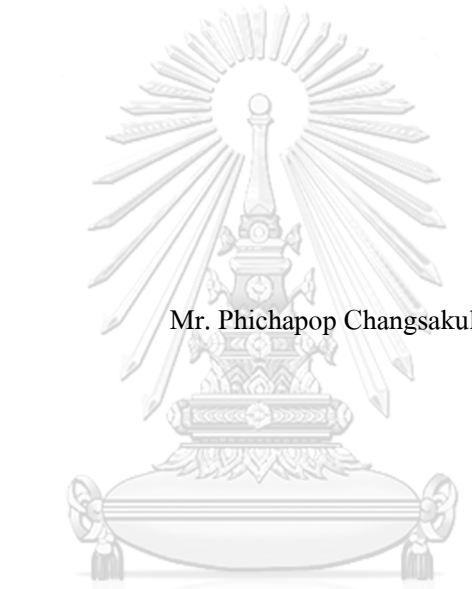
# PARAMETER-FREE OUTLIER SCORING USING MASS RATIO VARIANCE FOR STATIC AND STREAMING DATA

Mr. Phichapop Changsakul





A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Computer Science and Information Technology

Department of Mathematics and Computer Science

FACULTY OF SCIENCE

Chulalongkorn University

Academic Year 2021

# การให้คะแนนจุดผิดปกติไร้พารามิเตอร์โดยใช้ความแปรปรวนของอัตราส่วนมวลสำหรับข้อมูลสถิตและสตรีมมิ่ง

นายพิชาภพ แจ้งสกุล



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

| | |
|---|---|
| Thesis Title | PARAMETER-FREE OUTLIER SCORING USING MASS RATIO VARIANCE FOR STATIC AND STREAMING DATA |
| By | Mr. Phichapop Changsakul |
| Field of Study | Computer Science and Information Technology |
| Thesis Advisor | Assistant Professor SOMJAI BOONSIRI, Ph.D. |
| Thesis Co Advisor | Associate Professor KRUNG SINAPIROMSARAN, Ph.D. |

Accepted by the FACULTY OF SCIENCE, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

..... Dean of the FACULTY OF SCIENCE

(Professor POLKIT SANGVANICH, Ph.D.)

THESIS COMMITTEE

..... Chairman

(Assistant Professor ARTHORN LUANGSODSAI, Ph.D.)

..... Thesis Advisor

(Assistant Professor SOMJAI BOONSIRI, Ph.D.)

..... Thesis Co-Advisor

(Associate Professor KRUNG SINAPIROMSARAN, Ph.D.)

..... External Examiner

(Assistant Professor Chumphol Bunkhumpornpat, Ph.D.)

พิชาภพ แจ้งสกุล : การให้คะแนนจุดผิดปกติไร้พารามิเตอร์โดยใช้ความแปรปรวนของอัตราส่วนมวลสำหรับข้อมูลสถิตและสตรีมมิ่ง. ( PARAMETER-FREE OUTLIER SCORING USING MASS RATIO VARIANCE FOR STATIC AND STREAMING DATA) อ.ที่ปรึกษาหลัก : ผศ. ดร.สมใจ บุญศิริ, อ.ที่ปรึกษาร่วม : รศ.ดร.กรุง สินอภิรมย์สราญ

*การตรวจจับจุดข้อมูลผิดปกติเป็นปัญหาสำคัญที่ได้รับการศึกษาวิจัยและการประยุกต์จริงกับข้อมูล อย่างไรก็ตามมีการวิจัยเพียงเล็กน้อยเกี่ยวกับการให้คะแนนค่าความผิดปกติที่ปราศจากพารามิเตอร์แบบไม่มีผู้สอน วิทยานิพนธ์นี้เสนอค่าปัจจัยความผิดปกติของความแปรปรวนอัตราส่วนมวลหรือเอ็มโอเอฟ ซึ่งเป็นการให้คะแนนค่าความผิดปกติที่ปราศจากพารามิเตอร์แบบไม่มีผู้สอนกับข้อมูลสถิต ขั้นตอนวิธีนี้คำนวณคะแนนความผิดปกติตามความแปรปรวนของอัตราส่วนของมวล จุดข้อมูลที่มีคะแนนผิดปกติสูงจะสัมพันธ์กับจุดผิดปกติ ในขณะที่จุดข้อมูลที่มีคะแนนผิดปกติต่ำจะสัมพันธ์กับจุดปกติ วิทยานิพนธ์นี้ยังเสนอขั้นตอนวิธีปัจจัยความผิดปกติของความแปรปรวนอัตราส่วนมวลในข้อมูลสตรีมมิ่งหรือเอสเอ็มโอเอฟ ขั้นตอนวิธีนี้จะคำนวณคะแนนความผิดปกติอิงตามเอ็มโอเอฟและแบบจำลองหน้าต่างบานเลื่อนที่ไม่ทับซ้อนกัน ซึ่งเก็บกลุ่มข้อมูลหนาแน่นโดยวิธีการสุ่มตัวอย่างตามน้ำหนัก ทำให้วิธีการจัดเก็บข้อมูลมีประสิทธิภาพสูง วิทยานิพนธ์นี้ได้ทำการทดลองที่ครอบคลุม หลากหลายกรณีเพื่อประเมินประสิทธิภาพของเอ็มโอเอฟและเอสเอ็มโอเอฟโดยใช้ชุดข้อมูลที่สังเคราะห์และใช้งานจริง ผลการทดลองแสดงให้เห็นว่าวิธีดังกล่าวมีความแม่นยำมากกว่าเทคนิคการตรวจจับจุดข้อมูลผิดปกติที่ล้ำสมัย*



| | | |
|---|---|---|
| สาขาวิชา | วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ | ลายมือชื่อนิสิต ..... |
| ปีการศึกษา | 2564 | ลายมือชื่อ อ.ที่ปรึกษาหลัก ..... |
| | | ลายมือชื่อ อ.ที่ปรึกษาร่วม ..... |

## 6278507823 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

KEYWORD: Outlier scoring algorithm, Parameter-free outlier scoring algorithm, Mass-ratio variance, Anomaly in static and streaming data

Phichapop Changsakul : PARAMETER-FREE OUTLIER SCORING USING MASS RATIO VARIANCE FOR STATIC AND STREAMING DATA. Advisor: Asst. Prof. SOMJAI BOONSIRI, Ph.D. Co-advisor: Assoc. Prof. KRUNG SINAPIROMSARAN, Ph.D.

Outlier detection is a significant problem that has been studied in a variety of research and real-world applications. However, little research has been conducted on unsupervised parameter-free outlier scoring. This thesis proposes Mass ratio variance-based Outlier Factor, or MOF, which is unsupervised parameter-free outlier scoring for static data. This algorithm calculates outlier scores based on the variance of mass ratio. The data points with high outlier scores are associated with outliers while the data points with low outlier scores are associated with normal data points. This thesis also proposes Streaming Mass ratio variance-based Outlier Factor or SMOF. This algorithm calculates outlier scores based on MOF and the non-overlapping sliding window model which keeps the dense data points by weighted random sampling making highly efficient storage. Extensive experiments have been conducted to evaluate the performance of MOF and SMOF using synthesized and real-world data sets. The experimental results show that they have better accuracy than the state-of-the-art outlier detection techniques.



| | | |
|---|---|---|
| Field of Study: | Computer Science and Information Technology | Student's Signature ..... |
| Academic Year: | 2021 | Advisor's Signature ..... |
| | | Co-advisor's Signature ..... |

# ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Assistant Professor Somjai Boonsiri, who helped complete my research paper and also corrected the grammar of my thesis and suggestions. I would also like to thank Associate Professor Kung Sinapiromsarn for his advice, comments, and invaluable suggestions. Finally, I would like to sincerely thank my parents for their funds and mental support in my study for this master's degree.

Phichapop Changsakul

# TABLE OF CONTENTS

**Page**

# LIST OF TABLES

**Page**

# LIST OF FIGURES

**Page**

# Chapter 1
# Introduction

This chapter covers the basic outlier detection for both static and streaming data. In section 1.1, the basic outlier detection is presented. In section 1.2, the proximity-based outlier detection is stated. The parameter-free outlier detection is introduced in section 1.3. Section 1.4 discusses the basic outlier detection in streaming data. Section 1.5 states the aims and objectives of this thesis.

## 1.1 The basic outlier detection

An outlier is a data point that differs significantly from the rest of the data points in a data set [1]. It is sometimes called an abnormal or an anomaly. Generally, a normal data point is generated by a normal process, while an outlier is generated from a different process that rarely occurs. The outlier pattern provides valuable insight into an unusual event of a problem [2] as demonstrated by the following examples.

- In an intrusion detection system, data gathered from network logs may record unusual behavior due to malicious activities [3]. These patterns of logs will constitute an intrusion of the system.
- In credit card fraud, an unauthorized credit card use may exhibit an interesting pattern that can be classified as outliers among credit card transaction data [4].
- In a sensor-related event, sensors are often used to detect a change in an environment. Sudden changes in an underlying pattern may represent an anomaly, which is an interesting event [5].
- In a medical diagnosis, Magnetic Resonance Imaging scans (MRI) [6] recording anomaly patterns often reflect the disease state of a patient.

Other applications of outliers can arise for a variety of causes, such as an instrumental error, a setup error, a human error, or a catastrophe. Regardless of the reason behind outliers, they may be interesting to users because they carry some different information from normal data points. Some analysts define outliers as problems, and some analysts define them as interesting items, but they are unavoidable [7]. In brief, outliers are interesting and take different forms in different types of applications. V. Hodge and J. Austin [8] classified outliers into three major types as follows:

1. A point outlier (type I) is a data point that is placed very far from a region of normal data points, assuming all data points are mapped into an $n$-dimensional Euclidean space. This is the simplest type of outlier and very easy to identify on a static data set based on the distance metric. Intuitively, each attribute from a data

set will be mapped to a single dimension in the Euclidean space. For example, Figure 1.1 shows data points in 2-dimensional Euclidean space. Data point $O_1$ is far from cluster $C_1$, so it will be treated as a point outlier.

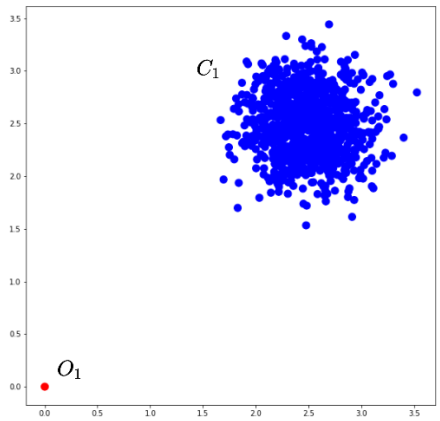



Figure 1.1 An example of a point outlier in a 2d synthesized data set

2. A contextual outlier (type II) is isolated from other data points with respect to the context. They can be considered as the context outlier. This type of outlier requires a contextual attribute such as time. An outlier differs from the surrounding data points according to the contextual attribute. Note that a point outlier is a contextual outlier, but a contextual outlier may not be a point outlier even if the contextual attribute is dropped. For example, Figure 1.2 shows an area's monthly temperature time series over the last few years. The contextual attribute is the time dimension. Note that $t_1$ and $t_2$ are both low temperatures. However, $t_2$ is the temperature during the winter season, so it is treated as a normal data point while $t_1$ is the low temperature during the summer season. Hence, it is treated as a contextual outlier.

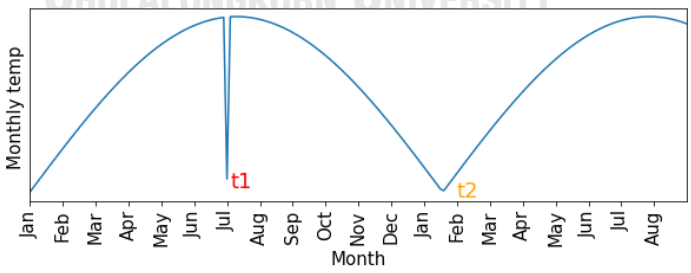



Figure 1.2 Temperature record by month

3. A collective outlier (type III) is a subset of data points treated as a single collection that deviates significantly from the rest of the data points in a data set. An individual data point may not be defined as a collective outlier by itself, but a collection of adjacent data points can be grouped as a collective outlier. In the time-series data, where the entire data set forms a sequence of data points, a

particular subsequence is a collective outlier if they exhibit different patterns with respect to the entire sequence. For example, Figure 1.3 shows an example of a patient's electrocardiogram [9] with a collective outlier highlighted with the red color due to consecutive low values corresponding to an Atrial Premature Contraction.

