5 and Eq. 3.6, respectively.

$$\text{Number of data points in the shell region, } N'_{t+1} = N'_t + 1 \tag{3.5}$$

$$\text{Center, } C^k_{t+1} = \frac{(N'_{t+1} - 1) \times C^k_t + X^k_{t+1}}{N'_{t+1}} \tag{3.6}$$

for k=1,2,3,…,D, where D is the dimension size of data point

To update the energy (E_{t+1}) of the micro-cluster, a new energy updating function is designed based on the Newton's law of gravitation(Newton, 1729), where the amount of energy gained by the micro-cluster is inversely proportional to the distance between the cluster center and the data point. Thus, the energy (E_{t+1}) of the newly mapped core micro-cluster is updated using Eq. 3.7.

$$\text{Energy, } E_{t+1} = E_t + \left\{\frac{R_t - d(X_{t+1}, C_t)}{R_t}\right\} \times \frac{1}{Decay} \tag{3.7}$$

Algorithm 2: ***Micro-cluster Update***

Input: *Data point* X_i, *micro-cluster* $T(N_t, N'_t, C_t, R_t, E_t, EL_t, M_t)$, *and distance* $d(X_i, C_t)$

Step 1: *Update the local density* (N_t) *of T using Eq. 3.1.*

Step 2: *If* $\left[\{(T \in MC_{potential}) \wedge (N_{t+1} = Th_{density})\} \vee (T \in MC_{weak})\right]$, *then*

Add the micro-cluster (T) *to core micro-cluster set,* $MC_{core} = MC_{core} \cup T$

Update the radius (R_{t+1}) *of micro-cluster* (T) *using Eq. 3.4.*

Set $E_{t+1} = 1$

Else If $T \in MC_{core}$, *then*

Update the radius (R_{t+1}) *of micro-cluster* (T) *using Eq. 3.4*

Update the energy (E_{t+1}) *of micro-cluster* (T) *using Eq. 3.7.*

[End If]

Step 4: *If* $\frac{R_{t+1}}{2} < d \le R_{t+1}$, *then*

Update the number of data points in shell region (N'_{t+1}) *of micro-cluster* (T) *using Eq. 3.5.*

Update the center (C_{t+1}) *of micro-cluster* (T) *using Eq. 3.6.*

[End If]

Step 4: *Exit*

From Algorithm 2, if the newly mapped micro-cluster is a weak one, then its energy (E_{t+1}) is simply reset to 1. In case of potential micro-cluster, the energy (E_{t+1}) is set to 1 if the density (N_{t+1}) is equal or above the density threshold $(Th_{density})$.

Furthermore, in Algorithm 2, if the mapped micro-cluster (T) is a potential micro-cluster and its local density (N_{t+1}) only meets the density threshold $(Th_{density})$, then the micro-cluster (T) is converted to core micro-cluster and added to the core micro-cluster set (MC_{core}). Meanwhile, a weak micro-cluster (a special type of core micro-cluster in buffer with no energy) has lost its energy to be live due to the evolving characteristic of data stream. The weak micro-cluster already meets its density

threshold. Thus, if the mapped micro-cluster is a weak one, then it is relevant to the current data stream contents and is immediately converted to core micro-cluster and added to the core micro-cluster set (MC_{core}).

3.2.2.4 Move the Weak Micro-cluster to Buffer

After clustering each data point, the energy of each core micro-cluster is decreased to reflect the evolving nature of the data stream. Any core micro-cluster with the energy below zero is marked as a weak micro-cluster. In this step, the irrelevant micro-cluster is stored in a special storage called buffer. The purpose of this operation is to give chance to an irrelevant micro-cluster to be alive again in case data comes to this micro-cluster in near future. The case has been illustrated in Section 1.3, Chapter 1. This operation prevents the frequent creation and deletion of micro-cluster to reduce the processing time. Algorithm 3 describes the weak micro-cluster identification process and the moving of weak micro-clusters in a special buffer.

Algorithm 3: ***Moving Weak Micro-clusters to Buffer***

Input: *Core micro-cluster set (*MC_{core}*), weak micro-cluster set (*MC_{weak}*) in buffer,* $Decay$

Step 1: *Reduce the energy of* $\frac{1}{Decay}$ *from all core micro-clusters in* MC_{core}.

Step 2: *For each micro-cluster* $T(N, N', C, R, E, EL, M) \in MC_{core}$, *do*

If $E \leq 0$, *then*

Remove all edges from T, $EL = \phi$

Remove the edge $Edge(T', T) \in EL(T')$ *from any core micro-cluster*

$T' \in MC_{core}$, $Edge(T', T) = \phi$

Reset the number of macro-clusters of T, $M = 0$

Remove T *from the core micro-cluster set,* $MC_{core} = MC_{core} - T$

Set the new dying energy $E = 0.5$

Add the micro-cluster T *to the weak micro-cluster set in buffer,* $MC_{weak} = MC_{weak} \cup T$.

[End If]

[End For]

Step 4: Exit.

$E \leq 0$ of a core micro-cluster (T) indicates that the micro-cluster has become weak (temporarily irrelevant).The identified weak micro-clusters are moved to the buffer memory with setting a new dying energy half of the initial energy (i.e. $E = 0.5$). The purpose of storing the weak micro-cluster in the buffer is to serve as reference to weak micro-clusters in the future. They do not participate in the cluster graph and are disconnected from the graph by the removal of the intersecting edges

3.2.2.5 Kill the Weak Micro-cluster in Buffer

Along with the reduction of the energy of the core micro-cluster (Section 3.2.4), the energy of each weak micro-cluster in the buffer is also reduced by $\frac{1}{Decay}$. The purpose of this operation is to identify the dying micro-clusters that are unrelated to the recent data stream contents for a long time. A weak micro-cluster with a zero or negative energy $(E \leq 0)$ is identified as a dying micro-cluster and is killed permanently from the memory. A dying micro-cluster is a totally irrelevant micro-cluster with respect to the current data stream content. The size of buffer is small and to accommodate the temporary irrelevant micro-cluster, the totally irrelevant or dying micro-cluster needs to be removed. The purpose of this operation is to delete the dying micro-cluster from memory to prevent the buffer from growing beyond its limit of size. The procedure for identifying and killing dying micro-clusters is presented in Algorithm 4.

Algorithm 4: ***Micro-cluster Removal***

Input: *Weak micro-cluster set (MC_{weak}), potential micro-cluster set ($MC_{potential}$), Decay*

Step 1: *Reduce an amount of* $\frac{1}{Decay}$ *energy from all weak micro-clusters in* MC_{weak}.

Step 2: *For each weak micro-cluster* $(W(N, N', C, R, E, EL, M) \in MC_{weak})$, *do*

If $E \leq 0$, *then*

Remove T *from the weak micro-cluster set in the buffer,* $MC_{weak} = MC_{weak} - W$.

[End If]

[End For]

Step 3: *Reduce an amount of* $\frac{1}{Decay}$ *energy from all potential micro-clusters in* $MC_{potential}$.

Step 4: *For each potential micro-cluster (* $P(N, N', C, R, E, EL, M) \in MC_{potential}$*), do*

If $E \leq 0$ *, then*

Remove P *from the potential micro-cluster set,* $MC_{potential} = MC_{potential} - P$.

[End If]

[End For]

Step 5: *Exit*.

Furthermore, from Algorithm 4, the weak micro-clusters with a non-positive energy $(E \leq 0)$ are identified as candidates for dying micro-clusters and are completely removed from the buffer. The energy of each potential micro-cluster is also decreased, and the micro-cluster with a zero or negative energy $(E \leq 0)$ is marked as an outlier. Similar to a weak micro-cluster, outliers are also removed from the memory permanently.

3.2.2.6 Update the Cluster Graph

Similar to other density-based clustering techniques, such as CODAS, CEDAS, BOCEDS maintains a clustering graph to generate a macro-cluster online. The clustering graph must be updated in four cases.

Case 1. A potential micro-cluster satisfies the minimum density threshold to be the core micro-cluster.

Case 2. A weak micro-cluster becomes a core micro-cluster because it contains the current data point.

Case 3. The center of a core micro-cluster is shifted.

Case 4. A core micro-cluster is moved to the buffer and changes to a weak micro-cluster.

In the above cases, the edge list of the micro-clusters may be changed; thus, the number of macro-clusters must be updated accordingly.

Algorithm 5: ***Update Cluster Graph(G)***

Input: *A core micro-cluster* $T(N, N', C, R, E, EL, M)$ *that has been generated or modified, core micro-cluster set (*MC_{core}*), clustering graph G*

Step 1: *For each core micro-cluster (*$T'(N, N', C, R, E, EL, M) \in MC_{core}$*), do*

Set d *=Euclidean distance between centers of micro-clusters T and* T'

Set d' *=intersecting distance between T and* T' *from Eq. 3.8*

If $d <= d'$*, then*

Add the edge $Edge(T', T)$ *to the edge list of* T'*,* $T'.EL = T'.EL \cup Edge(T', T)$

Add the edge $Edge(T', T)$ *to the edge list of* T*,* $T.EL = T.EL \cup Edge(T', T)$.

[End If]

[End For]

Step 2: *If any micro-cluster edge list has changed, then*

Set a new number of macro-clusters throughout the graph.

[End If]

Step 3: *Exit*.

The first two cases introduce a new vertex in the clustering graph. For Cases 1–3, the intersecting micro-clusters are calculated, and the edge lists are updated. If two c-micro-clusters T and T' with radii R and R', respectively, exist, then the intersecting distance (d') is the distance between the centers of the two c-micro-clusters and is calculated using Eq. 3.8.

$$d' = \begin{cases} R + \frac{R'}{2}, if \;\; R \geq R' \\ R' + \frac{R}{2}, if \;\; R' > R \end{cases} \qquad 3.8$$

If the edge list is changed, then the number of macro-clusters is updated on the basis of the edge list. In Cases 4 and 5, one vertex is removed from the graph. Any edge that connects the removed vertex and a vertex in the graph is removed from the graph. The number of macro-clusters is reassigned in the graph.

3.3 Summary

This chapter explains the developed online clustering algorithm called BOCEDS in details to overcome the identified research problems. BOCEDS is a single stage algorithm that generates the clusters from the data stream in a single online stage. The algorithm uses the concept of micro-cluster to summarize the data from data stream. BOCEDS defines four types of micro-cluster and they are core, potential, weak and outlier micro-clusters. Outlier micro-clusters are the background noise and core micro-clusters are the final micro-clusters those participate in cluster generation. On the other hand, potential micro-clusters are converted to either core or outlier micro-cluster. The rest weak micro-cluster is temporary irrelevant micro-clusters those are identified as totally irrelevant of core micro-cluster in later period of time. Euclidean distance measure has been used to compute the distance between micro-clusters and also the distance between data and micro-clusters.

Before execution of the algorithm, the application parameters (minimum and maximum radius, density threshold, decay) are set by the application expert. The algorithm executes in five distinct steps. In the first step, a micro-cluster is created in case the data does not fall into any micro-clusters in the current model. The micro-cluster searching operation is executed in the second step. The searching domains are weak, potential and core micro-cluster set. A data may falls into a region of a micro-cluster. In such a case, the information of the micro-cluster is updated in a fully online manner in the third step. The forgetting mechanism is utilized to design the radius updating procedure and Newton's gravity law is adapted to design the micro-cluster energy updating procedure. The energy of every micro-clusters including core, potential and weak micro-cluster are updated in every time, a data is clustered. In the fourth step, based on the current energy, some core micro-clusters become weak (temporary irrelevant) micro-clusters and stored in a special stage called buffer. On the other hand, some weak micro-clusters become dying (totally irrelevant) micro-cluster and are removed from buffer completely. In the final step, clustering graphs are generated based

on the connectivity among core micro-clusters. Each disjoint graph represents a single macro-cluster or simply cluster.

To summarize, BOCEDS uses the concept of maintaining the local optimal radius rather than global radius concept of other algorithms to effectively reduce the sparse regions in a micro-cluster by maintain the local optimal radius. This fact also helps to handle noise effectively. The algorithm has the ability to handle evolving data stream by considering an energy that which decreases over time. Another prominent feature of BOCEDS can keep track of the temporary irrelevant micro-clusters by storing them in buffer those are ignored by other density-based algorithms. The pruning of these micro-clusters prevents the frequent creation and removal of micro-clusters. The operation improves the processing time significantly. The whole steps works recursively those in turns provide a fully online behaviour of BOCEDS.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Introduction

In the previous chapter (Chapter 3), the developed BOCEDS algorithm has been discussed in details. The input parameters have also been defined. The developed clustering algorithm called BOCEDS is implemented using MATLAB R2014a and run on a Core i7 processor with 2GB primary memory environment. The algorithm is executed in Microsoft Windows 10 operating system environment. This chapter describes the correct functionality of BOCEDS algorithm in Section 4.2.1 to form clusters in dense regions, to detect noises in data stream, to detect the drift in data stream. The scalability and processing speed properties of the algorithm is discussed in Section 4.2.2. The cluster quality and memory requirement is measured and explained in Section 4.2.3 to evaluate the performance of BOCEDS. Some parameters are used in the algorithm, which are set based on expert knowledge on domain. The sensitivity of those parameters are measured and described in Section 4.2.4. To evaluate the applicability of developed algorithm to real-world data stream, a case study that describes the clustering of real-world weather data stream is discussed in Section 4.3.

4.2 Performance Metrics

Evaluating the performance of clustering algorithm is one of the important issues to validate the goodness of the clustering result (Maulik & Bandyopadhyay, 2002). The performance of clustering algorithm is defined in terms of five metrics by most of the clustering algorithms for data stream (Amini et al., 2014; Hyde & Angelov, 2015; Reddy & Bindu, 2017). The metrics are described in the following Section 4.2.1 to Section 4.2.5.

4.2.1 Cluster Formation and Noise Sensitivity

To validate the correct functionality of a clustering algorithm, it is necessary to test the formation of micro-clusters as well as clusters as new data arrives from a data stream. Cluster formation metric describes the fact that how the algorithm forms adds, merges and separates macro-clusters in a continuously evolving environment (Hyde et al., 2017). It is also important to visualize the creation and removal of micro-clusters over time. The cluster formation metric is evaluated on both of clean and noisy data stream to ensure the proper functionality of the algorithm. The generated micro-clusters and clusters visualized with different colour to differentiate them.