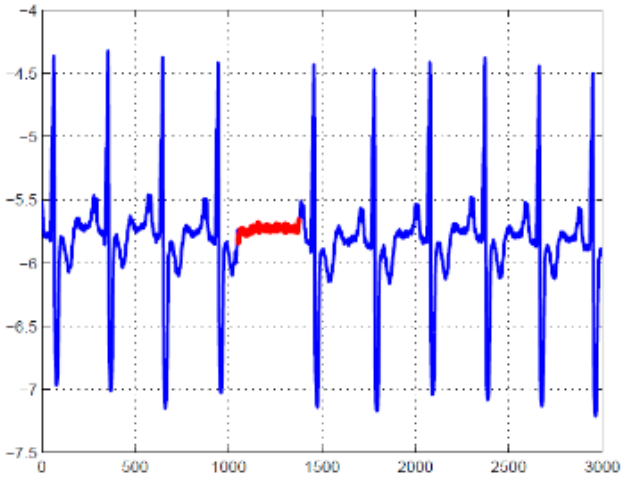



Figure 1.3 An example of a patient's electrocardiogram

Definitions of an anomaly and an outlier are different in the view of Aggarwal et al. [2]. He described an outlier as a data point that can be regarded as an irregular point, a noise, or an abnormal data point. In contrast, an anomaly is considered a particular type of outlier that needs attention from a specialist. Nevertheless, this thesis will concentrate on detection, so both keywords will not be treated differently.

Outlier identification is a challenging problem for a static data set. Because the notion of outliers is ambiguous and application-specific, no formal definition of outliers has been defined. Despite this fact, numerous publications perform outlier detection based on a user-specific criterion. They may be classified into supervised, semi-supervised, or unsupervised methodologies depending on the learning method.

1. A supervised methodology assumes the availability of class labels from a training data set for both normal data points and outliers. Researchers built a predictive model for the anomaly class against the normal type. Then, an outlier is predicted as a data point in the anomaly class.
2. A semi-supervised methodology assumes that the training data set contains only the normal data points. Since the label is not provided for outliers, this method is widely applicable to more general problems. Many unsupervised techniques can be adapted to operate in a semi-supervised mode using only normal data points as training. Such adaptation assumes that the test data contains outliers that cannot be captured by the unsupervised model during training. This method is sometimes called novelty detection [9].

3. An unsupervised methodology does not require a training label. The techniques in this category make an implicit assumption that a group of normal data points forms a cluster, while an outlier will not belong to any cluster of normal data points. If this assumption is not true, then such techniques suffer from a high false alarm rate.

Additionally, the output characteristic of outlier detection can be used to categorize the outlier detection method into one of the following two method types.

1. Outlier scoring: an outlier detection method generates a score of each data point as the degree of being an outlier. This score can also be used to rank data points based on their outlier tendency.
2. Outlier labeling: an output of an outlier detection method is a binary label that is true for an outlier and is false otherwise. This output can be generated from the outlier scoring algorithm by setting up a threshold.

Normally, an outlier detection constructs a model of the normal patterns and then computes outlier scores or labels of data points that deviate from these patterns. Examples of these models are generative models such as a Gaussian-mixture model, a regression-based model, or a proximity-based model. They use different assumptions for the "normal" behavior of data. This thesis will focus only on the proximity-based model.

## 1.2 The proximity-based outlier detection

The idea of outliers in the proximity-based outlier detection is that data points are isolated from the remaining data points based on similarity or distance functions such as Makowski distance ($L_p$-norm distance), Euclidean distance, Manhattan distance, Jaccard similarity, or Cosine similarity. A proximity-based method is among the most popular methods used in an outlier analysis. It may be applied in one of three methods which are a clustering-based method, a distance-based method, or a density-based method.

A clustering-based method is a method that groups data points having similar characteristics of a data set as a cluster. There are three main approaches in this direction. First, any normal data point must belong to a cluster, while outliers do not belong to any cluster such as DBSCAN [10] and SNN clustering [11]. Their algorithms concentrate on grouping data in a cluster, but they did not focus on identifying outliers. Second, a normal data point belongs to the cluster centroid, while anomalies are far from all cluster centroids. It consists of two steps. The first step is to find $k$-centroids via a clustering algorithm, and then the second step is to calculate the distance of each data point to its nearest centroid as the score. These types of algorithms are Self-Organizing Maps (SOM) [12], $k$-means Clustering [13], and Expectation-Maximization (EM) [14], which are studied by Smith et al. [15]. Third, a normal data point belongs to large and dense clusters, while outliers belong to small or sparse clusters. Techniques based on this assumption classify data points belonging

to a cluster by a cluster size or cluster density. If the size or the density of a cluster is less than a certain threshold, then this cluster contains anomalies. An algorithm of this type is the Cluster-Based Local Outlier Factor (CBLOF) [16]. The CBLOF score captures the size of the cluster to which a data point belongs, as well as the distance from the cluster centroid.

A distance-based method defines outlier scores based on the basis of the distance from other data points. The simplest example is the case in which the $k$ nearest neighbors distance of a point is used as its outlier score. It is the so-called $k$ Nearest Neighbors outlier detection (kNN) [17-19]. This algorithm uses mean, max, or median of $the\ k$-nearest neighbors distance as the outlier scores. kNN has been used to detect credit fraud by peer group analysis [20]. However, it did not work well if a data set contained clusters of various densities.

A density-based method defines an outlier score based on the number of data points within a specific region to estimate the density of the neighborhood of each data point. A data point that lies in a neighborhood of low density is declared to be an outlier, while a data point that lies in a dense neighborhood is declared to be normal. Considering the example data set in Figure 1.4 from [21], it shows that the data set contains two outliers ($O_1$ and $O_2$), and two clusters, one of which is much sparser than the other. Data point $O_2$ cannot be detected as an outlier by a distance-based algorithm unless a smaller distance threshold is used. LOF [21] can identify $O_2$ in a locally adjusted manner.

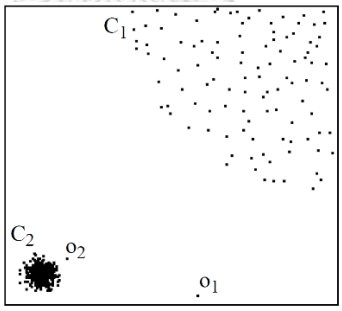



Figure 1.4 An example of a 2d synthesized data set

A proximity-based method is naturally designed to detect both noise and anomalies, although different methods are suitable for different kinds of outliers. For example, some clustering methods are designed to detect noise that does not belong to a cluster, whereas some methods detect large levels of deviation or sparsity in terms of density-based or distance-based as outliers or anomalies.

These methods are extremely popular because of their intuitive simplicity and interpretability. In fact, several methods for intuitive exploration and explanation of outliers [20] are based on proximity. Because of the simplicity of the underlying

methods, they can be easily generalized to any data type, such as time-series data, sequence data, or graph data.

## 1.3 The parameter-free outlier detection

Outlier detection of the proximity-based method depends on one or more user-defined parameters. For example, kNN is based on its $k$-distance, where $k$ is a user-defined parameter. It is common to vary a user-defined parameter of the algorithm in order to search for the best parameter of the current data set via AUC metric. There are two problematic issues with this search. First, suitable parameters cannot be identified if there is no outlier label for any data point. Second, even if there are some outlier labels, such evaluation will favor normal data points [2] over other outliers. Therefore, it is necessary to develop parameter-free algorithms or algorithms that have a small number of parameters for better predictive performance across diverse data sets [21].

Various outlier scoring algorithms have been proposed in the literature [22-24] based on different metrics. An improved parameter-free outlier scoring algorithm based on the distance metric and angle is proposed [25] that avoids the weakness from [22-24]. However, using only the distance metric will give low scores to outliers forming a small clump. The density concept is more appropriate.

## 1.4 The basic outlier detection in streaming data

In the new information era, the data assimilation process has changed significantly from humans to devices such as the sensor network, web-click monitoring, and network traffic monitoring into the form of streaming data. Streaming data is a continuous, unbounded sequence of data records accompanied and ordered by implicit or explicit timestamps. The arrival interval is usually very short, and the data distribution often changes over time. The streaming data concept started to gain popularity in the early 2000s; since then, numerous efforts have been put forward to study them from different perspectives, such as query processing, resource management, storage management, outlier detection, clustering, and novel detection.

Outlier detection for streaming data is a new area of research compared to a long history of outlier detection in static data [26]. In the static data, an outlier detection technique extracts a pattern from a data set and then compares every data point to the pattern to detect outliers (store-and-process paradigm). However, the store-and-process paradigm is not applicable for streaming data because an entire data set is never available due to its unbounded nature; learning any trend without store-and-process is a challenging issue and requires new research. In addition, due to the characteristics specific to streaming data, such as short interarrival rates, timelines,

and change of data distribution (or concept drift), they introduce new issues for outlier detection techniques. The issues related to streaming data characteristics are [27]:

1. Transient: any data point in streaming data is transient in nature. After a data point has been introduced from a long past, it will lose its importance. For streaming data, an outlier detection technique should detect the outlier score of data points immediately as it arrives [30].
2. Infinite: streaming data is an infinite sequence of data points as they are generated from a data source indefinitely. Therefore, at any specific time, the entire data set will not be available, so many static outlier detection techniques requiring access to the whole data set cannot be used. An outlier detection method for streaming data normally stores a summary of a data set that performs computation incrementally. The data point is compared to the summary instead of all previous data points. Thus, an outlier detection model must be incremental and cannot assume the availability of the entire data set.
3. Arrival rate: the arrival rate may be fixed or varied. An outlier detection technique for streaming data must process the current data point before the next available data point arrives. The set of data points or the summary of the data points to which the current data point is compared to detect outliers should be adjusted based on the available processing time. [27]
4. Concept drift: the distribution of data may change over time in streaming data, which is defined as concept drift. A data point that has been detected as an outlier for one data distribution may change its outliers for different data distributions. Therefore, the outlier detection technique for streaming data cannot assume any fixed distribution of data. [28]

Streaming data has a single temporal context for data points, so the outlier should be a contextual outlier. The streaming data is considered as infinite sequences of numbers, and the processing process must be online; therefore, at any moment, only a subset of the entire data set is presented [27].



To address the growing need for outlier detection for streaming data, researchers mostly used distance-based outlier detection in streaming environments [29-32]; the literature normally ignores the findings of local outliers in streaming data [32]. The local outlier factor (LOF) detection technique [7] and many of its varieties have been proposed for solving static data sets [33-35]. In terms of detecting local outliers in streaming data, researchers in [36] proposed incremental LOF (iLOF). However, this technique needs all previous data points to detect outliers precisely. In the case of streaming data, where the number of data points is unbounded and can arrive at a high rate, keeping all data points is impossible. Hence, such an approach has limited use in practice.

MiLOF [37] was proposed to solve the memory limitation by clustering past data points and storing only a small number of data points in memory. Due to the use of the $k$-mean algorithm, which does not preserve data density, this approach

invariably leads to a significant reduction in outlier detection accuracy. Another issue with existing LOF-based algorithms is that they are incapable of detecting outlier sequences in streaming data. The DiLOF [38] algorithm was proposed to address the limitations of the iLOF and MiLOF algorithms. The DiLOF algorithm is divided into two steps: the detection and the summarization steps. When new data points are inserted, the detection step is used to detect outliers and update the data points in the window. In this step, the iLOF algorithm is used to detect outliers, and a skipping scheme is used to detect a sequence of outliers, as in DoS attacks. In the summarization step, data points in a window are summarized to half (NDS). However, all of these are based on the LOF algorithm, which requires a sensitive parameter setting.

To avoid distance measure, the Streaming HS-Trees [38] algorithm, is a one-class outlier detection algorithm, which uses a mass profile or the number of data points in a full binary tree to measure an outlier score. The system processes two windows at any time which are the reference window and the current window. During the initial outlier detection stage, the algorithm extracts the mass profile of data in the reference window to measure outlier scores of new data points. A new data point that falls in the high-mass area is assigned as a normal data point, whereas a data point that falls in the low-mass or empty area is interpreted as an outlier; after that, a new data point is added to the current window and changes the mass profile. When the current window is full, the newly recorded profile takes precedence over the previous profile in the reference window, ensuring that the reference window always contains the most recent profile, which can be used to score the next batch of newly arriving data. The current window will then delete its previously saved profile and prepare to capture the profile of the next batch of newly arrived data. This process will continue for the next streaming data. However, this algorithm does not collect any normal data points to measure outlier scores. It assigns an outlier score to data points in the next batch and uses data points in the next batch to measure outlier scores after this batch.



## 1.5 Aims and objectives

Outliers in data need to be detected automatically in real-time. The popular learning algorithms for outlier detection use unsupervised learning and generate scores instead of outlier labels. This will give control to the user to set the threshold for detecting outliers. With intuitive simplicity and interpretability, a proximity-based model is normally applied to detect anomalies in a static data set. However, most of these algorithms need to set the number of nearest neighbors. To avoid setting this parameter, different distance-based methods are used. Nevertheless, current results were reported with lower performance than the one with the best parameter. Another approach is to apply the angle concept to this problem. Due to its high complexity, it is hardly used in real-world data sets.

Outlier detection in streaming data differs from one in static data due to streaming data characteristics which are transient, infinite, high arrival rate, and

concept drift. The outlier detection in static data that stores all data points before processing cannot be used. A batch of new data points should be evaluated before the next batch of data points appears. An algorithm should construct a model of normal data points to help detect future outlier patterns.

This thesis proposed two outlier scoring algorithms for static data and streaming data. The first algorithm is called the Mass ratio variance-based Outlier Factor (MOF) algorithm to generate outlier scores of data points in a static data set. The second algorithm is called the Streaming Mass ratio variance-based Outlier Factor (SMOF) algorithm to generate outlier scores in the next batch of streaming data.

The MOF algorithm is a parameter-free unsupervised outlier score that employs the proximity concept. The MOF algorithm assigns MOF scores to each data point in a data set, indicating its degree of outlier-ness. The MOF algorithm in this thesis assumes all data points lie in the Euclidean space, having low density points surrounding them compared to the rest of the data points, so the MOF algorithm collects a ratio of density surrounding a pair of data points with the sphere having the radius equal to the distance between a pair of data points. These ratios give the density of the current data point with the rest of the data points. An outlier will have a large variance from this collection.

This thesis also proposed an outlier scoring algorithm for streaming data called the SMOF algorithm. It addresses the special characteristics of streaming data to be applied to the MOF algorithm via the sliding window concept and weighted random sampling. The SMOF algorithm assigns SMOF scores to each data point in streaming data, indicating its degree of outlier-ness. The SMOF algorithm manages two streaming data windows which are the reference window and the current window. The reference window or synopsis window contains a summary of previous data points, capturing changes in streaming data. Initially, the MOF algorithm will be performed on this initial window to give outlier scores to all data points. The current window or a non-overlapping sliding window contains data points that currently arrive from streaming data. Each new data point will be assigned an outlier score, and these scores will be combined with data points in the reference window to generate a new synopsis window of the same size. Unlike other proximity-based outlier detections such as MiLOF, DiLOF, and HS-Trees, the SMOF algorithm does not construct a summary or synopsis via clusters or trees that requires multiple parameters. Instead, the SMOF algorithm uses weighted random sampling based on MOF scores to maintain outliers with low probability and keep normal data points with high probability.

This thesis tries to answer these questions: Is it possible to construct a density-based outlier scoring algorithm without any user-defined parameter? Could this method be extended to score data points in a data stream environment? In order to achieve these aims, the following objectives have been set.

1. Design a parameter-free scoring algorithm based on the density concept in static data.
2. Extend this algorithm to work with streaming data.
3. Synthesize various data sets to assess the performance of both algorithms based on known outliers in static data sets and streaming data sets.
4. Compare the performance of the MOF and the SMOF algorithms with known scoring algorithms using their best parameters.

To be able to assess the performance of the proposed methods compared with other methods, labeled data sets have been synthesized with different anomaly data patterns.

# Chapter 2
# Background

This chapter is organized as follows. In section 2.1, static and streaming data are defined. In section 2.2, machine learning algorithms using proximity-based are introduced. The window model is introduced in section 2.3. Section 2.4 discusses the Weighted Random Sampling (WRS).

## 2.1 Input Data

The nature of the input data is generally a collection of data points known as objects, records, vectors, events, cases, samples, observations, or entities. A set of attributes which is also referred to as a dimension or a feature can be used to describe each data point. In this thesis, the attributes are numerical, and each data point contains multiple attributes. Furthermore, input data can be classified as static and streaming data. The operational distinction between them lies in when the data is stored and analyzed. Before it is processed, static data is loaded into the main memory. On the other hand, streaming data is processed by loading one or two windows at a time.

### 2.1.1 Static data

Static data is defined as a fixed-size collection of data points forming a single data set. It normally assumes that the current computer memory can hold all data points during processing, and any data point can be recalled at any time. Static data is normally organized as a matrix of numeric numbers that will fit in the memory during the scoring process. Common static data refers to multidimensional data which is defined as follows:



**Definition 2.1** (Multidimensional data)

Multidimensional data, $D = \{X_1, \dots, X_n\}$, is a set of $n$ data points, where $X_i$ is the $i^{th}$ data point having $d$ numeric features denoted by $X_i = (x_i^1, \dots, x_i^d)$.

In static data, all data points are assumed to be collected independently at random. In practice, various data values may be related to one another in terms of time or explicit connections between data points. This can be defined as time-series data.

A time series is a collection of values that are typically generated over time by continuous measurement having equally spaced time intervals. Values at successive timestamps typically do not change abruptly. An abnormal event can be defined as a sudden change in consecutive values. As a result, finding outliers in time series is

closely linked to detecting anomalous events such as contextual or collective anomalies across related timestamps.

**Definition 2.2** (Multidimensional time-series data)

The time-series length $n$ containing $d$ numeric features is $\{Y_1, \ldots, Y_n\}$. Each timestamp contains a component having values of *d* attributes. So, the set of values received at the timestamp $i$ is $Y_i = (y_i^1, \ldots, y_i^d)$. The value of the $j^{th}$ attribute at the time stamp $i$ is $y_i^j$.

### 2.1.2 Streaming data

Streaming data is a potentially infinite sequence of data points arriving at a high rate on a regular basis which differs from time-series data. In the case of time series data, an individual data point shows a high correlation of temporal continuity, but multidimensional streaming data may not have this property. For example, in streaming data of texts, the current text record cannot be reliably predicted from its immediately preceding records. Because of the differences in the expected level of temporal continuity in this scenario, outlier analysis in multidimensional data streaming data differs significantly from outlier analysis in time-series data.

**Definition 2.3** (Multidimensional streaming data)

Multidimensional streaming data, $S = \{X_t\}_{t=1}^{n}$, is a stream of data points arrived at a timestamp *t* which is potentially unbound ($n \to \infty$). Each data point $X_t$ having $d$ features is denoted by $X_t = (x_t^1, \ldots, x_t^d)$.

## 2.2 Proximity-based outlier detection methods

Proximity-based outlier detection is simple to implement and makes no prior assumptions about the data distribution with the assumption that outliers are isolated from other data points. It uses the distance function to compute for each attribute and then combines them as a degree of isolation or an outlier score. In practice, different types of attributes are often encountered in the same data. A significant issue is that the range of the attributes may differ significantly. Scaling and normalization make each attribute equally important and make it easier to process with outlier detection. The details of distance function, scaling, and normalization are as follows:

### 2.2.1 Distance function

Most proximity-based outlier detection methods require a distance metric defined between two data points. The general distance metric for quantitative data is the L$p$-norm or Minkowski distance.

**Definition 2.4** ($L_p$ - norm between data points $X$ and $Y$)

Let data point $X = (x_1, \ldots, x_d)$, data point $Y = (y_1, \ldots y_d)$, and $p$ be a positive integer. $L_p$ - norm between two data points is defined as

$$d(X,Y) = \left( \sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

The Euclidean ($p$ = 2) and the Manhattan ($p$ = 1) distance metrics are two specific examples of the $L_p$ -norm. The sources for these metrics come from spatial applications with physical interpretability. The shortest straight-line distance between two data points is known as the Euclidean distance. The Manhattan distance is the "city block" driving distance with a rectangular grid of streets, such as New York City's Manhattan Island. Because the straight-line distance between two data points does not change with the orientation of the axis system, the Euclidean distance has the advantage of being rotation-invariant. It's worth noting that the data distribution, dimensionality, and data type are all highly sensitive to distance functions.

When calculating the distance between all pairs in a data set $D$, all distances are kept in a matrix form, which is a square matrix (two-dimensional array) containing the distances between row index $i$ corresponding to data point $i$ and column index $j$ corresponding to data point $j$ having the size $[n \times n]$.

**Definition 2.5** (Distance matrix of $D$)

Given data set $D = \{p_1, \ldots, p_n\}$. For row $i^{th}$, $Ad_i$ is defined as an array of a pairwise distance of data point $p_i$ to other data points in $D$. $DM$ is a distance matrix between pairs of data points defined as

$$Ad_i = \left(d_{i,j}\right)_{1 \times n} = \left(d(p_i, p_j)\right)_{1 \times n}$$

$$DM = \begin{bmatrix} Ad_i \\ \vdots \\ Ad_n \end{bmatrix}_{n \times n}$$

$$for\ i \in \{1, \ldots, n\}\ and\ j \in \{1, \ldots, n\}$$

Proximity-based methods use $DM$ for ordering and determining the neighbors of any data point. It is converted and stored in the sorted distance matrix and sorted distance index matrix. Both definitions are defined as follows:

**Definition 2.6** (Sorted distance matrix of $D$)

Given data set $D = \{p_1, \dots, p_n\}$ and the distance of $DM$ defines $d_{(i,c_k^i)}$ as the pairwise distance from a data point $p_i$ to its $k$ nearest neighbors (for $k = \{1,2, \dots, n\}$), and $Asd_i$ is an array of a sorted pairwise distance of data point $p_i$ to its $k$ nearest neighbors. $SDM$ is a sorted distance matrix of $D$. It is defined as follows:

$$Asd_i = sorted(Ad_i) = \left(d_{i,c_k^i}\right)_{1\times n}$$

$$SDM = \begin{bmatrix} Asd_i \\ \vdots \\ Asd_n \end{bmatrix}_{n\times n}$$

**Definition 2.7** (Sorted distance index matrix of $D$)

Given data set $D = \{p_1, \dots, p_n\}$ and $SDM$ defines $a_{i,k}$ as the index data point from a data point $p_i$ to its $k\ nearest\ neighbors$ Where $\{c_1^i, \dots, c_n^i\}$ is the permutation of $\{1, \dots, n\}$ that $d_{(i,c_1^i)} \le d_{\left(i,c_{k2}^i\right)} \le \cdots \le d_{(i,c_n^i)}$, $Aid_i$ is an array of an index of data point $p_i$ to its $k$ nearest neighbors. $ISM$ is the sorted distance index matrix of $SDM$. It is defined as follows:

$$a_{i,k} = c_k^i$$

$$Aid_i = argsort(Ad_i) = \left(a_{i,k}\right)_{1\times n}$$

$$SIM = \begin{bmatrix} Aid_i \\ \vdots \\ Aid_n \end{bmatrix}_{n\times n}$$



In a proximity-based method, most outlier detections require a distance function that will be used as an outlier factor. The following definitions will define the $k$-distance of data point $p$, $k$-distance neighborhood of a data point $p$, the following definitions are their notations.