The visualization of cluster formation on noisy data stream alone cannot describe the functionality of the algorithm. It is necessary to define a numerical metric. The solution is to measure the noise sensitivity of the clustering algorithm. Noise sensitivity describes the behavior of a clustering algorithm in a noisy data stream environment. This characteristic is calculated by comparing the percentage of data point assignment (Hyde & Angelov, 2015) before and after adding the noise to the data stream. The percentage of data point assigned to a cluster has been redefined as the data coverage. If N_t data points from data stream generate C core micro-clusters at time *t*, then data coverage is the ratio of the cumulative local densities to the total data points that appeared (N_t), as shown as follows:

$$\text{Data coverage, } DC_t = \frac{\sum_{i=1}^{C} n_i}{N_t} \times 100\% \qquad 4.1$$

Where, n_i is the local density of the micro-cluster *I* at time *t*.

If DC_{clean} is the data coverage over clean data stream (before adding noisy data) and DC_{noisy} is the data coverage over noisy data stream (after adding noisy data), then the noise sensitivity or identified noise is measured by

$$\text{Noise sensitivity, } Noise[\%] = \left| DC_{clean} - DC_{noisy} \right| \qquad 4.2$$

The noise sensitivity defines the percentage of noise in the data stream. An ideal clustering algorithm detects all the noises in the data stream.

4.2.2 Processing Speed and Dimensionality

Processing time is a very important metric to evaluate any algorithm. The data point processing speed is measured as the average time needed to complete the clustering of a data point from the data stream (Aggarwal et al., 2003). The processing time usually measured in a window basis way over the whole data stream and expressed in seconds, milliseconds or micro-seconds. On the other hand, the dimensionality metric describes the scalability behaviour of the clustering algorithm for low to high dimensional data stream. Usually, dimensionality property is defined in terms of scalability. Scalability is measured as the change in processing time from low- to high-dimensional data stream.

4.2.3 Cluster Quality

Cluster quality describes the quality of generated clusters by the clustering algorithm. The metric is defined in terms of two parameters, namely, cluster accuracy, and purity (Amini et al., 2014; Hyde & Angelov, 2015). These two parameters are measured with respect to the true cluster (class) labels that are known for the data stream.

Purity is defined as the number of data points that belong to a dominant cluster. The higher percentage of the dominant class labels in each cluster, the higher the cluster purity. If n_i samples exist in a cluster and among them, n_i^d samples lie in the dominant cluster, then for N clusters, the mean purity is measured as

$$Purity = \frac{\sum_{i=1}^{N} n_i^d}{n_i} \times 100\% \qquad 4.3$$

The presence of a high number of clusters with few data points may provide high purity despite most of the data points clustered incorrectly. Cluster accuracy solves this limitation; it is defined as the amount of data points in a cluster that truly belong to the cluster(Hyde & Angelov, 2015). If n_i samples exist in a cluster and among them, n_i^d samples lie in the dominant cluster, then for N clusters, the accuracy is measured as

$$Accuracy = \frac{\sum_{i=1}^{N} n_i^d}{\sum_{i=1}^{N} n_i} \times 100\% \quad 4.4$$

An ideal clustering algorithm shows a balanced accuracy and purity for any data stream. It is also expected the accuracy and purity equal or close to 100% in all time periods.

4.2.4 Memory Efficiency

Due to the volume property of data stream, data comes continuously from the data stream and the amount of data grows exponentially. However, low memory requirement is desired for any algorithm to reduce the maintenance cost. So, evaluating the memory efficiency is an important metric to define the performance of the clustering algorithm. Memory efficiency is measured as the storage required when clustering the data stream. However, the storage requirement is proportional to the micro-clusters in the model for online clustering algorithm as the data points are removed immediately after clustering (Baruah & Angelov, 2014; Hyde et al., 2017). Low memory requirement is desired for an online clustering algorithm for data stream.

4.2.5 Parameter Sensitivity

The operation of almost all the algorithms depends strongly on the initialization of its parameters. The sensitivity analysis evaluates the algorithms based on the analysis of these parameters (Baruah & Angelov, 2014; Riyadh et al., 2017). It shows how the algorithm's parameters affect the clustering quality and the best setting for the algorithm's parameters (Dong et al., 2018; Guha et al., 2001; Shao et al., 2018). The parameter sensitivity is defined in terms of three parameters and they are density threshold, decay and micro-cluster radius. The first step in this analysis of performance is the investigation of the sensitivity of the algorithms for varying parameters. The investigation of the sensitivity of density threshold and micro-cluster radius is evaluated by measuring the cluster quality based on Eq. 4.3 and Eq. 4.4. On the other hand, the sensitivity of decay is measured by computing the computational time for different decay setting.

4.3 Result Analysis

A series of experiments has been executed on two syntactic and one practical data stream to measure the performance of the developed BOCEDS algorithm and compare it with existing density-based clustering algorithms. The performance of the developed BOCEDS algorithm measured over three benchmark data stream in the field of online clustering and they are Mackey–Glass, helical, and KDDCUP'99 data stream.

Before execution of the BOCEDS algorithm on a data stream, four application dependent parameters are set by the user and they are minimum radius (R_{min}), maximum radius (R_{max}), decay (*Decay*) and density threshold($Th_{density}$). The optimal values of two parameters (*Decay* and $Th_{density}$) are directly derived from literature as the data streams in this experiment are well studied by several algorithms. As the concept of other two parameters is new, the optimal values of them (R_{min} and R_{max}) are set by the experiments. The popular trial and error method is used to find the optimal values of these two parameters. In this method, initially the values of R_{min} and R_{max} are set to be identical as the optimal value of radius exists in literature. The value of R_{min} is decreased by a small amount of 0.01 and the value of R_{max} is increased by a small amount of 0.01 till the cluster quality remains same or increases. In the case the cluster quality is found highest, the values of R_{min} and R_{max} are set to be used for the rest of experiments.

The performance of BOCEDS algorithm is defined in terms of cluster formation (Section 4.3.1.1), noise sensitivity (Section 4.3.1.2), data point processing speed (Section 4.3.2.1), scalability (Section 4.3.2.2), response to variable decay (Section 4.3.2.3), cluster purity (Section 4.3.3.1), cluster accuracy (Section 4.3.3.2), memory efficiency (Section 4.3.3.3), and parameter sensitivity (Section 4.3.4).

4.3.1 Cluster Formation and Noise Sensitivity

Mackey–Glass time series is a benchmark dataset that has been used to study the behaviour of dynamic changes in many evolving clustering algorithms (Blazic & Skrjanc, 2019; Kakkar et al., 2017; Makul & Ekinci, 2017). Although Mackey–Glass time series is a stationary dataset, the developed algorithm clusters data points online, wherein the data points are removed immediately after clustering. This dataset is used to show the manner in which micro-clusters are formed in the clean data stream with

local optimal radius. Adding some noisy data to this data stream, make the Mackey-Glass data stream as a noisy data stream environment. This noisy data stream is used to define the behaviour of BOCEDS algorithm in a noisy environment.

The dataset is a syntactic three-dimensional data stream that is composed of two Mackey–Glass time series (Glass & Mackey, 2010) generated by the following nonlinear time delay differential equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10} - bx(t)} \quad 4.5$$

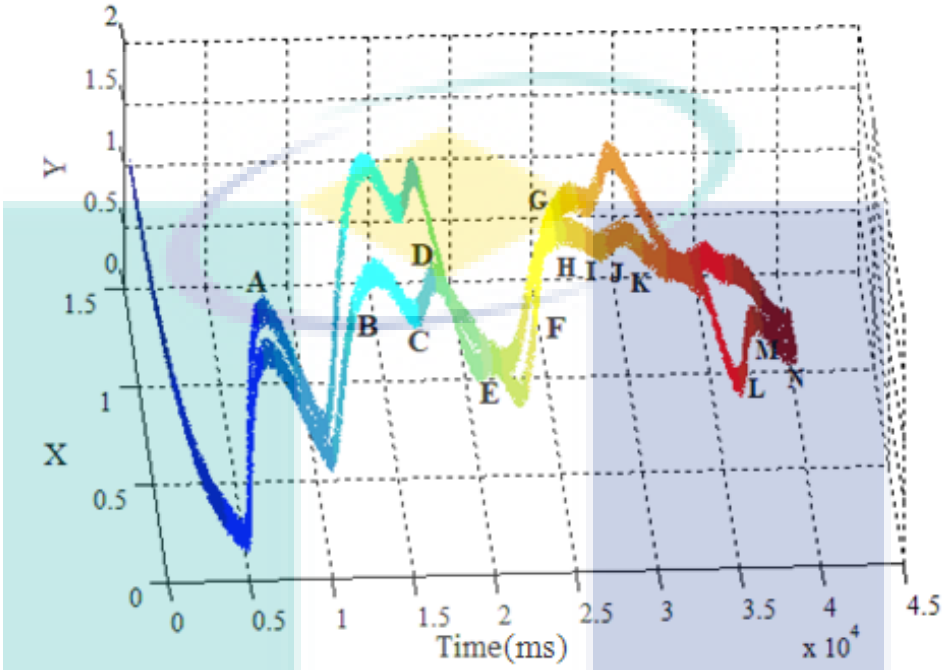
This equation is solved using the fourth-order Runge–Kutta numerical method with different values for *a* and *b*, and the data stream is generated. The 10^{th} data of data stream has been replaced with a noisy data to generate a noisy Mackey–Glass data stream, which contains 10% noisy data. The noisy data are those data those are not in the normal range of data. In case of Mackey-glass time series data, the data are normalized to the range from 0.0 to 1.0. The noisy data are chosen from the data beyond this range. The developed BOCEDS algorithm is applied on a clean Mackey–Glass data stream to understand the cluster formation in dense areas separated by sparse areas. BOCEDS is executed on the noisy Mackey–Glass data stream to validate the correct functionality in a noisy environment. The clustering parameters for clean and noisy Mackey–Glass data streams are set as $Decay = 1000$ data points, $Th_{density} = 15$ data points, $R_{\min} = 0.03$, and $R_{\max} = 0.07$.

4.3.1.1 Cluster Formation

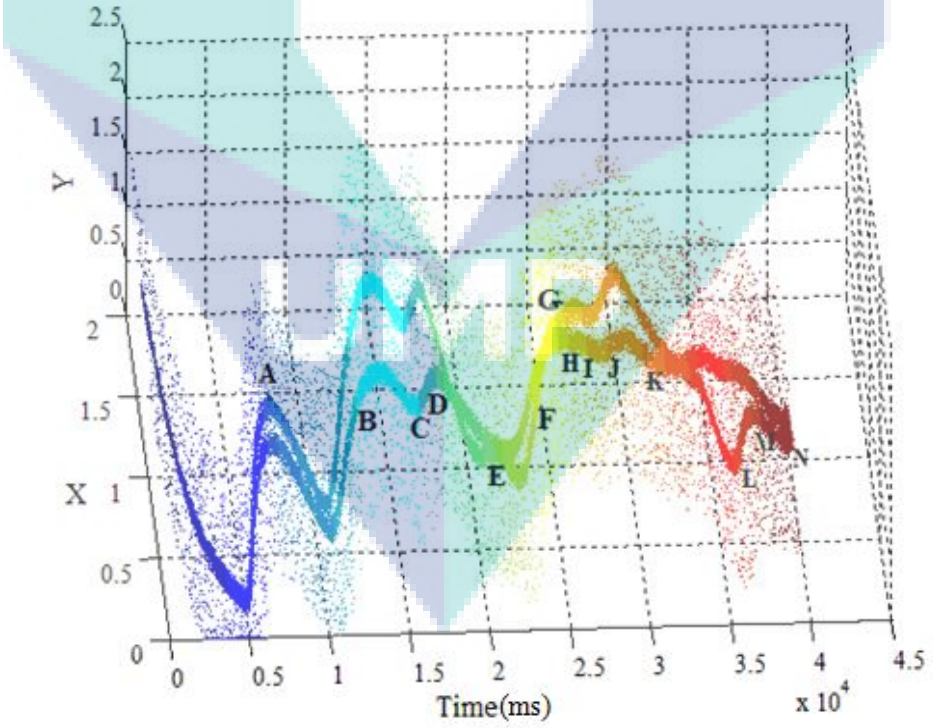
BOCEDS algorithm generates the macro-clusters or simple clusters in the highly dense areas separated by low dense areas of data stream. When this algorithm is executed on the clean Mackey–Glass data stream, the clustering results after four time periods are shown in Figure 4.1(a)–3(d). For one-fourth time period (first 10000 data points), the clustering result in Figure 4.1(a) confirms that micro-clusters and clusters are formed as data points that arrive in new dense areas. The generated micro-clusters have a nearly equal radius.

UMP

arrival. The data are shown in Figure 4.2(a) and Figure 4.2(b) for clean and noisy Mackey–Glass data streams, respectively, where the change in the number of macro-clusters is reflected by the change in data path color.



(a) Without noise



(b) With noise

Figure 4.2 Formation of macro-clusters in Mackey–Glass data stream

A comparison of Figure 4.2(a) and Figure 4.2(b) indicates that the formation of macro-clusters in noisy data stream is slightly delayed compared with that in clean data stream. The potential micro-clusters take more time to meet the density threshold than clean data stream due to the presence of noisy data points in noisy data stream. This fact is the cause of initial delay for forming the macro-clusters in case of noisy Mackey–Glass data stream. In Figure 4.2(a) and Figure 4.2(b), a single macro-cluster contains all the data points, except the noisy data points until point A. The trend of varying the number of macro-clusters is relatively similar in clean and noisy data streams. At Point A, the data stream is separated into two distinct macro-clusters, and the change is denoted by the color blue. The short duration of macro-cluster existence is illustrated in Points C–D, G–H, H–I, I–J, and J–K. The data stream ends at Point N.

From Figure 4.2(a) it can be seen that BOCEDS maintains the local radius during clustering the data stream that is updated towards its optimal value. Moreover, some micro-clusters are created and deleted over time that confirms the correct functionality of BOCEDS to handle evolving data stream. From Figure 4.2(b), it can be concluded that BOCEDS is able to generate clusters in both of clean and noisy environments.

4.3.1.2 Noise Sensitivity

The measured noise (using Eq. 4.2) of existing CEDAS and developed BOCEDS techniques over clean and noisy (10% noise) Mackey–Glass data streams is shown in Figure 4.3. From the figure, the initial identified noise by existing CEDAS algorithm is more than the original noise (10%) by a considerable amount. By contrast, the identified noise is more in the developed BOCEDS algorithm as the micro-cluster radius is less than its optimal value in these time periods and more some micro-clusters do not get enough data to be core micro-cluster. From Figure 4.3, it is also seen that the percentage of detected noise becomes up and down periodically due to the addition of noisy data in the noisy data stream. As time progresses, the identified noise by BOCEDS algorithm oscillates the original amount of noise (10%). In these time periods, the micro-clusters recursively update their radius toward their optimal radius, and the deviation in noise percentage from the original noise amount (10%) decreases.