**Definition 2.8** ($k$-distance of a data point $p$)

For any positive integer $k$, the $k\text{-}distance$ of data point $p$, denoted as $k\text{-}distance(p)$, is defined as the distance $d(p,o)$ between data points $p$ and $o \in D$ such that:

$$kE_o = \{o' \in D\backslash\{p\} | d(p,o') \le d(p,o)\}\ where\ |kE_o| \ge k$$

$$kL_o = \{o' \in D\backslash\{p\} | d(p,o') < d(p,o)\}\ where\ |kL_o| < k$$

**Definition 2.9** ($k$-distance neighborhood of data point $p$)

Given the $k$-distance of $p$, the $k$-distance neighborhood of $p$ contains every data point whose distance from $p$ is not greater than the $k\text{-}distance$. The data point $q$ in $\mathcal{N}_k(p)$ is called the $k$-nearest neighbors of $p$. It is defined by

$$\mathcal{N}_k(p) = \{ q \in D\backslash\{p\} \mid d(p,q) \leq k\text{-}distance(p) \}.$$

## 2.2.2 Scaling and normalization

In many scenarios, various features represent different scales of references, and they may not be comparable. For example, an age attribute is measured on a different scale than a salary attribute. The latter characteristic is usually several orders of magnitude greater than the former. As a result, any aggregate function based on various features (e.g., Euclidean distances) will be dominated by the attribute with a larger magnitude. Standardization or Z-score normalization are commonly used to solve this issue. Typically, standardization entails rescaling data to have a zero mean and a standard deviation of 1 (a unit variance) according to definition 2.12.

**Definition 2.10** (Mean)

Mean is defined as

$$\bar{X} = \frac{1}{n}\left(\sum_{i=1}^{n} X_i\right)$$

**Definition 2.11** (Standard deviation)

Standard deviation is defined as

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n}}$$



**Definition 2.12** (Standardization)

A data point is converted to a z-score via the following formula

$$\acute{X}_i = \frac{X_i - \bar{X}}{SD}$$

$where\ \acute{X}_i$ is the standardized value.

A second approach uses min-max scaling to map all values in numeric attributes to be in the range of $[0, 1]$ according to definition 2.13. This approach is not effective when the maximum and minimum values are extreme values or outliers. For example, if the data set has a single data point mistakenly recorded as 900 instead of 90. In this case, most normal age records will be in the range of $[0, 0.1]$. As a result, this attribute may be de-emphasized. Standardization is needed in such scenario.

**Definition 2.13** (Min-max scale)

Let $min(X)$ and $max(X)$ represent the minimum and maximum values of attributes $X_i$ in $D$.

$$\acute{X}_i = \frac{X_i - min\ (X)}{max(X) - min\ (X)}$$

$where\ \acute{X}_i$ is the min-max scale value.

# 2.3 Window models

When the streaming data is processed, it is more efficient to process recent data instead of the whole data. It is not possible to store the whole input data because streaming data is infinite, and all existing processing systems have main memory constraints. Different window models are developed for this purpose. There are three different window models, which are the landmark window model, the non-overlapping sliding window model, and the overlapping sliding window model [39].

## 2.3.1 The landmark window model

The landmark window model encompasses all data from a specific point in time up to the current moment. In the landmark window, the model is initialized in a fixed time point, the so-called landmark that marks the beginning of the window. In successive snapshots, the window size grows to consider all the data seen so far after the landmark. According to this definition, data points belonging to a window shown in Figure 2.1 are calculated using $W_T = (X_1, \ldots, X_{(T-1)A+w})$ where $w$ is initialized window size, $X_i$ is $i^{th}$ data point, and $W_T$ is $T^{th}$ window which indexes $T$ starting with one and add $O$ data points to a window at each step.

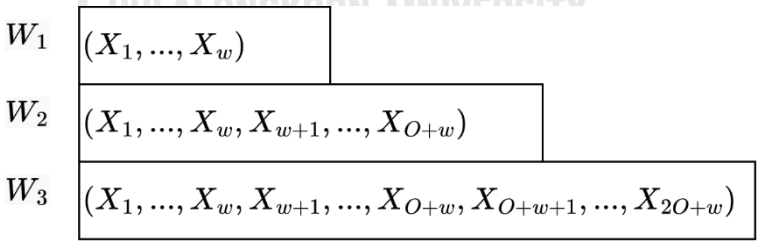



Figure 2.1 The landmark window model

## 2.3.2 The non-overlapping sliding window model

In the non-overlapping sliding window model, any data point belongs to a single window. The number of data points that belong to a single-window is called the window size and is usually indicated by $w$. The window size can be defined as a data point count or elapsed time. Consecutive windows do not intersect, and the new

window just begins from the data point the previous window ends. According to this definition, data points belonging to a window are shown in Figure 2.2 and denoted on $W_T = (X_{(T-1)w}, \dots, X_{Tw})$ where $w$ is the window size, $X_i$ is $i^{th}$ data point, and $W_T$ is $T^{th}$ window which indexes $T$ start with one.

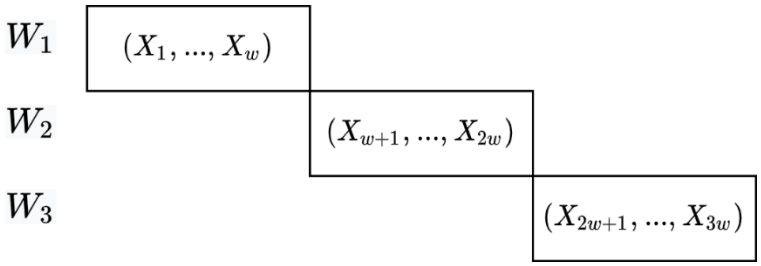



Figure 2.2 The non-overlapping sliding window model

### 2.3.3 The overlapping sliding window model

In the overlapping sliding window model, the window swaps data points at each step. The older data points move out of the window, and the most recent data points move into the window by the FIFO style. All data points in the window have equal weight, and consecutive windows mostly overlap. The window size is a user-parameter and should be decided according to the input data. Figure 2.3 shows that this window model and data points belonging to a window are calculated using $W_T = (X_{(T-1)A+1}, \dots, X_{(T-1)A+w})$ where $w$ is a window size, $X_i$ is $i^{th}$ data point, and $W_T$ is $T^{th}$ window which indexes $T$ start with one and add $O$ data points to a window at each step.

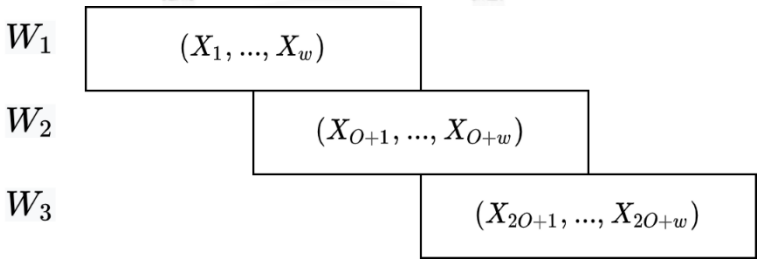



Figure 2.3 The overlapping sliding window model

## 2.4 Weighted Random Sampling (WRS)

Random sampling without replacement (RS) problem requires selecting *m* unique random data points from a data size *n*. All data points have the same probability of being chosen. In contrast, the data points are weighted in weighted random sampling (WRS) [40], and the probability of each data point being chosen is determined by its relative weight.

The size of a data set $D$ is $n$, the size of the random sample is $m$, and the weight of $X_i$ is $v_i$. The random function $(L, H)$ generates a uniform random number in (0, 1). All sampling problems are without replacement. Depending on the context, WRS is used to denote a weighted random sample. The pseudo code of the WRS algorithm is given as follows:

**Algorithm** $WRS\ (D, V, m)$

**Inputs:**

$D$ - data set, $X_i \in D$

$V$ – the weight of a data set, $v_i \in V$

$m$ - sample sizes

**Outputs:**

$WRS$ - sample data set, $WRS \subseteq D$

**Line:**

1. For $i \in \{1, 2, 3, \ldots n\}$
2. Generate: $U = \{ u_i \mid random(0,1) \}$
3. Calculate: $K = \{k_i | \ u_i^{\frac{1}{v_i}}\}$
4. Rank data points based on $K$ as $WRS$ and select the top $m$ data points
5. Return $WRS$

Figure 2.4 The pseudo code of the WRS algorithm

# Chapter 3
# Literature Surveys

This chapter presents existing outlier scoring techniques, which can be classified into two categories: algorithms for static data in section 3.1 and algorithms for streaming data in section 3.2. In addition, examples of calculating outlier detection in static data are illustrated in section 3.3.

## 3.1 Outlier detection algorithms in static data

To detect outliers in static data, the user inputs the entire data points in the data set $D$ and user-defined parameters to the model, it processes the entire data points in main memory and assigns outlier scores to the entire data points. Note that some algorithms don't need any user-defined parameters.

### 3.1.1 Local Outlier Factor (LOF)

The local outlier factor (LOF) [41] algorithm measures the local density of data points according to its $k$ nearest neighbor. If the region of a measured data point has a lower density than its neighbor by comparing the local density of itself to its neighbors, it could be considered an outlier. This LOF shares the concept of reachability distance and core distance similar to DBSCAN and OPTICS. The LOF algorithm consists of the following steps:

**Step 1:** Calculate the $k$-distance between each data point in $D$ and its $k$-nearest neighbors.



**Step 2:** Calculate the reachability distance of all data points in $D$ according to definition 3.1.

**Definition 3.1** (Reachability distance of a data point $p$ w.r.t. a data point $o$)

The reachability distance is the maximum of $d(p,o)$ and $k\text{-}distance(o)$. In other words, the reachability distance of data point $p$ from $o$ is the true distance of these two data points, but at least $k$-distance of o. It is mainly introduced for smoothing local density, which is defined as

$$reach\text{-}dist_k\,(p,o) = max\,\{k\text{-}distance(o), d\,(p,o)\}$$

**Step 3:** Calculate the local reachability density of each data point $p$ in $D$ according to definition 3.2.

**Definition 3.2** (Local reachability density of a data point $p$)

The local reachability density of a data point $p$ relative to $\mathcal{N}_k(p)$is the inverse of the average reachability distance over $\mathcal{N}_k(p)$. It is defined as

$$lrd(p) = \frac{|\mathcal{N}_k(p)|}{\sum_{o \in \mathcal{N}_k(p)} reach\text{-}dist_k(p, o)}$$

**Step 4:** Calculate the local outlier factor of each data point $p$ in $D$ according to definition 3.3.

**Definition 3.3** (Local outlier factor of a data point $p$)

The local outlier factor is the ratio between the average local reachability density of the neighborhood to that of the data point $p$. It is defined as

$$LOF(p) = \frac{\sum_{o \in \mathcal{N}_k(p)} lrd(o)}{|\mathcal{N}_k(p)| \cdot lrd(p)}$$

The LOF score is very sensitive to the number of nearest neighbors. The author of LOF provided a suggestion for $k$-nearest neighbors to smooth statistical fluctuations. The lower bound should be about 9, and the upper bound should be approximately 20.

The expectation of the LOF score for a normal data point is close to 1, while the LOF score of an outlier will be greater than 1. If a data set is sufficiently large, the criterion of an outlier can be that the $LOF$ score is larger than 1.

### 3.1.2 k-Nearest Neighbors (kNN)



The $k$ Nearest Neighbors (kNN) [42] algorithm is a simple algorithm to measure an outlier score of a data point in $D$. This algorithm uses the average or the median or the maximum of $k$-nearest neighbors. In this thesis, the average will be used for the KNN algorithm since most papers report with the average. The kNN algorithm consists of the following steps:

**Step 1:** Calculate the $k$-distance between each data point in $D$ and its $k$-nearest neighbors.

**Step 2:** Calculate the kNN score of each data point $p$ in $D$ according to definition 3.4.

**Definition 3.4** ($k$-nearest neighbors score of a data point $p$)

$kNN$ score, the average distance of $k$-nearest neighbors of data point $p$, is defined as

$$kNN(p) = \frac{\sum_{o \in \mathcal{N}_k(p)} d(p, o)}{|\mathcal{N}_k(p)|}$$

## 3.1.3 Ordered difference distance Outlier Factor (OOF)

The ordered difference distance outlier factor (OOF) [22] algorithm is a parameter-free outlier score that does not require any parameter. It uses the contribution of other points to the computed point using the difference distance by projecting other data points in $D$ along the line from the computed point to the reference point.