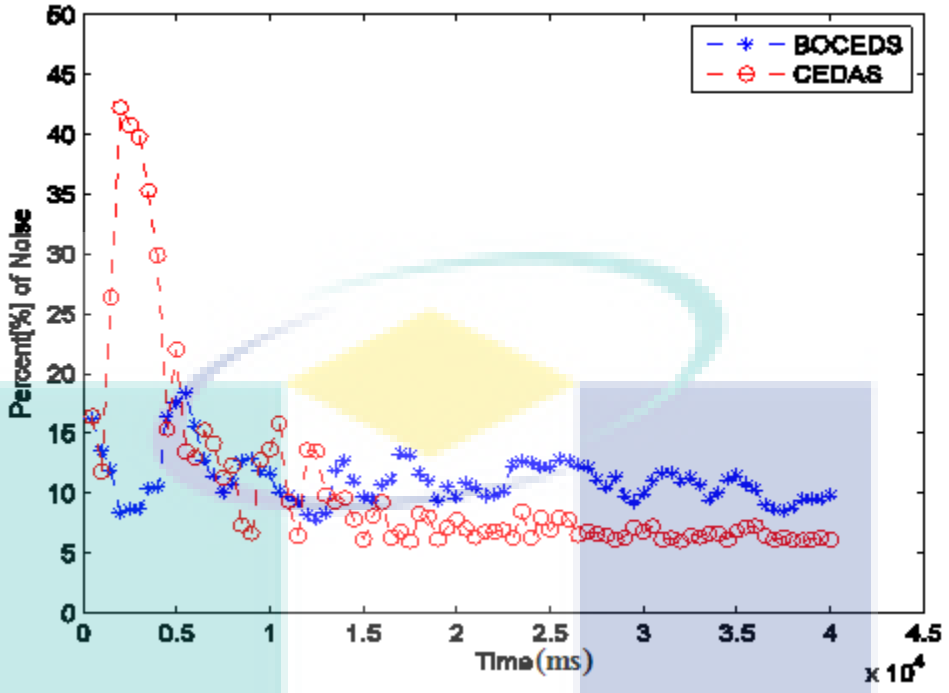


Figure 4.3 Noise sensitivity over Mackey–Glass data stream

Finally, the identified noise remains close to the original percentage of noise in the developed BOCEDS algorithm. Conversely, the amount of identified noise by CEDAS remains approximately at 7%. Thus, BOCEDS identifies nearly all the noisy data points, whereas CEDAS fails to identify some noise in the Mackey–Glass data stream. The developed BOCEDS also shows a superior performance in identification of noisy data points in comparison with the existing CEDAS.

4.3.2 Speed and Dimensionality

The helical data stream contains a set of helixes (Steinhaus, 1999). The original helical data stream consists of three helical data series of a circular helix, as shown as follows:

$$X = r\sin(t); Y = r\cos(t); Z = Z = ct \qquad 4.6$$

where r is the radius of the helix, and c is the pitch parameter (pitch=2πc).

In the equation, as the value of t increases, points X, Y, and Z produce a right-handed helix around the z-axis in a right-handed coordinate system. The helical data stream is generated using the above time series equation for different values of t and a

constant value of c. The data series is then moved into a higher-dimensional data space by adding additional data coordinates. The developed BOCEDS is executed on a high-dimensional data stream to measure the data point processing speed and response to high dimensionality. The clustering parameters are set as $Decay = 1000$ data points, $Th_{density} = 4$ data points, $R_{\min} = 0.04$, and $R_{\max} = 0.06$.

4.3.2.1 Processing Speed

It is desired for any clustering algorithm to show high processing speed to enable real time processing of data stream. Figure 4.4 shows the change in processing delay with time for clustering of a three-dimensional helical data stream using the developed BOCEDS algorithm and compares it with other two density-based online clustering techniques, namely, CEDAS and CODAS.

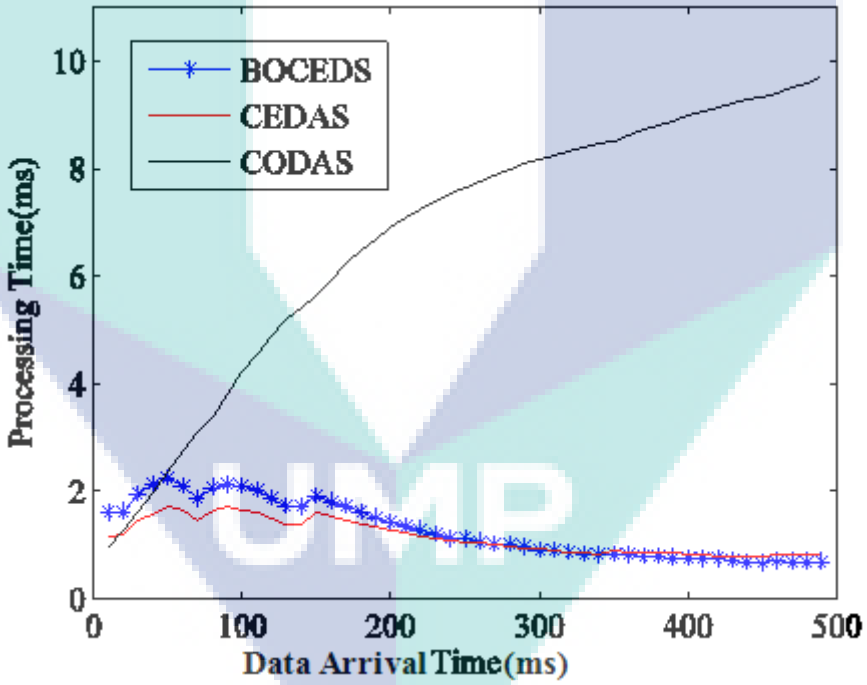


Figure 4.4 Processing time on a helical data stream

As shown in Figure 4.4, the processing time of CODAS is considerably higher than those of CEDAS and BOCEDS. This result is due to the fact that the micro-clusters in CODAS do not evolve, whereas those in CEDAS and BOCEDS are evolving in nature. In the developed BOCEDS algorithm, the initial micro-cluster radius is below its optimal, and the number of initial micro-clusters is more in BOCEDS than that in

CEDAS. Accordingly, the initial processing time is more in BOCEDS than in CEDAS. However, the radius is updated toward its optimal because increased data points reach a micro-cluster. Although a time penalty is incurred for pruning operation, BOCEDS recovers some micro-clusters rather than creating new micro-clusters by a time-consuming process, thereby reducing the processing time. Similarly, rather than updating the edges of micro-clusters in the clustering graph at every data point, the developed algorithm only rechecks the edges if the center of the core micro-cluster is shifted or a new core micro-cluster is added to the graph. Figure 4.4 depicts that as time progresses; the processing time becomes close by half of the data point arrival time. The processing time of BOCEDS finally goes below the processing time of CEDAS, and the trend continues because most of the micro-clusters either reach their optimal or maximum radius.

4.3.2.2 Scalability

A good clustering algorithm requires low processing time and low delay penalty for extending it to a higher-dimensional data space. Figure 4.5 compares the sample or data point processing speed of the developed BOCEDS clustering algorithm with those of CEDAS, CODAS, CluStream, and DenStream on a helical data stream.

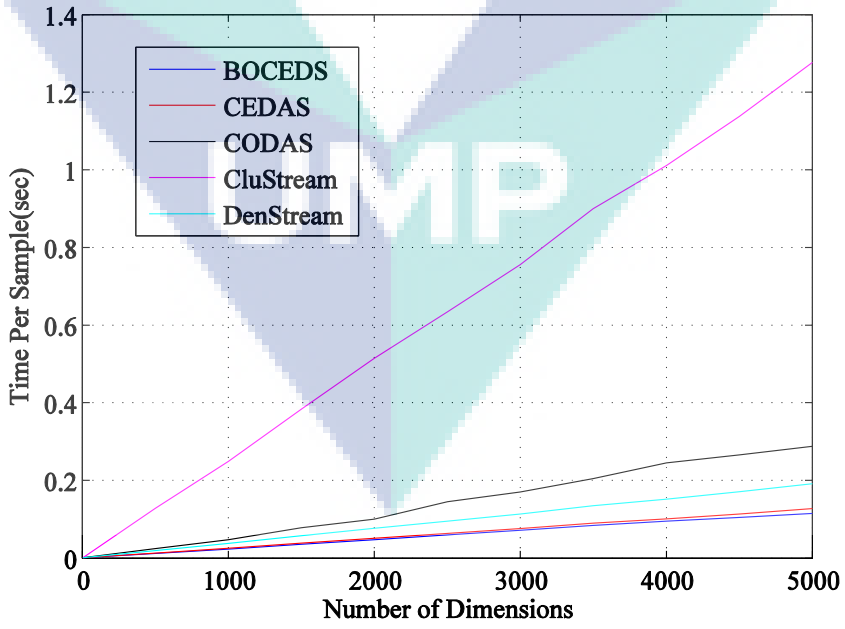


Figure 4.5 Processing speed on a helical data stream

The maximum limit of allowable number of micro-clusters in CluStream is 100 micro-clusters. All techniques form arbitrarily shaped macro-clusters similar to BOCEDS. Figure 4.5 illustrates that the data point processing time linearly increases in all mentioned techniques. The processing time of BOCEDS is considerably less than those of CluStream, CODAS, and DenStream. The processing speed of BOCEDS is less than that of CEDAS by a small amount. Although a time penalty is incurred for the pruning operation, the processing time is greatly reduced by adopting the spatiotemporal similarity concept in the developed BOCEDS. Thus, the time penalty factor of BOCEDS for increasing the dimension size is less than those of the online or hybrid clustering algorithms.

4.3.2.3 Response to Variable Decay

Decay defines the amount of data arrives per unit time from a data stream. A clustering algorithm shows the linear relationship between the decay and data processing time. The mean processing time is recorded for different decay periods for low- to high-dimensional helical data stream to measure the behaviour of the developed BOCEDS in variable decay. The plotted result of these records is shown in Figure 4.6.

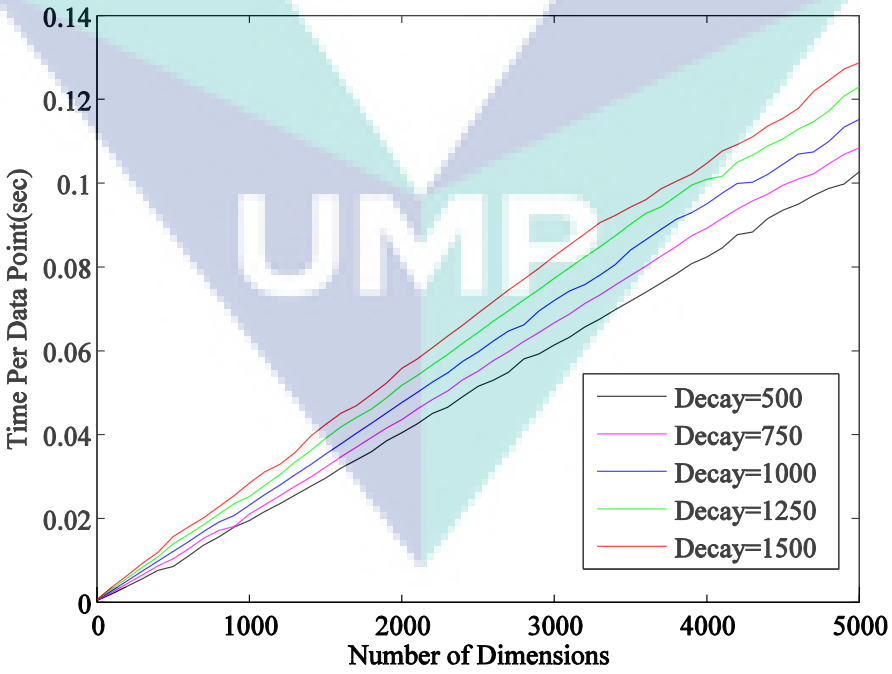


Figure 4.6 Processing time for various decay settings in the developed BOCEDS

The decay is increased from an initial value 500 to 1500 with an interval of 250. Eq. 3.7 states that the energy-reducing factor for each micro-cluster reduces with increasing decay for an evolving data stream. Thus, the decay time is also proportional to the number of micro-clusters due to the continuous drift in helical data stream. This relationship between the decay and the number of micro-clusters implies that the mean data point processing time increases with the decay. Figure 4.6 clearly reflects the relationship, in which the mean data point processing time increases with the decay. Moreover, the processing time also increases with increasing the number of dimensions. The reasons for this behaviour of BOCEDS algorithm has been discussed in Section 4.3.2.2. This characteristic shows that BOCEDS algorithm is efficient in handling the velocity property of data stream.

4.3.3 Cluster Quality

A well-known practical data stream called KDDCUP'99 (Bay et al., 2000) has been used for measuring cluster quality and memory. The data stream contains approximately 4900000 network traffics. The data stream is reduced to 10% to simulate the network intrusion attacks using the developed algorithm. Each data point of the stream is characterized with 41 features and an additional attribute for defining the attack. The database contains 21 types of network attack along with normal network traffic. The developed BOCEDS algorithm is applied on KDDCUP'99 data stream to identify the network traffic clusters. The clustering parameters are set as $Decay = 1000$ data points, $Th_{density} = 3$ data points, $R_{\min} = 0.06$, and $R_{\max} = 0.12$. The cluster analysis is performed for 500time intervals spaced at 10K data points. The data points are immediately removed after clustering. The performance parameters (i.e., accuracy, purity, and memory) are measured in every window.

4.3.3.1 Cluster Purity

The purity of clustering KDDCUP'99 using the developed BOCEDS is measured using Eq. 4.3 at several time periods. The purity of another two recent highly pure density-based clustering, namely, CODAS and CEDAS, is also measured on the same data stream. Moreover, the mean cluster purity of DStream and MRStream is taken from the results presented by Wan et al. (Wan et al., 2009). The cluster purity is

calculated for 500 time intervals spaced at 10K data points. Figure 4.7 compares the purity of BOCEDS with those of DStream, MRStream, CODAS, and CEDAS.

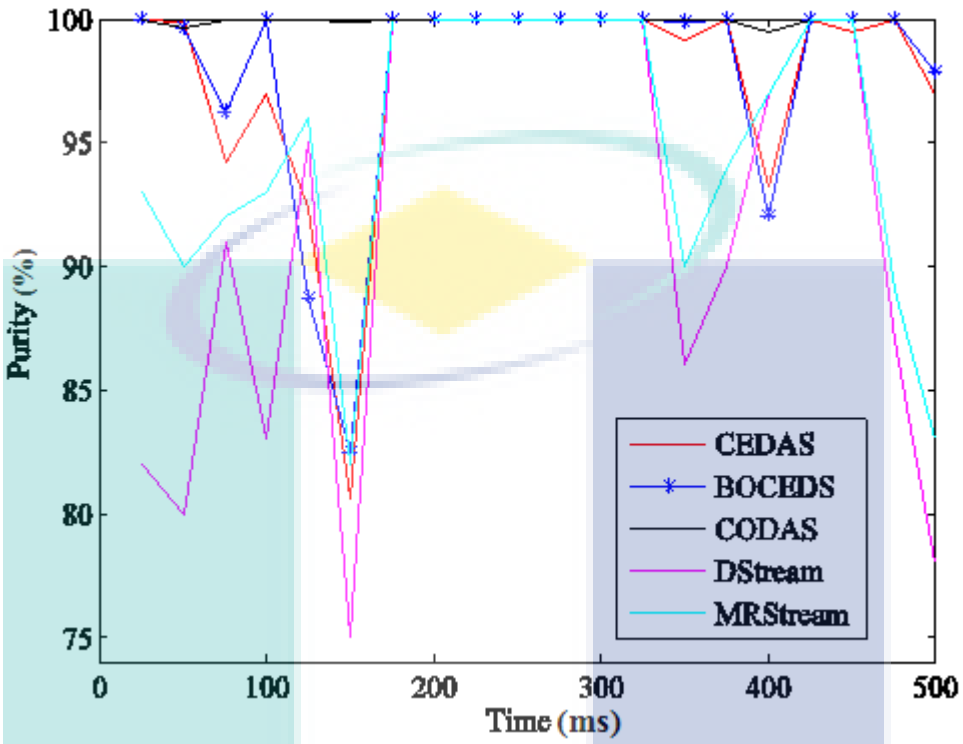


Figure 4.7 Purity for clustering of KDDCUP'99 data stream

As shown in Figure 4.7, CODAS exhibits the highest clustering purity for the experimental time period among the clustering techniques. Nevertheless, CODAS is a non-evolving clustering algorithm. The remaining algorithms are evolving clustering methods. The developed BOCEDS shows the maximum purity at most of the time period, whereas DStream shows the worst performance in terms of purity due to the appearance of outlier data that are made from errors. The contributing factor behind this success in BOCEDS is the online updating of the micro-cluster radius toward its local optimal in contrast to a unique and global radius of micro-clusters in CEDAS. The purity performance of BOCEDS is above 90% at all time periods, except for the time period of 150 (Figure 4.7). The purity of BOCEDS is also higher than that of CEDAS at nearly all time periods. Although the purity of MRStream is good, BOCEDS is still superior at most of the time periods. Figure 4.7 emphasizes the improvement in purity of BOCEDS.

4.3.3.2 Cluster Accuracy

The accuracy of clustering KDDCUP'99 using the developed BOCEDS is measured using Eq. 4.4 and recorded for different time periods. The clustering accuracy of another two recent highly accurate density-based clustering, namely, CODAS and CEDAS, is also measured on the same data stream. The cluster accuracy is measured for 500 time intervals spaced at 10K data points. Figure 4.8 plots the recorded accuracy and compares BOCEDS with CODAS and CEDAS.

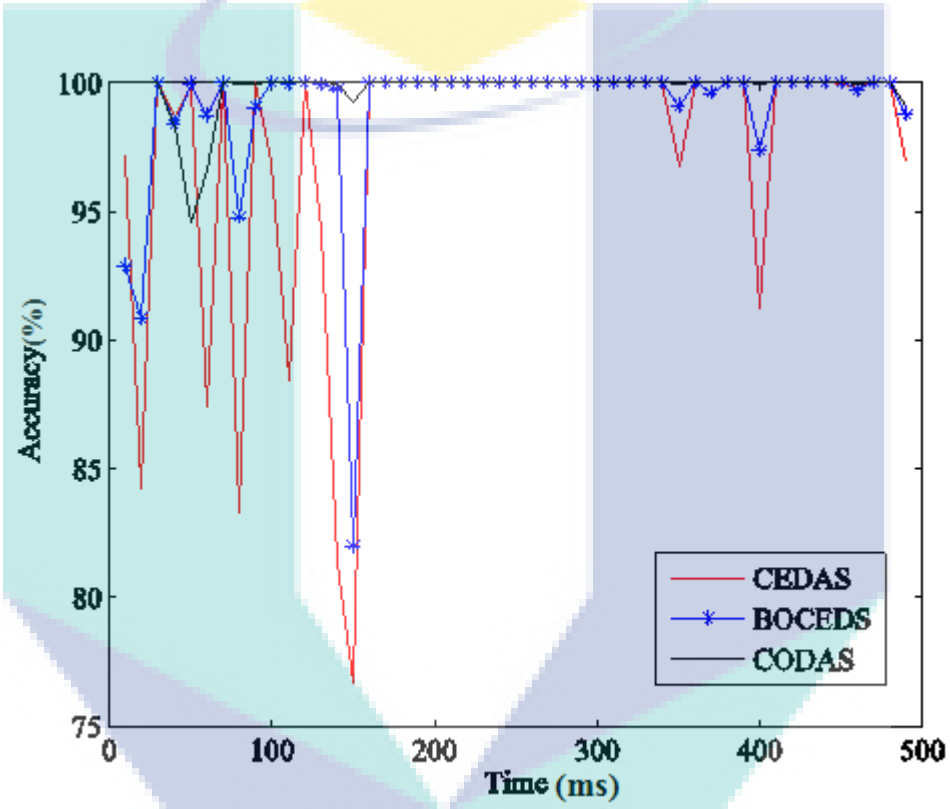


Figure 4.8 Accuracy for clustering of KDDCUP'99 data stream

Among BOCEDS, CEDAS, and CODAS, CODAS shows the highest clustering accuracy; however, the micro-clusters in CODAS do not evolve. Between the other evolving two clustering methods, the developed BOCEDS shows higher accuracy than CEDAS due to the same reason of improving cluster purity (Section 4.3.3.1). In Figure 4.8, the minimum accuracy in CEDAS is nearly 77%, which is found at approximately 150 time periods; meanwhile, the minimum accuracy is improved by more than 5% in the developed BOCEDS. In CEDAS, numerous downward spikes are found, which imply that the accuracy falls below 90% in considerable time period. On the contrary, only a single spike goes below 90% in BOCEDS accuracy. This result indicates that the

data points are more correct in BOCEDS than in CEDAS. Thus, the improvement in accuracy of BOCEDS is considerable (Figure 4.8).

4.3.3.3 Memory Efficiency

Hence, the number of micro-clusters in the developed BOCEDS is compared with the number of micro-clusters in CODAS, DenStream, and CEDAS when clustering the KDDCUP'99 data stream to demonstrate the memory efficiency. The result is plotted in Figure 4.9.

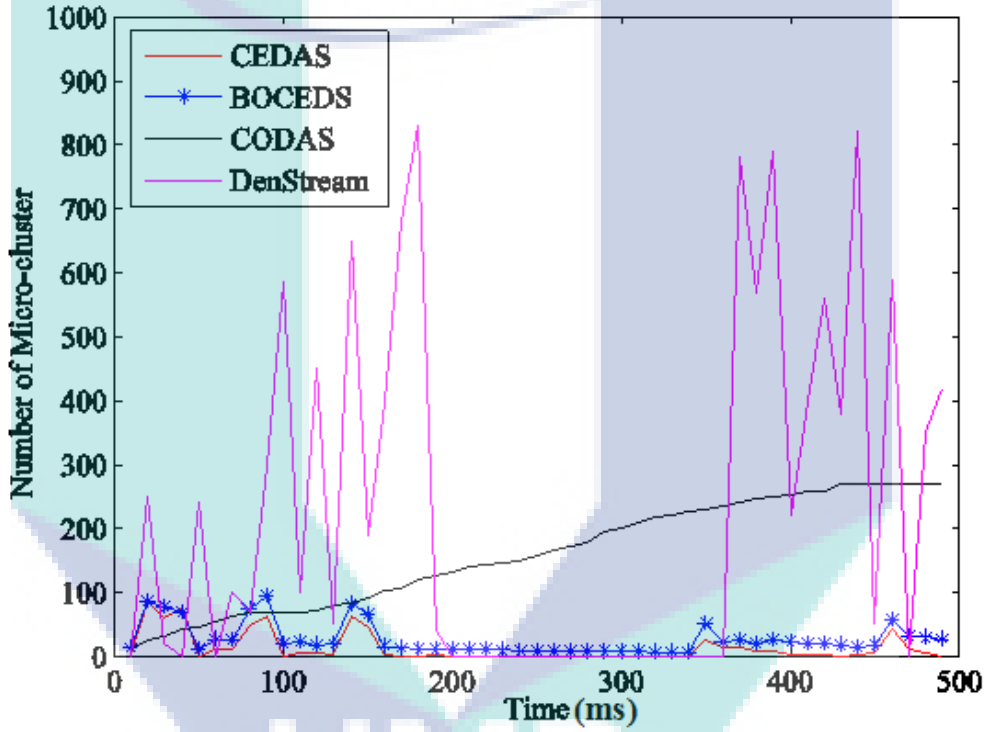


Figure 4.9 Memory usage in clustering of KDDCUP'99 data stream

According to Figure 4.9, the number of generated micro-cluster in CODAS shows an increasing trend because it is not an evolving clustering algorithm. The number of micro-cluster in DenStream is taken from the results presented by Wan et al. (Wan et al., 2009). Among the other evolving clustering methods, DenStream is the worst clustering algorithm in terms of memory efficiency as it stores the all of the outliers for future reference. The number of micro-clusters in CEDAS and BOCEDS is close to each other. However, the number of micro-clusters is more in BOCEDS than that in CEDAS. Although BOCEDS generates a lower number of core micro-cluster than CEDAS does, the number of micro-clusters is further increased by the weak micro-

cluster in BOCEDS. Moreover, the total micro-cluster is more in BOCEDS than in CEDAS. Thus, in comparison with CEDAS, BOCEDS shows a small amount of memory space penalty.

4.3.4 Parameter Sensitivity

Sensitivity analysis is an important metric to evaluate an algorithm, especially in the case when the algorithm takes some parameters from the user. Sensitivity analysis is an efficient and robust ways to realize the effect on the output of the algorithm due to the changes in input parameters (Tøndel et al., 2013). This evaluation metric has been well studied by many clustering algorithms like kDDBSCAN (Jungan et al., 2018), k-mean-sharp (Olukanmi & Twala, 2017), FDCA (Jinyin et al., 2017). In this subsection, the sensitivity of the developed BOCEDS has been analysed with respect to the density threshold of the clustering parameters and the micro-cluster radii in a similar manner as those in (Dong et al., 2018; Guha et al., 2001; Shao et al., 2018). The popular KDDCUP'99 (Bay et al., 2000) data stream has been used to measure the accuracy and purity for different parameter settings in our experiment.

4.3.4.1 Density Threshold ($Th_{density}$)

$Th_{density}$ is varied from 1 to 6 to study the behaviour of BOCEDS for different settings of density threshold. The clustering parameter $Decay$ is set to 1000, $R_{\min}$ is set to 0.06, and $R_{\max}$ is set to 0.12 (similar to the third experiment in Section 4.2.3. The clustering purity and accuracy have been measured using Eq. 4.3 and Eq. 4.4). Table 4.1shows the measured purity for different time periods.

In Table 4.1, the bold font values represent the maximum purity at different time periods for different values of density threshold. From the table, the measured purities are close to one another in most of the time periods. For example, in the time period of 50, the purities stay between 99.630 and 99.638. The maximum purity of 99.638 is found for $Th_{density} = 4,5,6$, which is larger than the other density threshold with a negligible amount. In some time periods (i.e., 100, 200, 250, 300, 450), the purities are 100% for all density threshold settings.

Table 4.1 Purity of clustering for KDDCUP'99 for different values of $Th_{density}$

Time (ms)	$Th_{density}=1$	$Th_{density}=2$	$Th_{density}=3$	$Th_{density}=4$	$Th_{density}=5$	$Th_{density}=6$
50	99.63	99.635	99.6375	**99.638**	99.638	99.638
100	100	100	**100**	100	100	100
150	80.708	81.565	**82.49752**	82.321	82.134	82.122
200	100	100	**100**	100	100	100
250	100	100	**100**	100	100	100
300	100	100	**100**	100	100	100
350	**98.229**	97.663	97.88945	97.765	97.967	98.028
400	90.053	90.922	**91.06759**	90.068	90.871	90.801
450	100	100	**100**	100	100	100
500	91.971	97.887	**97.88734**	95.189	93.95	92.636

In these time periods, the change in data stream is relatively low. On the contrary, purity is considerably lower for two time periods (i.e., 150 and 400) for all density threshold settings due to the frequent change in the content of data stream. The lowest purity is found in the time period of 150. In most of the time periods (i.e., 100, 150, 200, 250, 300,400, 450, and 500), the purity is highest for $Th_{density}=3$ setting. For the other two time periods, the purity of clustering KDDCUP'99 data stream is near the highest purity. Thus, the best purity result is found for $Th_{density}=3$ setting.

Table 4.2 shows the clustering accuracy for different time periods. The bold values represent the maximum accuracy at different time periods for different values of density threshold. Similar to purities, the accuracy values are close to one another for all threshold densities $(Th_{density}=1,2,34,5,6)$.

In all times periods, the accuracies for all $Th_{density}$ are close to 100%. Similar to purity values, the accuracies are 100% for some time periods (i.e., 100, 200, 250, 300, and 450) for all values of $Th_{density}$. The lowest purity is found in the time period of 150 for all density threshold settings due to the same reason for dropping the clustering purity. Moreover, 80% of time periods show the highest accuracy for $Th_{density}=3$.