The input of the OOF algorithm is a data set $D$. The output is the OOF scores of all data points in $D$. Given $p_r, p_i, p_j \in D$ where $p_r$ is the reference data point, $p_i$ is the computed data point, and $p_j$ is the contributed data point. The OOF algorithm consists of the following steps:

**Step 1:** Calculate all pairwise distances of each data point $p$ in $D$.

**Step 2:** Create a line from each data point to its farthest data point, then project other data points to this line, calculating the order difference distances of data points $p_r$ and $p_i$ according to definition 3.5.

**Definition 3.5** (Order difference distances between data points $p_r$ and $p_i$)

$$Diff(p_r, p_i) = min\{d(p_r, p_i) - d(p_r, p_j) | d(p_r, p_i) > d(p_r, p_j) \text{ and } p_i \neq p_r\}$$



**Step 3:** Calculate OOF scores of each data point $p$ in $D$ according to definition 3.6.

**Definition 3.6** (OOF of a data point $p_i$)

OOF score is the average minimum between the different distances, and its 1-$distance(p_i)$. It is defined by

$$OOF(p_i) = \frac{\sum_{r=1}^{n} min\ (Diff(p_r, p_i), 1\text{-}distance(p_i))}{n - 1}$$

OOF shares the idea of the nearest distance as an outlier score. So, the $OOF$ score should not be greater than 1-$distance(p_i)$ which may cause some problems in a normal region. If 1-$distance(p_i)$ is not used, it may lead to a problem that the outlier in the micro-outlier cluster may have the same OOF scores or have a lower value similar to data points in the normal region. This algorithm has time complexity $O(n^2 logn)$.

### 3.1.4 Acute angle order difference distance Outlier Factor (AOF)

Acute angle order difference distance outlier factor (AOF) [24] algorithm measures outlier scores similar to the unblocked different distance of OOF but adds a condition that the acute angle of reference data point $p_r$ to a pair of cover data point $p_i$ and measured data point $p_j$. The AOF algorithm does not use the order different distance method for all data points. The details of the AOF algorithm are discussed step-by-step here:

**Step 1:** Calculate all pairwise distances of each data point $p_i$ in *D*.

**Step 2:** Create a distance vector of each data point to its farthest data point, then project other data points to it by calculating the order distance of each reference data point $p_r$ to other data point $p_i$ in $D$.

**Step 3:** Calculate the order of different distances of all data points $p_r$ and $p_i$ by adding the acute angle condition according to definition 3.7.

**Definition 3.7** (Acute angle different distances of data points $p_r$ and $p_i$)

$$aDiff(p_r, p_i) = min\left\{ d(p_r, p_i) - d(p_r, p_j) \middle| \begin{array}{c} d(p_r, p_i) > d(p_r, p_j), \\ d(p_i, p_j)^2 < d(p_r, p_i)^2 + d(p_r, p_j)^2 \\ p_i \neq p_r \end{array} \right\}$$

**Step 4:** Calculate AOF scores of each data point $p_i$ in *D* according to definition 3.8.

**Definition 3.8** (AOF of a data point $p_i$)

$$AOF(p_i) = \frac{\sum_{r=1}^{n} aDiff(p_r, p_i)}{n-1}$$



This algorithm is based on OOF, but it includes a condition with an acute angle condition. As a result, unlike the OOF algorithm, this algorithm does not require a minimum first nearest neighbor. The performance of this algorithm was slightly better than that of OOF. Due to the additional angle condition, this algorithm has a high time complexity $O(n^3)$.

### 3.1.5 Angle-Based Outlier Detection (ABOD)

Angle-based outlier detection (ABOD) [25] algorithm measures outlier score as a variance of the angle between data points in *D* weighted by distance, rather than using a distance of neighborhoods as proximity-based concepts. This algorithm is parameter-free and detects outliers in high dimensions.

In practice, the outlier scores relative to not only the angle but also the distance between the points are weighted by division in order to account for distance. So, the low outlier score led to an outlier, while the high outlier score led to a normal

data point. the ABOD algorithm consists of $A$ as a measured data point, $B$ and $C$ are a pair of data points w.r.t $A$.

**Step 1:** Calculate all pairwise distances of each data point $A$ in $D$.

**Step 2:** Find a set of all pairs of data points in $D$ with respect to data point *A*.

**Step 3:** Calculate variance over the angles between the difference $A$ to its all pairs of data points in $D$, weighted by the distance of its pair of data points according to definition 3.9.

**Definition 3.9** (Angle-based outlier factor of a data point $A$)

For two data points $B, C \in D$, $BC$ denotes the difference vector $C - B$. The angle-based outlier factor $ABOF(A)$ is the variance over the angles between the difference vector of $A$ to all pairs of points in $D$ weighted by the distance of a pair of data points

$$ABOF(A) = VAR_{B,C \in D} \left(\frac{AB \cdot AC}{||AB||^2 \cdot ||AC||^2}\right)$$

The basic approach of the ABOD algorithm has an obvious flaw. Since all pairs of points must be considered for each data point that has a time complexity of $O(n^3)$. This is not appealing when compared to the OOF or the LOF algorithm.

To reduce the time complexity, the Fast Angle-Based Outlier Detection (FastABOD) [25] algorithm is an approximation algorithm that approximates ABOF using a sample of data points from $k$-nearest neighbors instead of the entire data set $D$. Because data points that are far from the measured data point are less important due to being weighted by distance. As a result, employing the nearest neighbors may yield a more accurate approximation, particularly in low-dimensional data sets where the distance is more significant. FastABOD consists of the following steps:



**Step 1:** Calculate the distance between each data point in $D$ and its $k$-nearest neighbors.

**Step 2:** Find all pairs of data points in $D$ with respect to each data point and its $k$-nearest neighbors.

**Step 3:** Calculate variance over the angles between the differences between each data point to its all pairs of data points in its $k$-nearest neighbors, weighted by the distance of the data point according to definition 3.10.

**Definition 3.10** (Fast angle-based outlier factor of a data point $A$)

For two points $B, C \in D, BC$ denotes the difference vector $C - B$. $N_k(A)$ is denoted as the set of $k$-nearest neighbors of $A$. The approximate angle-based outlier factor $FastABO(A)$ is the variance over the angles between the difference vector of $A$ to all pairs of data points in $N_k(A)$ weighted by the distance:

$$FastABOF(A) = VAR_{B,C \in N_k(A)} \left( \frac{AB \cdot AC}{||AB||^2 \cdot ||AC||^2} \right)$$

Although the FastABOD algorithm has a time complexity lower than ABOD, FastABOD has accuracy lower than ABOD. To use this algorithm to detect outliers, FASTABOF and ABOF scores need to be used oppositely because the high score leads to a normal data point, while the low score leads to an outlier.

# 3.2 Outlier detection algorithm in streaming data

To detect outliers in streaming data, the user inputs the streaming data $S$ and user-defined parameters to the model. Model processes each data point or chunk of data points that are sent from streaming data $S$ and assigns outlier scores. This processing continues until it stops generating streaming data.

## 3.2.1 Streaming Half-Space-Trees (HS-Trees)

The Streaming Half-Space-Trees or HS-Trees [37] algorithm is one class classification anomaly detection that can measure outlier scores. The model of the HS-Trees algorithm, which is tree-based, updates new data points and measures its outlier score. To measure outliers, the HS-Trees algorithm uses mass (then the number of data points) as the outlier score. It consists of nodes in binary trees which can consist of the number of data points and the depth of the tree, which is used for measuring outlier scores that make the HS-Trees algorithm calculate faster than distance and density-based. The HS-Trees algorithm uses two windows, a reference window and a current window. At the start of detection, the HS-Trees algorithm learns the mass profile in the reference window to measure outlier scores in the current window. A data point in a high-density area is a normal data point, while a data point in a low-density area is an outlier. A new data point is added in the current window until full. The mass profile in the reference window is discarded instead of the mass profile in the current window for measuring the next batch of data points. The streaming process continually without reconstructing the group of the binary trees. The notation in the HS-Trees algorithm is defined in Table 3.1.

Table 3.1 Notation of the HS-Trees algorithm

| | |
|---|---|
| $X$ | A streaming data point |
| $n$ | The number of streaming data points |
| $\mathcal{T}$ | $\mathcal{T}^{th}$ full binary tree |
| $Node$ | A node in a full binary tree |
| $\mathcal{K}$ | The current depth of a node or $Node^{*}\mathcal{K}$ |
| $w$ | A window size |
| $\mathcal{E}$ | The number of HS-Trees in an ensemble |
| $h$ | The maximum depth (level) of a tree |
| $r$ | A mass of a node in the reference window |
| $l$ | A mass of a node in the latest window |
| $Score(X, \mathcal{T})$ | A score from $\mathcal{T}^{th}$ full binary tree |
| $HS\text{-}trees(X)$ | An HS-trees score |

The HS-Trees algorithm uses a full binary tree consisting of $2^{h+1} - 1$ nodes, in which all leaves are at the same depth, $h$.

The idea of the HS-Trees algorithm is to create a full binary tree in which each node is along $q$ dimensions. After that, the midpoint is used to bisect the workspace into two half-spaces and creates the left child and right child of the node until the binary tree is full.

Each node contains the following information: (1) an array of minimum and maximum values of each dimension; (2) variables $r$ and $l$, which consist of a mass profile of the reference and the current windows, respectively; (3) variable $\mathcal{K}$, which is the current depth of the binary tree; (4) the current node's left and right children, each associated with half-space after the split.

Creating the diverse full binary tree is achieved by using the following procedure. Initialize workspace, before constructing the binary tree, and normalize attributes' ranges to [0, 1]. Let $s_q$ be the uniform random [0,1)of dimension $q^{th}$, so $max_q$ is $(s_q + 2 \cdot \max(s_q, 1 - s_q))$ and $min_q$ is $(s_q - 2 \cdot max(s_q, 1 - s_q))$ which is defined by every $q$ dimension. Each node is randomly selected dimension $q^{th}$ to form two half-spaces; the split point is the midpoint of the current range of $q$. Initially, all mass profiles are set to zero.

After building full binary trees, a mass profile of data points must be updated in the tree to measure the outlier score. The data point, $x$, traverses through node and updates mass $r$ and $l$. The reference window variable is boolean. If it is true, update the reference window; otherwise, the current window.

To measure an outlier score, the properties of the depth and the mass profile of each full binary tree are used to measure the outlier score. Let $Score(X, \mathcal{T})$ be a function that traverses a test data point $X$ from the root of an $HS\text{-}Tree\ (\mathcal{T})$ until

reaching a terminal node. This function then returns the outlier score of a data point $X$ by evaluating $Node^*r \times 2^{Node^*\mathcal{K}}$, where $Node^*\mathcal{K}$ is the depth level of the terminal node containing $Node^*r$ data points. Here, a $Node^*$ is a node that has reached the maximum depth or a node that contains size-limit data points. The final score for $X$ is the sum of scores obtained from each full binary tree. The $sizeLimit$ is not a critical parameter, and a good default setting is $0.1w$, where $w$ is the window size.

The operational procedure for the HS-Trees algorithm, including two stages, is as follows:

1) Initialization stage:

**Step 1:** Build the full binary trees according to user-parameter $\mathcal{E}$.

**Step 2:** Add the first $w$ data points of the stream to record its initial reference mass profile in full binary trees. Since these data points come from the initial reference window, only the mass $r$ of each traversed node is updated.