Table 4.2 Accuracy of clustering the KDDCUP'99 for different values of $Th_{density}$

Time (ms)	$Th_{density}=1$	$Th_{density}=2$	$Th_{density}=3$	$Th_{density}=4$	$Th_{density}=5$	$Th_{density}=6$
50	99.985	99.985	**99.98547**	99.985	99.985	99.985
100	100	100	**100**	100	100	100
150	80.708	81.848	**81.99446**	81.848	81.771	81.815
200	100	100	**100**	100	100	100
250	100	100	**100**	100	100	100
300	100	100	**100**	100	100	100
350	98.943	96.216	99.10307	99.103	**99.165**	99.165
400	98.275	**98.585**	97.36508	97.365	98.081	98.081
450	100	100	**100**	100	100	100
500	92.266	98.765	**98.76543**	97.705	94.863	93.202

A density threshold setting less than the optimal value creates false clusters, which are originally outliers. By contrast, a density threshold more than the optimal, identifies some true clusters as outliers. The effect is reflected in Table 4.2, where an increasing trend in accuracy occurs in most of the time periods as the density threshold $(Th_{density})$ increases until it reaches 3, and the accuracy then decreases with increase in threshold value in most of time periods. Thus, the best accuracy is found for $Th_{density}=3$ setting.

4.3.4.2 Maximum and Minimum Radii

The clustering accuracies and purities are measured using Eq. 4.3 and Eq. 4.4 and for various radius settings to study the sensitivity of maximum $(R_{\max})$ and minimum $(R_{\min})$ radii. The clustering parameter *Decay* is set as 1000, and $Th_{density}$ is set to 3 (similar to the third experiment in Section 4.2.3).

The first study towards sensitivity analysis is to investigate the improvement in clustering accuracy and purity as a result of using the concept of local radius rather than global radius. For this purpose, the radius is varied from 0.06 to 0.12 with an increment of 0.01, and the result is compared with the range of radius setting. For the first seven cases, the minimum and the maximum radius of micro-cluster set to an identical value. This setting confirms that the algorithm uses a unique global radius. The last case uses the proposed range of radius where the minimum radius must be lower than the

maximum radius. For each case, the accuracy and purity is calculated in a window of 10000 data (using Eq. 4.3 and Eq. 4.4) and recorded. The mean accuracy and purity is computed from the recorded accuracy and purity values respectively. Figure 4.10 illustrates the improvement in cluster quality in terms of accuracy and purity when a range micro-cluster radius is used instead of a global optimal radius in clustering the KDDCUP'99 data stream.

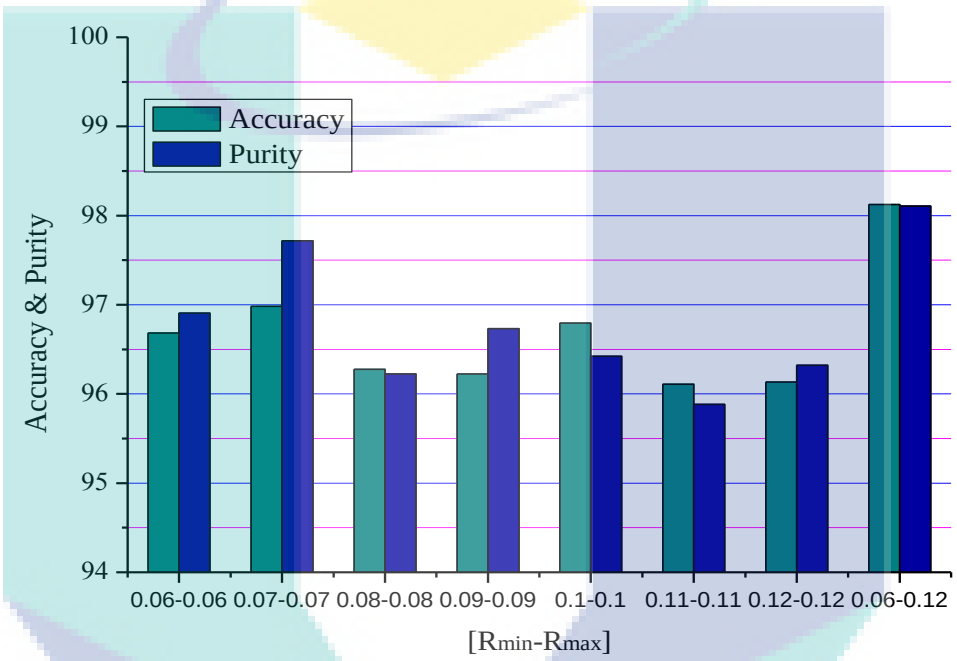


Figure 4.10 Accuracy and purity for identical $(R_{min} = R_{max})$ and radius range $(R_{min} < R_{max})$ in clustering of KDDCUP'99 data stream

From Figure 4.10, the accuracies and purities are consistently below 97% for all cases of global optimal radius setting, except for the setting of $R_{min} = R_{max} = 0.07$, where accuracy is approximately 97% and purity is 97.7%. On the contrary, the proposed local optimal radius concept shows a clear improvement in cluster quality, in which accuracy and purity are found as 98.995% and 98.107%, respectively. It is stated every micro-cluster sets its radius to local optimal value independently rather than using a predefined global radius for all micro-clusters (Section 3.2.2.3). This fact contributes to improve the cluster quality (accuracy and purity) in the last case where less sparse regions are presented than the first seven cases.

The experiment is extended to study the sensitivity of radius range $(R_{\min}, R_{\max})$ setting in KDDCUP'99 data stream. $R_{\min}$ varies from 0.03 to 0.08 with an increment value of 0.01, and $R_{\max}$ varies from 0.06 to 0.15 with an increment value of 0.01. For this experiment, the accuracy and purity are measured for every 10000 data points. The average accuracy and purity values are recorded in Table 4.3.

Table 4.3 Clustering accuracy for different settings of $[R_{\min}, R_{\max}]$

Accuracy		***Rmax***									
		0.06	**0.07**	**0.08**	**0.09**	**0.10**	**0.11**	**0.12**	**0.13**	**0.14**	**0.15**
Rmin	**0.03**	97.616	97.618	97.618	97.618	97.618	97.618	97.618	97.128	97.128	97.128
	0.04	97.682	97.942	97.174	97.313	97.164	97.174	97.784	97.775	97.772	97.772
	0.05	97.525	97.623	97.646	97.753	97.753	97.753	97.744	97.892	97.892	97.892
	0.06	96.682	97.939	97.971	97.918	97.918	97.918	**98.125**	97.979	97.996	97.996
	0.07	-	96.984	97.779	97.01	97.884	97.844	96.982	97.541	97.54	97.54
	0.08	-	-	96.275	97.116	97.51	97.459	97.458	97.448	97.448	97.448

For example, with $R_{\min} = 0.03, R_{\max} = 0.06$ setting, the average accuracy is 97.616%, and the purity is 97.734%. In Table 4.3, the maximum clustering accuracy and purity are found for $(R_{\min} = 0.06, R_{\max} = 0.12)$ setting. The accuracies and purities degrade by a small amount when the minimum radius is decreased from 0.06 to 0.03. Similarly, accuracies and purities increase when the maximum radius is decreased from 0.12 to 0.06.

Table 4.4 Clustering purity for different settings of $[R_{\min}, R_{\max}]$ purity

		Rmax									
Purity		**0.06**	**0.07**	**0.08**	**0.09**	**0.10**	**0.11**	**0.12**	**0.13**	**0.14**	**0.15**
Rmin	**0.03**	97.734	97.657	97.657	97.657	97.657	97.657	97.657	95.941	95.941	95.941
	0.04	97.643	97.731	97.79	97.817	97.817	97.817	97.817	96.084	96.084	96.084
	0.05	97.063	97.38	97.356	97.43	97.43	97.43	97.421	95.771	95.771	95.771
	0.06	96.906	97.205	97.532	97.361	97.02	97.717	**98.107**	97.064	95.957	95.957
	0.07	-	97.716	96.589	96.581	96.662	96.588	96.657	96.54	96.553	96.553
	0.08	-	-	96.224	96.568	96.423	96.338	96.366	96.421	96.414	96.414

This degradation is due to the fact that each cluster does not contain sufficient data points for being a cluster, and they are falsely identified as outliers. A minimum radius greater than 0.06 also results in lowering the accuracy and purity; in these cases,

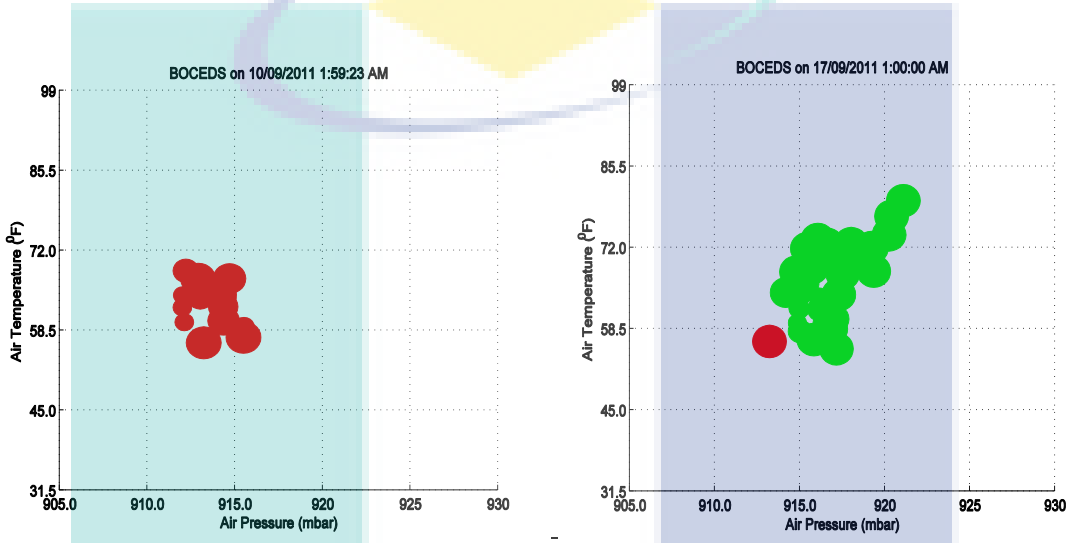
a negative separation exists between clusters and outliers, and many outliers are falsely considered main clusters. The same trend is found when the maximum radius is greater than 0.12. ($R_{\min} = 0.06, R_{\max} = 0.12$) is a good range of radius, which dampens the effects of outliers. However, Table 2 indicates that when a deviation exists in setting the radius parameters from their optimal value, accuracy and purity degrade by a small amount because the micro-cluster radius is recursively updated to its local optimal. Therefore, accuracy and purity remain high despite a deviation in selecting the radius parameters.

4.4 Case Study: Clustering of Weather Data Stream Using BOCEDS

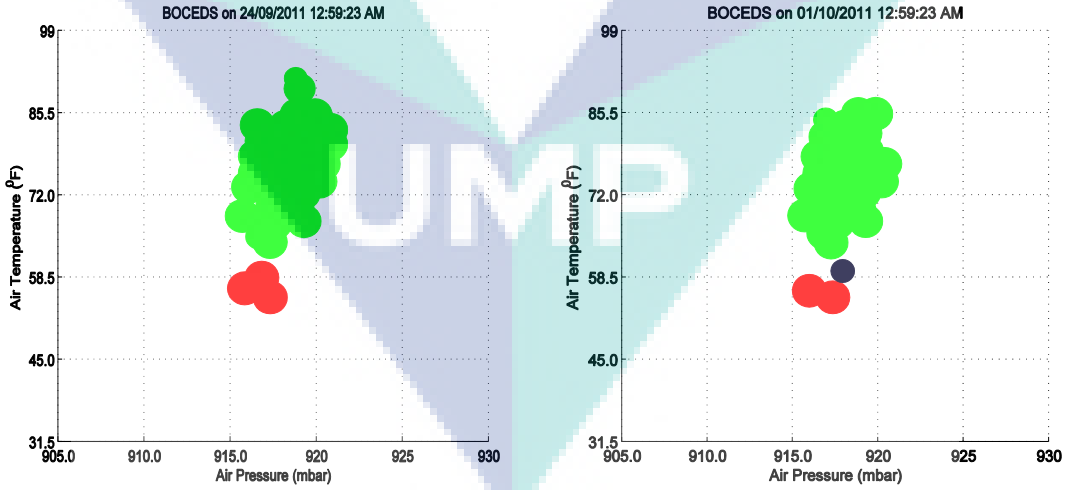
Changes in clusters are detected and tracked as time progresses to investigate the evolving behaviour of data stream (Shao et al., 2018). The developed BOCEDS algorithm has been applied to atmospheric data stream of San Paulo, Brazil City. The data stream is downloaded from Kaggle dataset repository(Jose, 2018). In our clustering process, only two dimensions (i.e., air pressure and temperature) are used to visualize the clusters in a two-dimensional environment. The data stream is captured at a 1-min interval for a time period of 1 year, 11 months, and 6 days from September 10 to August 16. The data stream contains a total of 1048576 data points. The data points appear in the BOCEDS sequentially in air pressure–temperature pairs to mimic an online data stream. This data stream is used to examine the capability of BOCEDS to detect the temporal drift in real data stream similarly to (Hyde et al., 2017).For the clustering task, the data points are normalized to a range of 0 to 1. The air pressure has an actual value range of 905.00mbar to 929.50mbar and the air temperature of 31.64 ^{o}F to 99.50 ^{o}F. It is demonstrated how BOCEDS handles the three types of drifting (i.e., short, medium, and long terms) in the data stream. The *short-term* is defined as 1 week, *medium-term* is defined as 1 month, and *long-term* is defined as 6 months. The clustering parameter $Th_{density}$ is set to 1 to observe the entire data space that contains the data points. The radius parameter $R_{\min}$ is set to 0.02, whereas $R_{\max}$ is set to 0.04. Similar to (Hyde et al., 2017), these values are estimated by considering historical data points at the distances from the main clusters to data points that are considered as an outlier.

4.4.1 Short-term Drift Analysis

The variation in data stream from September 10, 2011 to October 1, 2011 is plotted to visualize the *short-term* drift. The plots with clusters are drawn for 1-week (7 days) interval. The micro-clusters with identical colors belong to the same clusters. Figure 4.11(a)–Figure 4.11(d) show the cluster at four distinct dates for *short-term* drift analysis. The clustering plot in Figure 4.11(b) (September 10, 2011) is remarkably different from that in Figure 4.11(b) (September 17, 2011).



(a) Clustering result on Sep 10, 2011 (b) Clustering result on Sep 17, 2011



(c) Clustering result on Sep 24, 2011 (d) Clustering result on Oct 01, 2011

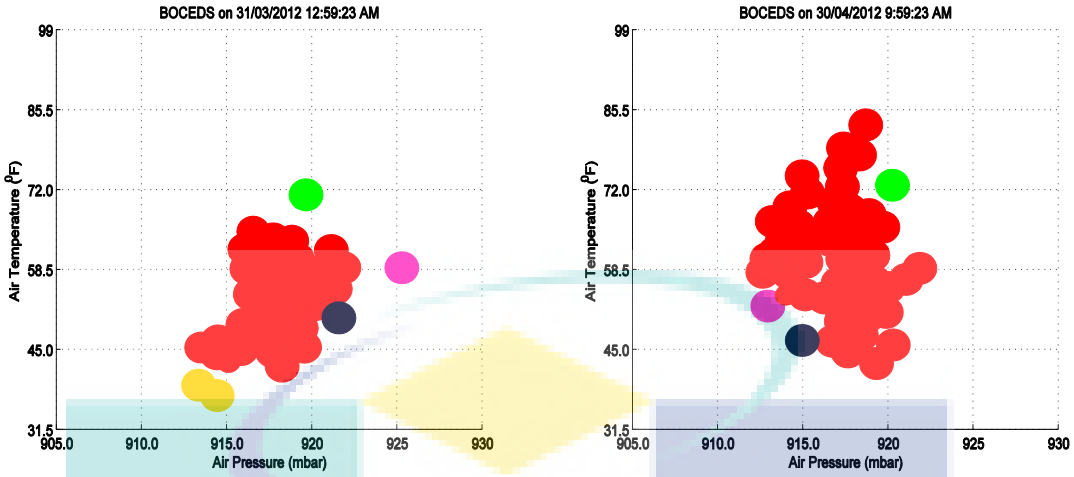
Figure 4.11 Plots of BOCEDS clustering from September 10, 2011 to October 1, 2011 with 1-week interval for *short-term* drift visualization

On September 10, only one cluster is generated. After 7 days, on September 17, two clusters are generated. The green-colored cluster has multiple micro-clusters. By contrast, the red-colored micro-cluster creates a single cluster. In Figure 4.11(c), although the number of total micro-clusters increases, the number of clusters remains two on September 24, 2011. Some micro-clusters are dying out due to their evolving nature, and some micro-clusters are generated. On October 1, 2011 the total number of generated clusters is 3, where one cluster contains a single micro-cluster, another cluster contains two micro-clusters, and the third cluster contains more than two micro-clusters. Thus, the first cluster is circular, the second cluster is ellipsoidal, and the third one is arbitrarily shaped. Despite the first 2 weeks showing a noticeable difference in terms of *short-term* drifting, the spread of data points in data space is consistent for the preceding two weeks and shows a less noticeable difference. This phenomenon demonstrates that the weather data stream changes over short time periods and how the developed BOCEDS algorithm follows these changes to detect the *short-term* drift in a fully online approach.

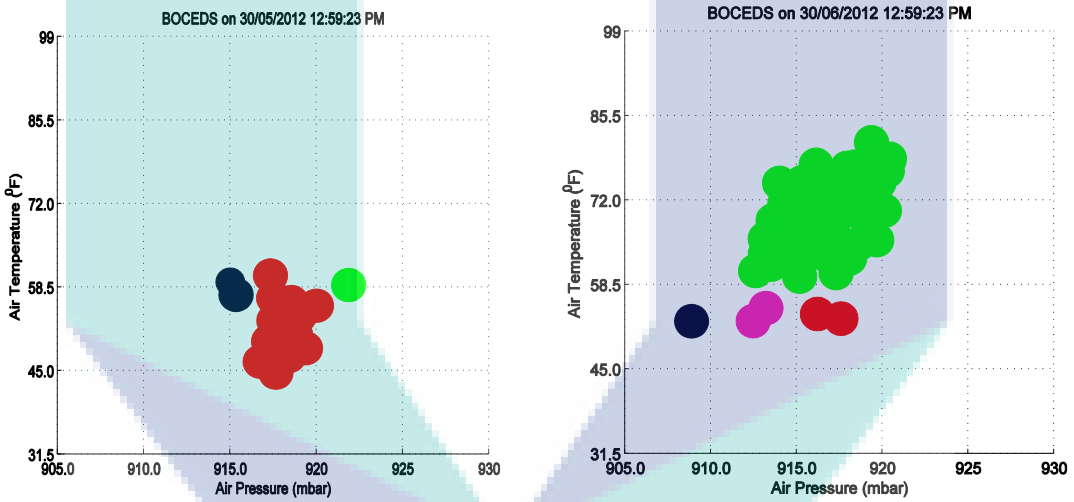
4.4.2 Medium-term Drift Analysis

The weather data stream from March 31, 2012 to June 30, 2012 with 1-month interval is considered to present the *medium-term* drift. The clustering results are plotted at four consecutive months for *medium-term* drift analysis, as shown in Figure 4.12(a)–(d). Figure 4.12 depicts that the clustering results are clearly distinguishable from one another.

From the figures, five clusters exist on March 31, 2012. In the next month, some micro-clusters are newly generated, whereas some are removed due to the evolving nature of the experimental data stream. Some micro-clusters in Figure 4.12 (a) are also present in Figure 4.12(b) because they receive data points in the interval. On April 30, 2012, four clusters are present. In the next two months, a considerable change in the data stream is observed. In the two months, frequent changes in the data stream occur, and BOCEDS follows these changes to handle the drift. To illustrate, only three clusters are present in Figure 4.12(c) that illustrates that the weather in the month of May is less bumping. However, comparing to the weather of May, the weather in June is more changing.



(a) Clustering result on March 31, 2012 (b) Clustering result on April 30, 2012



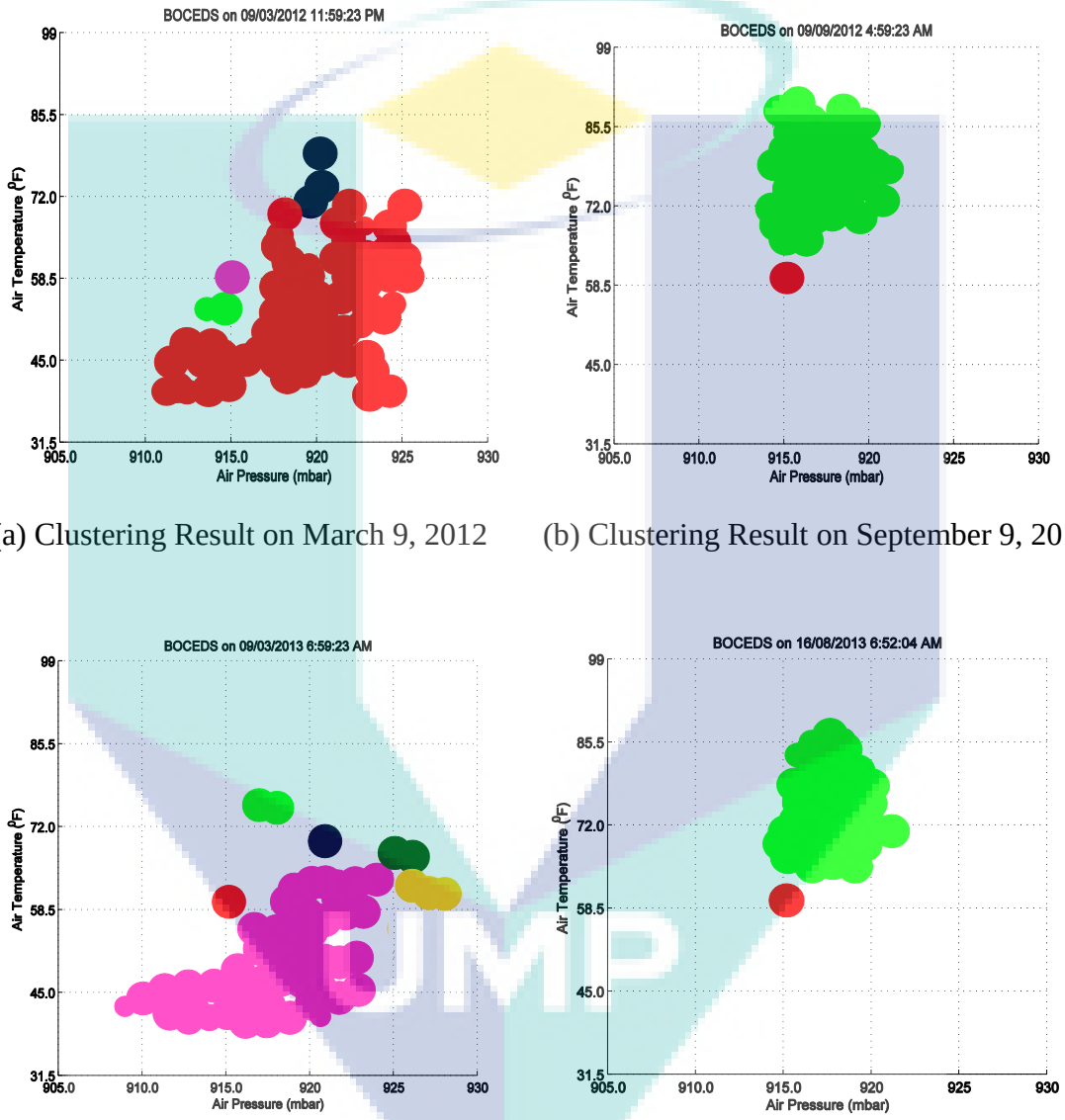
(c) Clustering result on May 30, 2012 (d) Clustering result on June 30, 2012

Figure 4.12 Plots of BOCEDS clustering from March 31, 2012 to June 30, 2012 with 1-month interval for *medium-term* drift visualization

From the figures, it can be seen that that weather of San Paulo city changes remarkably from one month to next month. The changes in weather are well visualized in the cluster analysis by the developed BOCEDS algorithm. Thus, Figure 4.12 demonstrates the capability of the algorithm to identify the *medium-term* (1 month) drift.

4.4.3 Long-term Drift Analysis

To present the *long-term* drift, the BOCEDS clustering result is plotted of the weather data stream from March 31, 2012 to June 30, 2012 with 6-month interval. The plots are shown in Figure 4.13(a)– Figure 4.13(d) for *long-term* drift analysis.



(a) Clustering Result on March 9, 2012 (b) Clustering Result on September 9, 2012

(c) Clustering result onMarch9, 2013 (d) Clustering result on August 16, 2013

Figure 4.13 Plots of BOCEDS clustering from March 9, 2012 to August 16, 2013 with 6-month interval showing for *long-term* drift visualization

The clustering results in Figure 4.13(a)–Figure 4.13(d) show that the micro-clusters are generating and fading away from the first half of the year to the second half of the year. The most noticeable point in Figure 4.13 is that a similarity occurs between

Figure 4.13(a) and Figure 4.13(c) and between Figure 4.13(b) and Figure 4.13(d) to some level. This trend illustrates that the weather is repeated in a yearly basis. Thus, Figure 4.13 demonstrates how the developed BOCEDS clustering algorithm follows the long-term drift in weather data stream.

4.5 Summary

In this chapter, the characteristics of the developed BOCEDS algorithm have been described based on some synthetic and practical data stream. Seven well-known evaluation metrics including cluster formation, noise sentivity, processing speed, scalability, cluster accuracy, cluster purity, memory efficiency are selected to show the high quality of the developed algorithm. The metrics are calculated on selected time units, stream speeds and horizons. Both of syntactic and real word data stream are used for the evaluation of BOCEDS. The real and synthetic data stream are chosen from the reviewed literature which are most used. They have variety in size, number of clusters, and differences in their densities. A wide spectrum of experiments has been conducted in this section as well.

The ability of the developed algorithm to generate clusters with varying the micro-cluster radius has been evaluated for both of clean and noisy Mackey-Glass data streams (Section 4.3.1.1). The evaluation includes the evolution of clusters and cluster change as time progresses. The result shows that BOCEDS is able to generate clusters in both clean and noisy data stream environment. The noise sensitivity is also measured by numerical analysis that shows an improvement in noise detection by developed algorithm when compared to the existing algorithm To evaluate the processing time characteristics, BOCEDS has been applied on helical data stream (Section 4.2.2). The processing time has been described by measuring and analysing the mean data processing time by the algorithm. The experiment also evaluates the behaviour of BOCEDS over low to high dimensional helical data stream. This evaluation describes the scalability properties. BOCEDS shows linear scalability on both the number of clusters and the number of data dimensions. BOCEDS is faster and more scalable than the other aligned clustering algorithms in the literature. To measure the cluster quality, BOCEDS is applied on practical KDDCUP'99 network data stream (Section 4.2.3). The cluster quality is described in terms of cluster accuracy and purity. The results show that the developed algorithm provides more pure and accurate cluster than the existing

CEDAS, CODAS, DStream and MRStream algorithms. The sensitivity of algorithmic parameters is also measured by varying important parameters of BOCEDS with numerical analysis and described in details. The sensitivity analysis determines the best range for prominent parameters of the algorithm. Finally, BOCEDS has been executed on real word weather data stream to show the capability of algorithm to detect the drift in data stream for handling the evolving characteristics (Section 4.3).

Summing up, BOCEDS algorithm clearly shows the best performance in terms of cluster accuracy and cluster purity among the aligned clustering algorithms due to maintaining the local optimal radius of micro-cluster in an online manner and buffering the micro-cluster. Moreover, the algorithm provides better processing speed and scalability comparing to other exiting algorithms by formulating non-linear energy function and pruning the irrelevant cluster. Thus, it is proved from the experimental result that BOCEDS is an effective and efficient density-based algorithm for clustering of evolving data stream.

CHAPTER 5

CONCLUSION

5.1 Introduction

Analysis of data stream is beneficial for several IT-based applications such as traffic management, anomaly detection and weather forecasting. Data from data stream arrives continuously over time with high speed. The size of a stream grows rapidly and become unbounded. Clustering of data stream helps the data mining scientist to extract the pattern from data stream. Density-based clustering is one kind of clustering technique that has gained the remarkable popularity among all clustering techniques due to its excellent clustering performance over data stream. The aim of the current study is to design a new density-based clustering algorithm to handle the challenges of evolving data stream clustering efficiently. The objectives of this research study were as follows:

i. To design an online clustering algorithm based on the concept of local optimal radius and irrelevant micro-cluster buffering.

ii. To implement the algorithm by adapting a non-linear procedure for updating the micro-cluster energy and pruning the micro-clusters.

iii. To evaluate the performance of the developed algorithm against selected benchmark functions as case studies.