2) Incremental stage:

**Step 1:** Calculate a HS-trees score of a new data point $X$.

**Step 2:** Update the mass profile in full binary trees. At the end of each window, the model is updated. any mass $r$ in the nodes is discarded and replaced by mass $l$ in the same nodes. Initially, mass $l$ of all nodes are set to zeros.

## 3.3 Examples of calculating outlier detection in static data

This section will show an example of how to calculate state-of-the-art outlier detection techniques in section 3.1. with only static data.



### 3.3.1 Examples of calculating outlier scores

The data set $D$ is {[0,0], [0,1], [1,1], [4,4]} in two-dimensional space and these 4 data points in Figure 3.1. All pairwise distances of data points in $D$ are shown in Table 3.2. To better understand the computation, this section uses the Manhattan distance instead of the Euclidean distance.

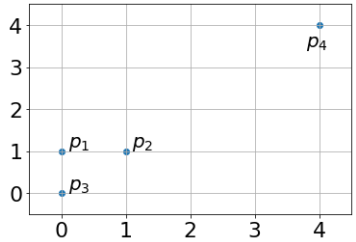



Figure 3.1 2d synthesized data set for calculating outlier scores

Table 3.2 All pairwise distances of data points in *D*

| $d(p_1, p_2)$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|
| $p_1$ | 0 | 1 | 1 | 7 |
| $p_2$ | 1 | 0 | 2 | 6 |
| $p_3$ | 1 | 2 | 0 | 8 |
| $p_4$ | 7 | 6 | 8 | 0 |

## 3.3.2 LOF algorithm

The procedure for calculating the LOF algorithm with the data set from section 3.3.1 is shown as follows:

**Step 1:** Calculate the $k$-distance between each data point in $D$ and its $k$-nearest neighbors.

$$2\text{-}distance(p_1) = 1, \quad 2\text{-}distance(p_2) = 2,$$
$$2\text{-}distance(p_3) = 2, \quad 2\text{-}distance(p_4) = 7,$$
$$\mathcal{N}_2(p_1) = \{p_2, p_3\}, \quad \mathcal{N}_2(p_2) = \{p_1, p_3\},$$
$$\mathcal{N}_2(p_3) = \{p_1, p_2\}, \quad \mathcal{N}_2(p_4) = \{p_2, p_1\},$$

**Step 2:** Calculate the reachability distance of all data points in $D$.

$$reach\text{-}dist_2(p_1, p_2) = 2, \quad reach\text{-}dist_2(p_1, p_3) = 2,$$
$$reach\text{-}dist_2(p_2, p_1) = 1, \quad reach\text{-}dist_2(p_2, p_3) = 2,$$
$$reach\text{-}dist_2(p_3, p_1) = 1, \quad reach\text{-}dist_2(p_3, p_2) = 2,$$
$$reach\text{-}dist_2(p_4, p_2) = 6, \quad reach\text{-}dist_2(p_4, p_1) = 7,$$

**Step 3:** Calculate the local reachability density of data points in $D$.

$$lrd(p_1) = \frac{|\mathcal{N}_2(p_1)|}{(reach\text{-}dist_2(p_1, p_2) + reach\text{-}dist_2(p_1, p_3))} = 0.5$$

$$lrd(p_2) = \frac{|\mathcal{N}_2(p_2)|}{(reach\text{-}dist_2(p_2, p_1) + reach\text{-}dist_2(p_2, p_3))} = 0.667$$

$$lrd(p_3) = \frac{|\mathcal{N}_2(p_3)|}{(reach\text{-}dist_2(p_3, p_1) + reach\text{-}dist2(p_3, p_2))} = 0.667$$

$$lrd(p_4) = \frac{|\mathcal{N}_2(p_4)|}{(reach\text{-}dist_2(p_4, p_2) + reach\text{-}dist_2((p_4, p_1)} = 0.154$$

**Step 4:** Calculate the local outlier factor of all data points in $D$.

$$LOF(p_1) = (lrd(p_2) + lrd(p_3))/lrd(p_1)/\,|\mathcal{N}_2(p_1)| = 1.333$$

$$LOF(p_2) = (lrd(p_1) + lrd(p_3))/lrd(p_2)/|\mathcal{N}_2(p_2)| = 0.875$$

$$LOF(p_3) = (lrd(p_1) + lrd(p_2))/lrd(p_3)/|\mathcal{N}_2(p_3)| = 0.875$$

$$LOF(p_4) = (lrd(p_2) + lrd(p_1))/lrd(p_4)/|\mathcal{N}_2(p_4)| = 3.792$$

## 3.3.3 OOF algorithm

The procedure for calculating the OOF of the data set from section 3.3.1 is shown as follows:

**Step 1:** Calculate all pairwise distances of data points in *D*. This step is already calculated in Table 3.2.

**Step 2:** Create a distance vector of each data point to its farthest data point then, project other data points to it then calculate the different distances of data point $p_r$ and $p_i$ Calculate the different distances of data point $p_r$ and $p_i$. The result of $Diff(p_r, p_i)$ is as follows in Table 3.3.

Table 3.3 The value of *Diff* ($p_r$, $p_i$)

| $Diff(p_r, p_j)$ | $p_j$ | | | |
|---|---|---|---|---|
| $p_r$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
| $p_1$ | 0 | 1 | 1 | 1 |
| $p_2$ | 1 | 0 | 1 | 1 |
| $p_r$ | 0 | 1 | 0 | 1 |
| $p_r$ | 6 | 4 | 6 | 0 |

**Step 3:** Calculate OOF scores of the data point $p_i$.

$$1\text{-}distnace(p_1) = 1, \quad 1\text{-}distnace(p_1) = 1,$$

$$1\text{-}distnace(p_3) = 1, \quad 1\text{-}distnace(p_4) = 6,$$

$$OOF(p_1) = \sum_{r=1}^{n} \frac{min(aDiff(p_r, p_1), 1)}{3} = 1$$

$$OOF(p_2) = \sum_{r=1}^{n} \frac{min(aDiff(p_r, p_2), 1)}{3} = 1$$

$$OOF(p_3) = \sum_{r=1}^{n} \frac{min(aDiff(P_r, p_3), 1)}{3} = 0.667$$

$$OOF(p_4) = \sum_{r=1}^{n} \frac{min(aDiff(p_r, p_4), 6)}{3} = 5.333$$

### 3.3.4 kNN algorithm

The procedure for calculating the kNN algorithm with the data set from section 3.3.1 is shown as follows:

**Step 1:** Calculate the $k$-distance between each data point in $D$ and its $k$-nearest neighbors.

$$\mathcal{N}_2(p_1) = \{p_2, p_3\}, \quad \mathcal{N}_2(p_2) = \{p_1, p_3\},$$
$$\mathcal{N}_2(p_3) = \{p_1, p_2\}, \quad \mathcal{N}_2(p_4) = \{p_2, p_1\},$$

**Step 2:** Calculate the kNN score of each data point in D.

$$kNN(p_1) = \frac{d(p_1, p_2) + d(p_1, p_3)}{|\mathcal{N}_2(P1)|} = 2.333$$

$$kNN(p_2) = \frac{d(p_2, p_1) + d(p_2, p_3)}{|\mathcal{N}_2(P1)|} = 2.219$$

$$kNN(p_3) = \frac{d(p_3, p_1) + d(p_3, p_2)}{|\mathcal{N}_2(p_1)|} = 2.69$$

$$kNN(p_4) = \frac{d(p_4, p_2) + d(p_4, p_1)}{|\mathcal{N}_2(p_1)|} = 4.966$$

# Chapter 4
# Research Methodology

This chapter presents the Mass ratio variance-based Outlier Factor and the Streaming Mass ratio variance-based Outlier Factor.

## 4.1 Mass ratio variance-based Outlier Factor (MOF)

This section explains the motivation of the MOF algorithm, the overview of the MOF algorithm, the procedure of the MOF algorithm, and the complexity analysis for the MOF algorithm.

### 4.1.1 The motivation of the MOF algorithm

In the literature surveys, chapter 3, most proximity-based outlier detection measures outlier scores related to the $k$-nearest neighbors (parameter). For example, Figure 4.1 from [41] shows the average LOF scores for each data point in clusters $S_1$, $S_2$, and $S_3$ by grid search $minPts$ or $k$-nearest neighbors value from 10 to 50. The data points in clusters $S_2$ and $S_3$ are normal data points, while data points in the cluster $S_1$ is an outlier. The $minPts$ value between 10 and 35 leads to data points in a cluster $S_1$ being an outlier, the $minPts$ value between 36 and 45 leads to all clusters being a normal data point, and the $minPts$ value between 46 to 50 concludes that each data point in clusters $S_1$ and $S_2$ is an outlier. This illustrates that its parameter is sensitive and biased to detect outliers. The user must know the distribution of data sets or use the grid search technique for the best tune parameters. However, the grid search technique cannot be used for unsupervised learning.

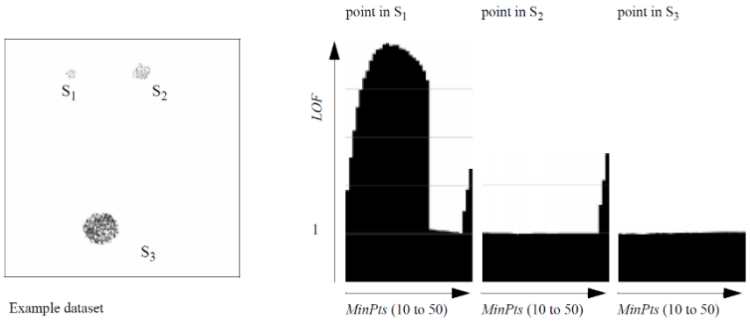



Figure 4.1 Range of LOF values for different data points in an example data set

To avoid using user-defined parameters, the methodology of measuring outlier scores by the order difference distance is proposed to be AOF [24], OOF [22], and WOF [23]. However, they fail to measure outlier scores correctly because these

algorithms are based on the first nearest neighbor ($k = 1$ in the kNN algorithm) which is distance-based. They cannot detect small clusters of outliers and also fail to detect the edge of normal regions as outliers, according to Figure 4.2 from [24]. The radius of the red circle is the outlier score. The AOF and the OOF algorithms assign a low outlier score on outlier micro-cluster around $[-2, 9]$ that should be high. While kNN can detect them correctly with the appropriate setting of parameter $k$.

The other methodology is angle-based outlier detection (ABOD) [25]. ABOD can assign scores correctly. However, ABOD needs to calculate the entire angle between two vectors in the data set, leading to a very high time complexity $O(n^3)$. To reduce the time complexity, FastABOD [25] uses only $k$ data points instead of the entire data set. However, its accuracy drops significantly.

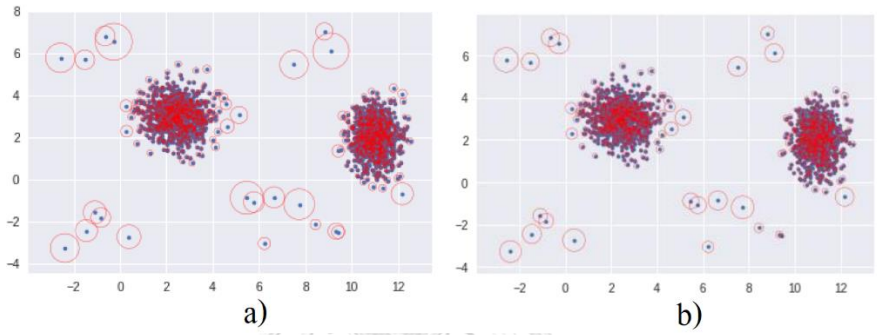



Figure 4.2 a) AOF scores and b) OOF scores in the example data set

This thesis proposes the novel unsupervised outlier score without parameters and uses the density concept. It is referred to as the mass ratio variance outlier factor (MOF). Since the scoring degree of an outlier, which is the so-call the MOF score, uses the whole contributed data points in the data set.