Addressing the first objective, a new density-based fully online clustering algorithm called BOCEDS in order to achieve the first objective. The algorithm introduces the concept of local radius where each micro-cluster maintains its own value of radius independently. This fact confirms that the micro-clusters as well as clusters

have less or no sparse region. Based BOCEDS identifies the irrelevant micro-clusters based on its energy and stores them into special buffer storage. The algorithm works in two distinct stages. In the first stage, the data points map to a micro-clusters in the current model or creates a new micro-clusters in case the data point lies in the data space outside all micro-clusters. The information of newly mapped micro-cluster is updated recursively to enable an online process (Section 3.2.2.3). The radius of newly mapped micro-cluster is updated towards its optimal that is local to the micro-cluster. The forgetting mechanism is adapted to formulate the micro-cluster updating procedure. This operation reduces the dependency on user to set the optimal value of micro-cluster radius prior to the execution of algorithms. The micro-cluster with zero or negative energy is identified as irrelevant micro-cluster and stored in a special storage called buffer. In the second stage, the micro-clusters except irrelevant micro-clusters generate micro-clustering graph based on their connectivity to compute the clusters. The connected micro-clusters form an arbitrary shaped cluster which is maintained in an online manner. The second stage confirms that the algorithm has the updated clustering result at all the time period. Both of the stages of BOCEDS algorithm are online which ensures that BOCEDS is an online clustering algorithm.

Every time a micro-cluster receives a new data, its energy is updated based on positional information of the data in the data space. The energy updating procedure is described by a non-linear formula. The formula is designed by adapting the gravity law of Newton. The recursive nature of this non-linear formula again supports the online process of BOCEDS algorithm. The energies of micro-clusters, except the newly mapped micro-cluster are reduced by a specific amount. The micro-clusters with zero or negative energy are identified as irrelevant micro-clusters and stored in buffer with new energy. Moreover, a pruning operation is introduced to identify the micro-clusters with zero or negative energy in buffer and considered as totally irrelevant. The totally irrelevant micro-clusters are pruned out from buffer to ensure no out-dated micro-cluster is stored to represent the current data stream. On the other, a micro-cluster is called temporary irrelevant if a data mapped to that micro-clusters in buffer. In this case, this micro-cluster is considered for cluster generation and move to primary memory from buffer. This operation prevents the frequent creation and removal of micro-cluster. This two operations help to achieve the second objective successfully.

To achieve the final objective, the BOCEDS algorithm is evaluated against the standard performance metrics in the field of density-based clustering. The algorithm executed on both of syntactic and practical data stream to evaluate the effectiveness and efficiency. From the experiment, it is visualized that BOCEDS is able to form the micro-clusters as well as clusters at the dense regions in clean and noisy data stream successfully (Section 4.3.1.1). It is also confirmed that new micro-clusters are created and old micro-clusters are removed from the system to prove the correct functionality of the algorithm over evolving data stream. The experiment for measuring noise sensitivity found that BOCEDS is able to identify about 100% noise where the existing algorithm detects about 70% noise (Section 4.3.1.2). The data processing time of the developed algorithm is lower than other popular online density-based algorithms (Section 4.3.2.1). To test the scalability of BOCEDS algorithm, the change in processing time is tracked for low to high dimensional data stream and compared with other density-based clustering algorithms like CODAS, CEDAS, CluStream, DenStream (Section 4.3.2.2). It is shown that BOCEDS shows the best performance in terms of scalability. BOCEDS demonstrates its capability to generate high-quality clusters in practical network attacks in KDDCUP'99 data stream. The result shows that the clusters generated in BOCEDS are purer and more accurate with a lower variance than those of similar existing clustering algorithms (Sections 4.3.3.1-4.3.3.2). Nevertheless, the memory requirement of the developed algorithm is relatively more than that of the fully online density-based CEDAS due to storing the temporarily irrelevant micro-clusters in a special buffer. The memory requirement, however, remains considerably lower than those of other clustering algorithms (Section 4.3.3.3). The parameter sensitivity experiment illustrates that BOCEDS still generates high-quality clusters in a small deviated optimal radius, density threshold, and decay setting (Sections 4.3.2.3, 4.3.4.1 and 4.3.4.2). From the experiment, it is stated that BOCEDS is less sensitive to its parameters. The execution of BOCEDS on a real-world weather data stream demonstrates the capability of the developed algorithm to generate and evolve clusters in a non-stationary dynamic environment (Section 4.4).

To summarize, the developed BOCEDS algorithm is a fully online algorithm for clustering the noisy evolving data stream into arbitrarily shaped. This algorithm outperforms the existing algorithms for density-based clustering in terms of noise sensitivity, cluster quality, processing speed and scalability. BOCEDS is also proved to

be less sensitive to its parameters. Therefore, the developed BOCEDS algorithm shows the effectiveness of the operational framework to generate clusters in evolving data stream.

5.2 Contributions

There are a number of density-based clustering algorithms for data stream. However, a majority of them are offline clustering algorithm those are designed for static data set; not for data stream. Some of them online-offline clustering those suffers from storage problem and not ideal for data stream. Yet only a few algorithms are fully online. However, they suffer from low cluster quality and low noise sensitivity problem due to pre-setting of algorithmic parameter like micro-cluster radius. Moreover, they have high computation time and low scalability as the micro-clusters are created and deleted frequently to handle the evolving nature of data stream. Therefore, a new density-based clustering was presented in this thesis to overcome the aforementioned problems. Furthermore, according to data stream properties, the challenges in clustering data streams had to be considered in the developed algorithm. The specific contributions of this study include

- a new density-based clustering algorithm for data streams that generate arbitrary clusters in a fully online manner.
- a new online procedure for adapting the micro-cluster radius based on the forgetting mechanism to improve the cluster quality and noise sensitivity.
- a new non-linear procedure for computing the micro-cluster energy based on the Newton's gravity law to handle the evolving property efficiently.
- a new mechanism for micro-cluster buffering and pruning to improve the processing time.
- a new method for numerical analysis of noise sensitivity of algorithm
- an extensive evaluation

5.3 Limitations of Current Study

This study presents a fully online clustering algorithm that shows better performance when comparing to other algorithms in literature. However, the study has some limitations as follows:

i. It is observed that, the developed BOCEDS requires more memory by a small amount comparing to other density-based clustering algorithm. This fact makes the algorithm less memory efficient. This issue need to be solved in future to handle the data stream efficiently.

ii. The algorithm uses two radius parameters (maximum and minimum radius). Though, the algorithm improves cluster quality, noise sensitivity and processing speed, the number of algorithm parameter increased. However, an ideal should minimize the number of user dependent parameter.

iii. In this study, Euclidean distance is used for mapping the newly arrived data into micro-cluster. However, more analysis is required to determine if other kinds of distances such as Minkowski distance, Manhattan distance, Chebyshev distance, Cosine distance can increase the quality.

5.4 Future Research Directions

Data stream clustering is an unsupervised learning technique in the field of data stream mining. Density based clustering algorithm requires some parameters to be set and the performance of clustering heavily depends on the optimality of these parameters. This fact generates the scarcity of designing an appropriate algorithm that can automatically update its parameters towards their optimal value. Also some temporarily irrelevant clusters need to be identified to enhance the clustering performance in an evolving application environment. These issues are solved in this research. However, this research opens some research issues in the future:

i. This research takes the constant value of density threshold from application user. A deviation of this parameter from its optimal value affects the noise detection result and clustering result remarkably. Therefore, density-based clustering algorithm is still desired that can adapt all of its parameters in an online basis.

ii. It is desired for any algorithm to process all types of attributes as many IT-based applications produce data stream that contains textual, categorical or mixed attributes. However, BOCEDS is able to generate clusters from the data stream that contains only numerical attributes. To make attribute type independent, the current BOCEDS algorithm needs extension. The current research can be extended in future to process all types of data stream.

iii. Low data processing time and low memory space are two vital requirements of data stream clustering algorithms. The current research shows good performance in terms of these two criteria. The future research can target to increase the processing speed and reduce the required memory further.

iv. The developed algorithm has been applied on real world weather data stream that shows an excellent performance in terms of drift detection. The future work would integrate the algorithm towards designing several intelligent systems in real time decision making like anomaly detection in financial transactions, attack identification on security system, tracking malicious activities on social networks, pattern detection on biomedical images, and so on.

REFERENCES

Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., & Sohler, C. (2012). StreamKM++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics, 17*, 2.4.

Aggarwal, C. C., Philip, S. Y., Han, J., & Wang, J. (2003, September 12-13). *A framework for clustering evolving data streams.* Paper presented at the 29th International Conference on Very Large Databases, Berlin, Germany.

Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., & Taha, K. (2015). Efficient machine learning for big data: A review. *Big Data Research, 2*(3), 87-93.

Albertini, M. K., & Mello, R. F. D. (2018). Estimating data stream tendencies to adapt clustering parameters. *International Journal of High Performance Computing and Networking, 11*(1), 34-44.

Amini, A., & Wah, T. Y. (2012, December 21-22). *DENGRIS-Stream: A density-grid based clustering algorithm for evolving data streams over sliding window.* Paper presented at the International Conference on Data Mining and Computer Engineering, Bangkok, Thailand.

Amini, A., Wah, T. Y., & Saboohi, H. (2014). On density-based data streams clustering algorithms: A survey. *Journal of Computer Science and Technology, 29*(1), 116-141.

Ansah, J., Kang, W., Liu, L., Liu, J., & Li, J. (2018). *Information Propagation Trees for Protest Event Prediction*, Cham.

Baruah, R. D., & Angelov, P. (2012, June 10-15). *Evolving local means method for clustering of streaming data.* Paper presented at the 2012 IEEE International Conference on Fuzzy Systems, Brisbane, QLD, Australia.

Baruah, R. D., & Angelov, P. (2014). DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models. *IEEE Transactions on Cybernetics, 44*(9), 1619-1631.

Bay, S. D., Kibler, D., Pazzani, M. J., & Smyth, P. (2000). The UCI KDD archive of large data sets for data mining research and experimentation. *ACM SIGKDD Explorations Newsletter, 2*(2), 81-85.

Bhatnagar, V., Kaur, S., & Chakravarthy, S. (2014). Clustering data streams using grid-based synopsis. *Knowledge and Information Systems, 41*(1), 127-152.

Blazic, S., & Skrjanc, I. (2019). Incremental Fuzzy C-regression Clustering from Streaming Data for Local-model-network Identification. *IEEE Transactions on Fuzzy Systems*, 1-1.

Bohm, C., Railing, K., Kriegel, H. P., & Kroger, P. (2004, November 01-04). *Density connected clustering with local subspace preferences.* Paper presented at the 4th IEEE International Conference on Data Mining, Brighton, UK.

Bouguettaya, A., Yu, Q., Liu, X., Zhou, X., & Song, A. (2015). Efficient agglomerative hierarchical clustering. *Expert systems with applications, 42*(5), 2785-2797.

Bryant, A., & Cios, K. (2018). RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Transactions on Knowledge and Data Engineering, 30*(6), 1109-1121.

Can, U., & Alatas, B. (2017). Big social network data and sustainable economic development. *Sustainability, 9*(11), 2027.

Cao, F., Estert, M., Qian, W., & Zhou, A. (2006, April 20-22). *Density-based clustering over an evolving data stream with noise.* Paper presented at the 2006 SIAM International Conference on Data Mining, Bethesda, MD, USA.

Chen, J., Lin, X., Xuan, Q., & Xiang, Y. (2018). FGCH: A fast and grid based clustering algorithm for hybrid data stream. *Applied Intelligence*, 1-17.

Chen, J. Y., & He, H. H. (2016). A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. *Information Sciences, 345*, 271-293.

Chen, Y., & Tu, L. (2007, August 12 - 15). *Density-based clustering for real-time stream data.* Paper presented at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA.

Chenaghlou, M., Moshtaghi, M., Leckie, C., & Salehi, M. (2018). *Online clustering for evolving data streams with online anomaly detection.* Paper presented at the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Cham.

Cheng, W., Wang, W., & Batista, S. (2018). Grid-based clustering *Data Clustering* (pp. 128-148): Chapman and Hall/CRC.

Ding, S., Zhang, J., Jia, H., & Qian, J. (2016). An adaptive density data stream clustering algorithm. *Cognitive Computation, 8*(1), 30-38.

Dong, S., Liu, J., Liu, Y., Zeng, L., Xu, C., & Zhou, T. (2018). Clustering based on grid and local density with priority-based expansion for multi-density data. *Information Sciences, 468*, 103-116.

Donoho, D. L. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture, 1*, 32.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, 26*(1), 29-41.

Eberhart, R. C., Shi, Y., & Kennedy, J. (2001). *Swarm Intelligence*: Elsevier.

Esposito, C., Ficco, M., Palmieri, F., & Castiglione, A. (2015). A knowledge-based platform for Big Data analytics based on publish/subscribe services and stream processing. *Knowledge-Based Systems, 79*, 3-17.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). *A density-based algorithm for discovering clusters in large spatial databases with noise.* Paper presented at the 2nd International Conference on Knowledge Discovery and Data Mining.

Fahy, C., Yang, S., & Gongora, M. A. (2018). Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams. *IEEE Transactions on Cybernetics, 49*(6), 2215-2228.

Fan, W., & Bifet, A. (2013). Mining big data: Current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter, 14*(2), 1-5.

Fong, S., Wong, R., & Vasilakos, A. (2016). Accelerated PSO swarm search feature selection for data stream mining big data. *IEEE Transactions on Services Computing*(1), 1-1.

Forestiero, A., Pizzuti, C., & Spezzano, G. (2013). A single pass algorithm for clustering evolving data streams based on swarm intelligence. *Data Mining and Knowledge Discovery, 26*(1), 1-26.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science, 315*(5814), 972-976.

Gao, J., Li, J., Zhang, Z., & Tan, P. N. (2005, May 18-20). *An incremental data stream clustering algorithm based on dense units detection.* Paper presented at the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hanoi, Vietnam.

Garofalakis, M., Gehrke, J., & Rastogi, R. (2016). Data stream management: A brave new world. In M. Garofalakis, J. Gehrke & R. Rastogi (Eds.), *Data Stream*

Management: Processing High-Speed Data Streams (pp. 1-9). Berlin, Heidelberg: Springer Berlin Heidelberg.

Glass, L., & Mackey, M. (2010). Mackey-glass equation. *Scholarpedia*, *5*(3), 6908.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., . . . Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, *106*(9), 1469-1495.

Gudenas, B. L., Wang, J., Kuang, S. Z., Wei, A., Cogill, S. B., & Wang, L. J. (2019). Genomic data mining for functional annotation of human long noncoding RNAs. *Journal of Zhejiang University-SCIENCE B*, *20*(6), 476-487.

Guha, S., Rastogi, R., & Shim, K. (2001). Cure: An efficient clustering algorithm for large databases. *Information Systems*, *26*(1), 35-58.

Han, J. (2005). *Data mining: Concepts and techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques* (3rd ed.). San Francisco, CA, USA: Elsevier.

Hannachi, A., & Trendafilov, N. (2017). Archetypal analysis: Mining weather and climate extremes. *Journal of Climate*, *30*(17), 6927-6944.

Hassani, M., Spaus, P., Gaber, M. M., & Seidl, T. (2012, September 17-19). *Density-based projected clustering of data streams.* Paper presented at the 6th International Conference on Scalable Uncertainty Management, Marburg, Germany.

Hyde, R., & Angelov, P. (2015, June 24-26). *A new online clustering approach for data in arbitrary shaped clusters.* Paper presented at the 2nd IEEE International Conference on Cybernetics Gdynia, Poland.

Hyde, R., Angelov, P., & MacKenzie, A. (2017). Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Information Sciences*, *382*, 96-114.

Isaksson, C., Dunham, M. H., & Hahsler, M. (2012, July 13-20). *SOStream: Self organizing density-based clustering over data stream.* Paper presented at the International Workshop on Machine Learning and Data Mining in Pattern Recognition, Berlin, Heidelberg.

Jacques, J., & Preda, C. (2014). Functional data clustering: A survey. *Advances in Data Analysis and Classification*, *8*(3), 231-255.

Jia, C., Tan, C., & Yong, A. (2008, September 25-26). *A grid and density-based clustering algorithm for processing data stream.* Paper presented at the 2nd International Conference on Genetic and Evolutionary Computing, Jingzhou, Hubei, China.

Jinyin, C., Huihao, H., Jungan, C., Shanqing, Y., & Zhaoxia, S. (2017). Fast Density Clustering Algorithm for Numerical Data and Categorical Data. *Mathematical Problems in Engineering, 2017*, 15.

Jose, J. (2018). *Minute Weather*. Retrieved from: https://www.kaggle.com/julianjose/minute-weather

Jungan, C., Jinyin, C., Dongyong, Y., & Jun, L. (2018). A -Deviation Density Based Clustering Algorithm. *Mathematical Problems in Engineering, 2018*, 16.

Kakkar, S., Singh, S., Singh, S., & Banga, V. K. (2017, November 19). Investigations on the performance of fuzzy logic system when evolved using genetic algorithm for different number of fuzzy rules. Paper presented at the 30th International Conference on Instrumentation, Electrical and Electronics Engineering, Bangalore,India.

Kaufman, L., & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis* (Vol. 344): John Wiley & Sons.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems, 9*(1), 1-23.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*(1), 59-69.

Kosmidis, I., & Karlis, D. (2016). Model-based clustering using copulas with applications. *Statistics and Computing, 26*(5), 1079-1099.

Kranen, P., Assent, I., Baldauf, C., & Seidl, T. (2011). The ClusTree: Indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems, 29*(2), 249-272.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion, 37*, 132-156.

Kumar, K. M., & Reddy, A. R. M. (2016). A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method. *Pattern Recognition, 58*, 39-48.

Li, J., Maier, D., Tufte, K., Papadimos, V., & Tucker, P. A. (2005, June 14-16). *Semantics and evaluation techniques for window aggregates in data streams.* Paper presented at the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA.

Lin, J., & Lin, H. (2009). *A density-based clustering over evolving heterogeneous data stream.* Paper presented at the 2009 ISECS International Colloquium on Computing, Communication, Control, and Management.

Liu, L. X., Guo, Y. F., Kang, J., & Huang, H. (2009, December 15-18). *A three-step clustering algorithm over an evolving data stream.* Paper presented at the IEEE International Conference on Intelligent Computing and Intelligent Systems, Phoenix, Arizona, USA.

Losee, R. M. (2006). Browsing mixed structured and unstructured data. *Information Processing & Management, 42*(2), 440-452.

Lv, Y., Ma, T., Tang, M., Cao, J., Tian, Y., Al-Dhelaan, A., & Al-Rodhaan, M. (2016). An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing, 171*, 9-22.

MacQueen, J. (1967). *Some methods for classification and analysis of multivariate observations.* Paper presented at the Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability.

Makul, Ö., & Ekinci, M. (2017, May 15-18). *A graph form data stream clustering approach based on dimension reduction.* Paper presented at the 25th International Conference on Signal Processing and Communications Applications, Antalya,Turkey.

Malsiner, W. G., Frühwirth, S. S., & Grün, B. (2016). Model-based clustering based on sparse finite Gaussian mixtures. *Statistics and Computing, 26*(1), 303-324.

Mansalis, S., Ntoutsi, E., Pelekis, N., & Theodoridis, Y. (2018). An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 11*(4), 167-187.

Maulik, U., & Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(12), 1650-1654.

McParland, D., & Gormley, I. C. (2016). Model based clustering for mixed data: clustMD. *Advances in Data Analysis and Classification, 10*(2), 155-169.

Mohana, N. C., Rao, H. Y., Rakshith, D., Mithun, P., Nuthan, B., & Satish, S. (2018). Omics based approach for biodiscovery of microbial natural products in

antibiotic resistance era. *Journal of Genetic Engineering and Biotechnology, 16*(1), 1-8.

Monath, N., Zaheer, M., Silva, D., McCallum, A., & Ahmed, A. (2019). *Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space*. Paper presented at the Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA.

Newton, I. (1729). In [experimental] philosophy particular propositions are inferred from the phenomena and afterwards rendered general by induction":" Principia (Vol. 3): General Scholium.

Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge & Data Engineering*(5), 1003-1016.

Nguyen, H. L., Woon, Y. K., & Ng, W. K. (2015). A survey on data stream clustering and classification. *Knowledge and Information Systems, 45*(3), 535-569.

Ntoutsi, I., Zimek, A., Palpanas, T., Kröger, P., & Kriegel, H. P. (2012, April 26-28). *Density-based projected clustering over high dimensional data streams.* Paper presented at the 12th SIAM International Conference on Data Mining, Anaheim, California, USA.

Olukanmi, P. O., & Twala, B. (2017). *Sensitivity analysis of an outlier-aware k-means clustering algorithm.* Paper presented at the 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics.

Oussous, A., Benjelloun, F. Z., Lahcen, A. A., & Belfkih, S. (2018). Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences, 30*(4), 431-448.

Park, H. S., & Jun, C. H. (2009). A simple and fast algorithm for K-medoids clustering. *Expert systems with applications, 36*(2), 3336-3341.

Park, N. H., & Lee, W. S. (2004). Statistical grid-based clustering over data streams. *ACM SIGMOD Record, 33*(1), 32-37.

Pilevar, A. H., & Sukumar, M. (2005). GCHL: A grid-clustering algorithm for high-dimensional very large spatial data bases. *Pattern Recognition Letters, 26*(7), 999-1010.

Puschmann, D., Barnaghi, P., & Tafazolli, R. (2017). Adaptive clustering for dynamic IoT data streams. *IEEE Internet of Things Journal, 4*(1), 64-74.

Ramírez, G. S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing, 239*, 39-57.

Rasmussen, C. E. (2000). *The infinite Gaussian mixture model.* Paper presented at the Advances in Neural Information Processing Systems.

Reddy, K. S. S., & Bindu, C. S. (2017, February 10-11). *A review on density-based clustering algorithms for big data analysis.* Paper presented at the IEEE International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), Palladam, India.

Ren, J., Cai, B., & Hu, C. (2011). Clustering over data streams based on grid density and index tree. *Journal of Convergence Information Technology, 6*(1).

Ren, J., & Ma, R. (2009). *Density-based data streams clustering over sliding windows.* Paper presented at the 6th International Conference on Fuzzy Systems and Knowledge Discovery.

Riyadh, M., Mustapha, N., Sulaiman, M., & Sharef, N. B. M. (2017). CC_TRS: Continuous Clustering of Trajectory Stream Data Based on Micro Cluster Life. *Mathematical Problems in Engineering, 2017*.

Ruiz, C., Menasalvas, E., & Spiliopoulou, M. (2009, October 3-5). *C-denstream: Using domain knowledge on a data stream.* Paper presented at the 12th International Conference on Discovery Science, Porto, Portugal.

Ruiz, C., Spiliopoulou, M., & Menasalvas, E. (2007, October 11-14). *C-DBSCAN: Density-based clustering with constraints.* Paper presented at the International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing, Halifax, NS, Canada.

Sardar, T. H., & Ansari, Z. (2018). Partition based clustering of large datasets using MapReduce framework: An analysis of recent themes and directions. *Future Computing and Informatics Journal, 3*(2), 247-261.

Shao, J., Tan, Y., Gao, L., Yang, Q., Plant, C., & Assent, I. (2018). Synchronization-based clustering on evolving data stream. *Information Sciences, 501*, 573-587.

Sharma, P., & Sharma, A. (2017, 6-7 July 2017). *Online K-means clustering with adaptive dual cost functions.* Paper presented at the 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., De Carvalho, A. C., & Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys, 46*(1), 13.

Statista. (2018, May 2018). Information created globally 2005-2025. Retrieved October 16, 2018, from https://www.statista.com/statistics/871513/worldwide-data-created/

Steinhaus, H. (1999). *Mathematical snapshots* (3rd ed.). New York: Dover: Courier Corporation.

Sumathi, S., & Sivanandam, S. (2006). Data mining tasks, techniques, and applications. *Introduction to Data Mining and its Applications, 29*, 195-216.

Tøndel, K., Vik, J. O., Martens, H., Indahl, U. G., Smith, N., & Omholt, S. W. (2013). Hierarchical multivariate regression-based sensitivity analysis reveals complex parameter interaction patterns in dynamic models. *Chemometrics and Intelligent Laboratory Systems, 120*, 25-41.

Vallim, R. M., José Filho, A. A., de Mello, R. F., de Carvalho, A. C., & Gama, J. (2014). Unsupervised density-based behavior change detection in data streams. *Intelligent Data Analysis, 18*(2), 181-201.

W.M. Ma, E., & Chow, T. W. S. (2004). A new shifting grid clustering algorithm. *Pattern Recognition, 37*(3), 503-514.

Wan, L., Ng, W. K., Dang, X. H., Yu, P. S., & Zhang, K. (2009). Density-based clustering of data streams at multiple resolutions. *ACM Transactions on Knowledge discovery from Data, 3*(3), 14.

Wang, W., & Vrbanek, J. (2008). An evolving fuzzy predictor for industrial applications. *IEEE Transactions on Fuzzy Systems, 16*(6), 1439.

Wang, Z., Yu, Z., Chen, C. P., You, J., Gu, T., Wong, H. S., & Zhang, J. (2018). Clustering by local gravitation. *IEEE Transactions on Cybernetics, 48*(5), 1383-1396.

Win, K. N., Chen, J., Chen, Y., & Fournier-Viger, P. (2019). PCPD: A Parallel Crime Pattern Discovery System for Large-Scale Spatiotemporal Data Based on Fuzzy Clustering. *International Journal of Fuzzy Systems, 21*(6), 1961-1974.

Wu, B., & Wilamowski, B. M. (2017). A fast density and grid based clustering method for data with arbitrary shapes and noise. *IEEE Transactions on Industrial Informatics, 13*(4), 1620-1628.

Wu, X., Zhu, X., Wu, G., & Ding, W. (2014). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering, 26*(1), 97-107.

Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science, 2*(2), 165-193.

Xu, L., Jiang, C., Wang, J., Yuan, J., & Ren, Y. (2014). Information security in big data: Privacy and data mining. *IEEE Access, 2*, 1149-1176.

Yang, C., & Zhou, J. (2006, December 18-22). *Hclustream: A novel approach for clustering evolving heterogeneous data stream.* Paper presented at the 6th IEEE International Conference on Data Mining Workshops, Hong Kong, China.

Yang, Y., Liu, Z., Zhang, J. P., & Yang, J. (2012, May 29-31). *Dynamic density-based clustering algorithm over uncertain data streams.* Paper presented at the 9th International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China.

Yu, Y., Wang, Q., & Wang, X. (2013). Continuous clustering trajectory stream of moving objects. *China Communications, 10*(9), 120-129.

Zahn, C. T. (1970). Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Transaction on Computers, 20*, 68.

Zhan, Y., Tan, K. H., & Huo, B. (2019). Bridging customer knowledge to innovative product development: a data mining approach. *International Journal of Production Research*, 1-16.

Zhang, J. P., Chen, F. C., Liu, L. X., & Li, S. M. (2013, May 23-25). *Online stream clustering using density and affinity propagation algorithm.* Paper presented at the 4th IEEE International Conference on Software Engineering and Service Science, Beijing, China.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). *BIRCH: An efficient data clustering method for very large databases.* Paper presented at the 1996 ACM SIGMOD International Conference on Management of Data.

Zhang, X., Furtlehner, C., Perez, J., Germain, R. C., & Sebag, M. (2009, June 28-30). *Toward autonomic grids: analyzing the job flow with affinity streaming.* Paper presented at the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France.

Zhang, X., & Xu, Z. (2015). Hesitant fuzzy agglomerative hierarchical clustering algorithms. *International Journal of Systems Science, 46*(3), 562-576.

Zhao, Y., & Song, J. (2001, 29 Oct.-1 Nov. 2001). *GDILC: A grid-based density-isoline clustering algorithm.* Paper presented at the 2001 International Conferences on Info-Tech and Info-Net.

APPENDIX A
ACHIEVEMENTS

Journal Papers

i. Islam, M. K., Ahmed, M. M., Zamli, K. Z. (2019). A buffer-based online clustering for evolving data stream. *Information Sciences*, 489, 113-135. **(ISI Q1, IF:5.524)**

ii. Islam, M. K., Ahmed, M. M., Zamli, K. Z.. i-CODAS: An improved online data stream clustering into arbitrary shaped clusters. *Engineering Letters*. **(SJR 0.3, Accepted, 2019)**

iii. Islam, M. K., Ahmed, M. M., Zamli, K. Z., Mehbub S.. Online tweet stream processing to predict civil unrest based on recursive weight, event diffusion and location graph. *Journal of Information and Communication Technology*. **(SJR 0.22, Accepted, 2019)**

Conference Paper

i. Islam, M. K., Ahmed, M. M., Zamli, K. Z. (December 14-15, 2018). Identifying the pornographic video on *YouTube* using vlog stream. Paper presented at the 4th IEEE International Conference on Computing Communication and Automation, India.