Give $\rho, \rho_1, \rho_2$ as density, $M, M_1, M_2$ as mass, and $V$ as volume. According to the formula, the ratio of mass is equal to the ratio of density, if the volumes are the same.

$$\rho = \frac{M}{V}, \qquad \frac{\rho_1}{\rho_2} = \frac{M_1 \times \cancel{V}}{M_2 \times \cancel{V}} = \frac{M_1}{M_2}$$

According to the formula, MOF defines volume ($V$) as the space of the hypersphere of the same radius and mass as the number of data points inside the hypersphere, according to Figure 4.3. It shows that the mass ratio of a contributed data point $q$ (green line) with respect to a measured data point $p$ (red line) is the ratio of the number of data points inside the green and red hypersphere. It can be explained that the isolated data point $p$ has a mass less than an engulfed data point $q$ ($2 < 5$).

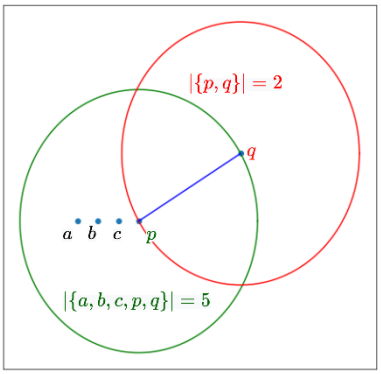



Figure 4.3 The mass ratio of $p$ with respect to $q$

The measured data point $p$ must be contributed by all other data points in a data set. The following figures show the differences between the mass of a normal data point and an outlier. The following Figure 4.4 and Figure 4.5 show the differences between the mass of all contributed data points with respect to an outlier a normal data point and. It shows that the data set includes normal data points ($P_1, P_2, P_3$, and $P_4$) and an outlier ($P_5$) . The details are as follows:

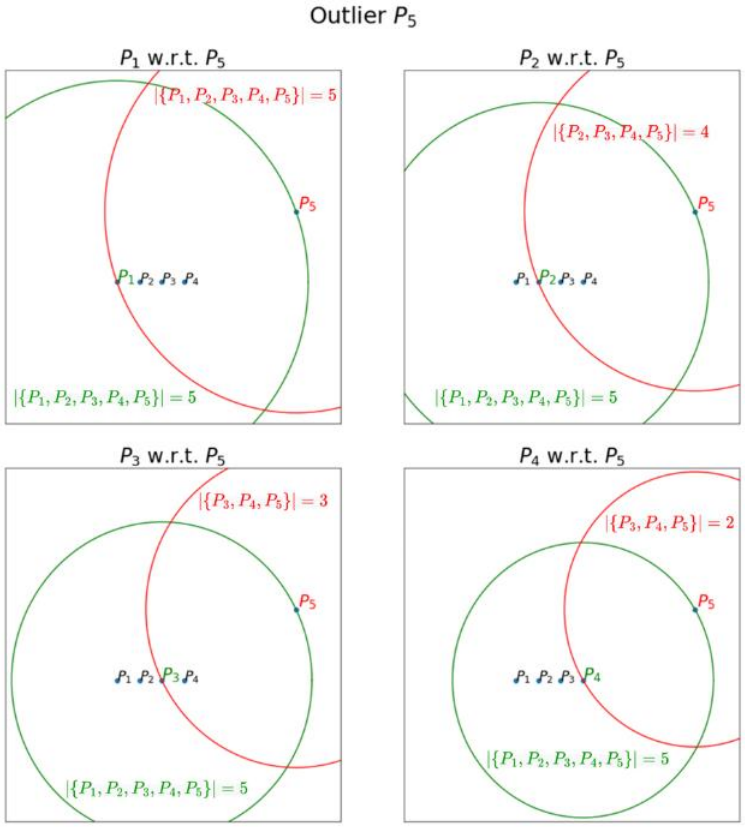



Figure 4.4 Mass ratio of other data points w.r.t. an outlier

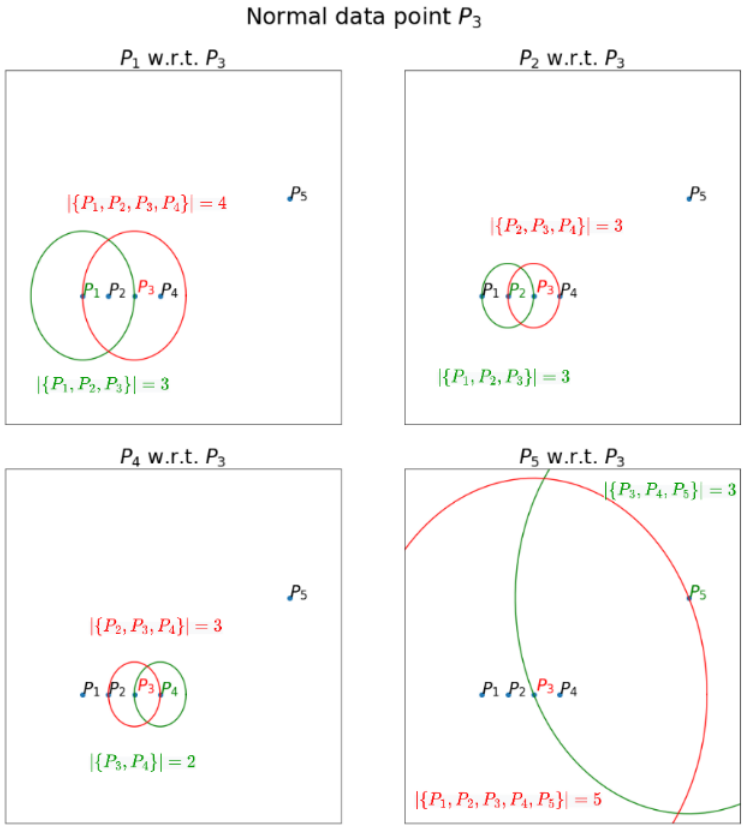



Figure 4.5 The mass ratio of other data points w.r.t a normal data point

- Figure 4.4 shows the mass ratio of $P_1, P_2, P_3$, and $P_4$ (green circle) with respect to $P_5$ (red circle) is equal to 5/5, 5/4, 5/3, and 5/2. The variance of them is 0.32
- Figure 4.5 shows the mass ratio of $P_1, P_2, P_4$, and $P_5$with respect to $P_3$ is equal to 3/4, 3/3, 2/3, and 3/5. The variance of them is 0.023
- In conclusion, a MOF score of $P_5$ is around 10 times larger than a MOF score of $P_3$. This is because the number of outliers in data sets is always less than the normal data points in data sets. Therefore, the outlier is contributed by normal data points rather than outliers resulting in high mass ratio variance, while normal data points are supported by the same normal data points, resulting in the low mass ratio variance.

So, the first advantage of MOF is to calculate the variance of mass ratio that can measure outlier scores of outliers and normal data points in a data set. The second advantage of MOF is that the outlier score is based on density-based, which can detect local outliers, which distance-based cannot do. The third advantage is MOF avoids using a parameter to measure density around data points. The details of MOF are presented in the next section.

### 4.1.2 The overview of the MOF algorithm

The MOF algorithm measured the MOF scores of data points in the entire data set. Its input is a data set $D$ with $n$ data points and $d$ dimensions, and its output is the MOF scores of each data point in $D$. Given $p$ is a measured data point which $p \in D$, and $q$ is a contributed data point which $q \in D - \{p\}$.

The notation of this section is the neighborhood of a data point $q$ with respect to a data point $p$, the mass ratio of a data point $q$ with respect to a data point $p$, and the mass ratio variance outlier factor of data point $p$. Note that all of these are sequentially consistent.

Definition 4.1 (Neighborhoods of a data point $q$ w.r.t. a data point $p$)

To calculate the mass of data points, MOF uses the idea of neighborhoods of data point $q$ with respect to data point $q$ or $N_p(q)$ is defined as the set of data points that lies within the hypersphere centered at data point $q$ with the radius $d(p,q)$ or the mass of a data point $q$ with respect to a data point $p$. $N_p(q)$ is defined as follows:

$$N_p(q) = \{o \in D | d(q,o) \leq d(p,q)\}$$

However, MOF does not need to find the members of $N_p(q)$, but it needs to find the cardinality of $N_p(q)$ or $|N_p(q)|$ for calculating a mass ratio. In addition, MOF needs to find $N_p(q)$ with all $p$ and all $q$ in a data set $D$. Therefore, the cardinality of a neighborhood of all contributed data points with respect to other data points is calculated by expanding the radius to reach from the farthest data points for themselves to create hyperspheres. For example, Figure 4.6 shows that a data point $P_3$ expands the radius to reach all data points in a space to create hyperspheres. Data point $P_5$ is the fifth farthest, $P_1$ is the fourth farthest, $P_2$ and $P_4$ are the third farthest (co-rank), and $P_3$ is the first farthest. The rank distance can be used as the mass of data points inside a hypersphere. Therefore, $|N_{P_3}(P_5)|$ is equal to 5, $|N_{P_3}(P_1)|$ is equal to 4, $|N_{P_3}(P_2)|$ and $|N_{P_3}(P_4)|$ equal to 3, and $|N_{P_3}(P_3)|$ is equal to 1. However, it must add a condition in case of any redundant outliers entering the data set.

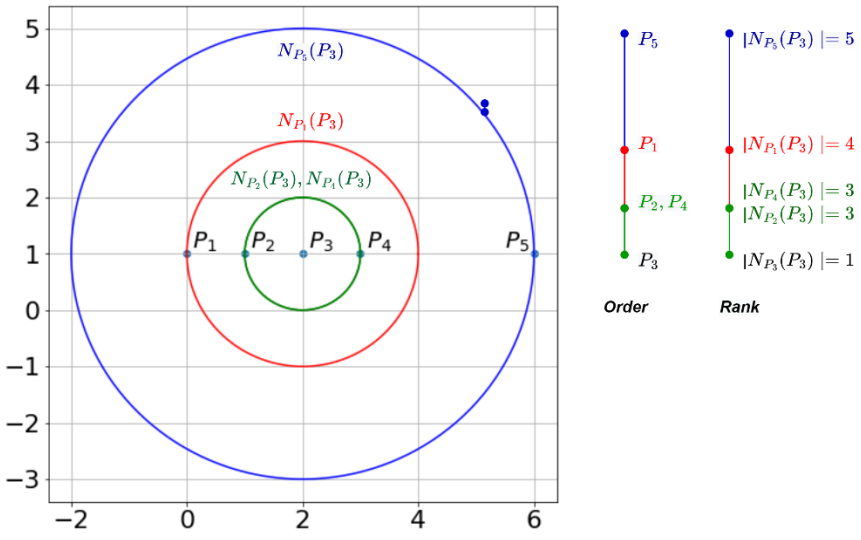



Figure 4.6 Mass and neighborhoods of other data points with respect to $P_3$

**Definition 4.2** (Mass ratio of data point $q$ with respect to data point $p$)

The ratio of the number of data points lies within the hypersphere centered at data point $q$ to lies within the hypersphere centered at data point $p$:

$$massR_p(q) = \frac{|N_p(q)|}{|N_q(p)|}$$

To explain how to compute the $massR_p(q)$, consider a sample data set with 13 data points, as shown in Figure 4.7. $P_1, P_4, P_9$, and $P_{12}$ are the corner points. $P_6$ and $P_7$ are the inner points.

To calculate the $massR_{P_{13}}(P_4)$ and $massR_{P_4}(P_{13})$. Note that two points ($p_{12}$ $and$ $p_3$) are inside the red hypersphere centered on $P_{12}$, therefore $|N_{P_{13}}(P_4)|$ is equal to 13. Also, 13 data points (all points) are inside the green hypersphere centered on $p_3$, therefore $|N_{P_4}(P_{13})|$ is equal to 2. Thus, $massR_{P_{13}}(P_4)$ is equal to 13/2 and $massR_{P_3}(P_{13})$ is equal to 2/13.

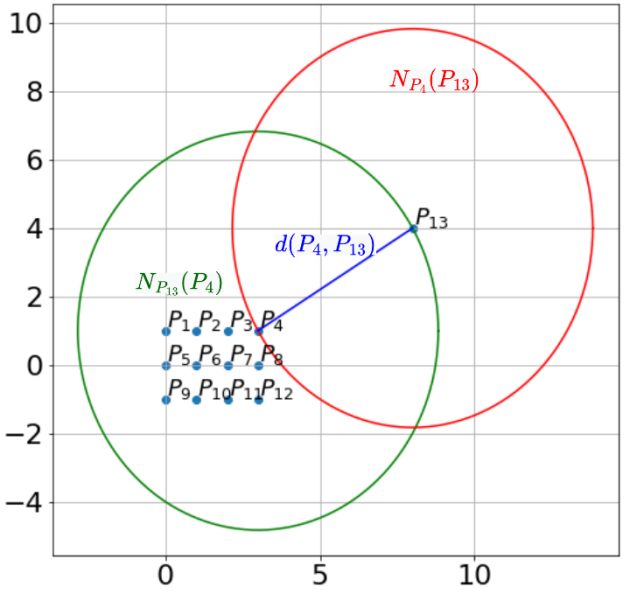



Figure 4.7 Mass ratio of data points with respect to data points $p$

Intuitively, the minimum of $|N_P(q)|$ is 2 (data points $p$ and $q$), and the maximum of $|N_P(q)|$ is $n$ (all data points in the data set). So, an interval of $massR_p(q)$ is in $[2/n, n/2]$. The strength of this mass ratio effect is that outliers are distributed from normal data points with a wider range of values than normal data points leading to high variance.

So far, only one mass ratio of a contribution with respect to a measured data point is calculated. The notation of calculations is defined as mass ratio variance-based outlier score of data point $p$ or MOF scores of data point $p$.

**Definition 4.3** (Mass ratio variance outlier factor of a data point $p$)

$\mu_p$ is defined as the average mass ratio of other data points with respect to the data point $p$ or MOF score of data point $p$. $MOF(p)$ is defined as the variance of the mass-ratio distribution of other data points with respect to a data point $p$:

$$\mu_p = \frac{\sum_{q \in D - \{p\}} massR_p(q)}{n-1}$$

$$MOF(p) = \frac{\sum_{q \in D - \{p\}} \left(massR_p(q) - \mu_p\right)^2}{n-1}$$

The outlier score of data point $p$ is equal to $MOF(p)$. It is the variance of the mass ratio of data points in a data set except for data point $p$ with respect to data point $p$.

Continually calculating, the boxplots of the mass ratio of other data points with respect to data points in the data set is calculated and shown in

Figure 4.8a). After the box plot of all data points is summarized by variance resulting in the bar chart in

Figure 4.8b) in which the height of a bar is a MOF score. It shows that $MOF(P_{13})$ is higher than other data points exhibiting as an outlier. Another interesting, the corner data points ($P_1$, $P_2$, $P_4$, and $P_{12}$) also have higher MOF scores than the data points inside a cluster.

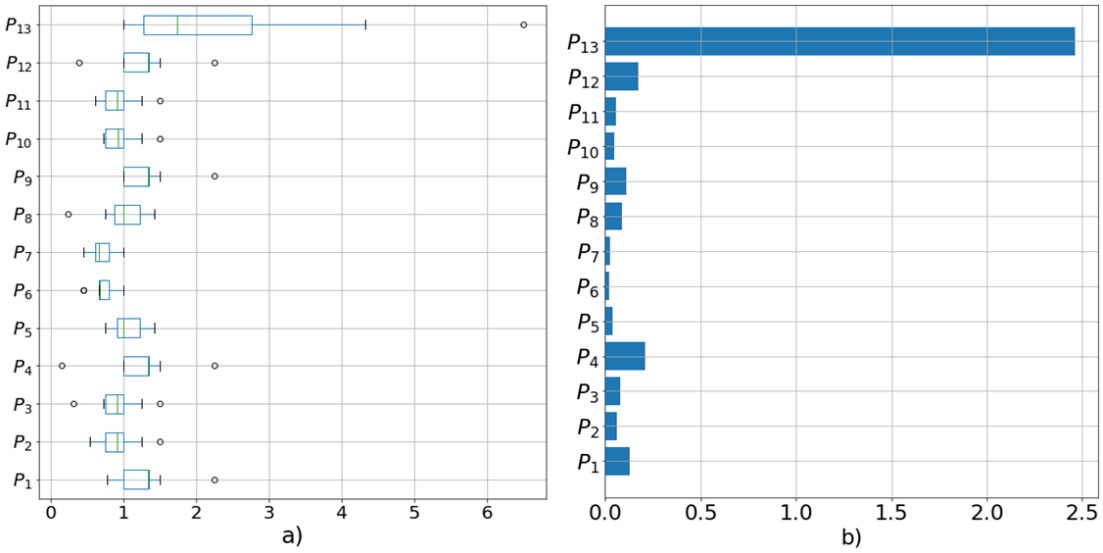



Figure 4.8 a) The boxplots of the mass ratio of data points $q$ with respect to data points $p$ and b) $MOF$ scores of data points in $D$

## 4.1.3 The procedure of the MOF algorithm

The flowchart of the MOF algorithm workflow is provided in Figure 4.9 and the pseudo code of the MOF algorithm is in Figure 4.10. The steps are as follows:

**Step 1:** Input a data set ($D$).

**Step 2:** Calculated distance matrix ($DM$) in line 1.

**Step 3:** Calculated sorted distance matrix ($SDM$) and sorted distance index matrix ($SIM$) in line 2-3.

**Step 4:** Calculate the mass ratio matrix ($MassRatio$) in line 4-20. It also adds conditions in line 8-9 to transform sorted and index distance to rank distance and condition of duplicated data points in line 14-16.

**Step 5:** The MOF score of each data point ($MOF$) is calculated in line 21-25 and returned as an array that is ordered according to data points in the data set.

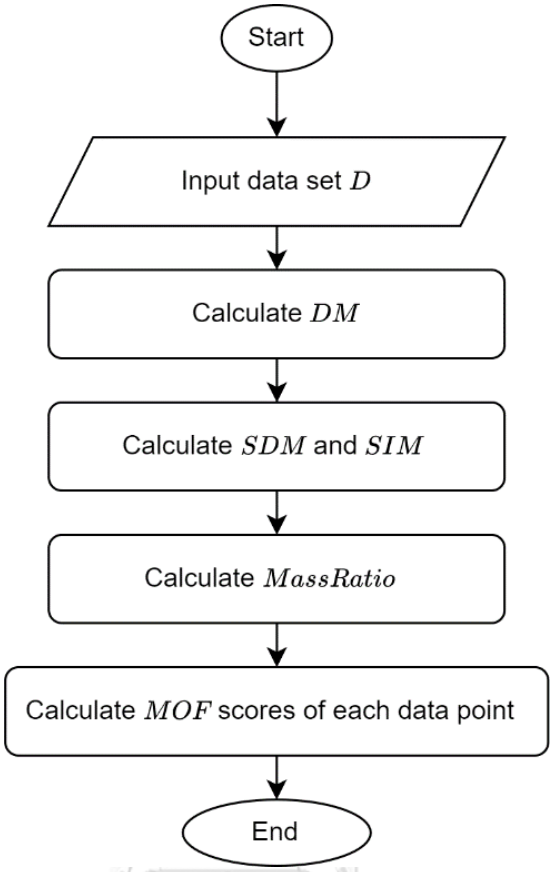



Figure 4.9 The MOF flowchart

**Algorithm** $ProcessMOF(D)$

**Input** : $D -$ a $data\ set$

**Output** : $MOF -\ MOF\ scores$

**Line**:

1. $DM \leftarrow DM$
2. $SDM \leftarrow SDM$
3. $SIM \leftarrow SIM$
4. $MassRatio \leftarrow [1]_{nxn}$
5. $\boldsymbol{for}\ i \leftarrow 1\ ;\ i \leq n\ ;\ i++\ do$:
6. $\quad rm \leftarrow SDM[i,n]$
7. $\quad Npq \leftarrow n$
8. $\quad \boldsymbol{for}\ k \leftarrow n\ ;\ k > 0\ ;\ k--\ do$:
9. $\quad\quad \boldsymbol{if}\ SDM[i,k]\ ! \leftarrow rm$
10. $\quad\quad\quad rm \leftarrow SDM[i,k]$
11. $\quad\quad\quad Npq \leftarrow k$
12. $\quad\quad \boldsymbol{end\ if}$
13. $\quad\quad j \leftarrow SIM[i\ ,k]$
14. $\quad\quad \boldsymbol{if}\ SDM[i,1] == 0$ and $k == 1$ do
15. $\quad\quad\quad Npq \leftarrow 1$
16. $\quad\quad \boldsymbol{end\ if}$
17. $\quad\quad MassRatio[i,j] \leftarrow MassRatio[i,\mathrm{j}]\ \times\ Npq$
18. $\quad\quad MassRatio[j,i] \leftarrow MassRatio[j,\mathrm{i}]\ \times \frac{1}{Npq}$
19. $\quad \boldsymbol{end\ for}$
20. $\boldsymbol{end\ for}$
21. $MOF \leftarrow [1]_{nx1}$
22. $\boldsymbol{for}\ i \leftarrow 1\ ;\ i \leq n\ ;\ i++\ do$:
23. $\quad \mu \leftarrow \frac{\sum_{j=1,\ i\neq j}^{n} MassRatio[i,j]}{n-1}$
24. $\quad MOF[i] \leftarrow \frac{\sum_{j=1,\ i\neq j}^{n}(MassRatio[i,j]-\mu^2)}{n-1}$
25. $\boldsymbol{end\ for}$
26. $\boldsymbol{return}\ MOF$



Figure 4.10 The MOF algorithm procedure

## 4.1.4 The complexity analysis for the MOF algorithm

To calculate MOF scores for $n$ data points in a data set $D$, the MOF algorithm implements a two-step algorithm. In the first step, a set of the nearest neighborhood of all contributed data points with respect to all measured data points is calculated. In the second step, the MOF scores are calculated. In the first step, all pairwise distances of each data point in a data set are calculated. The result of them is used to calculate a set of the nearest neighborhoods of all contributed data points with respect to outlier measured data points based on a sorted algorithm resulting in time complexity $O(n^2 logn)$. After that, a set of the mass ratio of all contributed data points with

respect to outlier measured data points are calculated by the ratio of pairwise of nearest neighborhoods resulting in run time complexity $O(n)$. The result of this step is time complexity $O(n^2 logn)$ and space complexity is $O(n^2)$ from containing all pairwise distances, nearest neighborhoods, and mass ratios in a data set. In the second step, the MOF scores are calculated by the variance of the mass ratio of each data point with respect to its contributed data points which has time complexity $O(n)$ and space complexity $O(n)$. In summary, the MOF algorithm has time complexity $O(n^2 logn)$ and space complexity $O(n^2)$.

## 4.2 Streaming Mass ratio variance-based Outlier Factor (SMOF)

This section explains the motivation of the SMOF algorithm, the overview of the SMOF algorithm, The procedure of the SMOF algorithm, and the complexity analysis for the SMOF algorithm.

### 4.2.1 The motivation of the SMOF algorithm

The MOF algorithm is designed to detect outliers in static data. If this algorithm detects outlier in streaming data, it has several limitations as follows:

1. The MOF algorithm can only work with data that is both bound and limited. It cannot detect outliers across the entire streaming data, which is infinite and unconstrained in terms of the arrival of future data points. Furthermore, the memory storage cannot store the entire streaming data according to space complexity $O(n^2)$. In addition, MOF processes the entire data set. It cannot detect outlier type I and type II.
2. The landmark, non-overlapping sliding, and overlapping sliding window models [39] are used to process data on streaming data in limited storage space. However, these techniques do not work well. They keep new data points and drop old data points by the FIFO method which cannot preserve the normal behavior of data.

To solve this limitation, this section proposed the Streaming Mass ratio variance Outlier Factor or SMOF algorithm processes in the non-overlapping sliding window for measuring outlier scores which are the so-called the SMOF scores based on the MOF algorithm. To facilitate learning of SMOF scores in evolving streaming data, the SMOF algorithm works on two consecutive windows, the reference window, followed by the current window based on the Streaming HS-Trees algorithm. Both windows have the same window size, which contains $w$ data points and is also equal to the batch from streaming data. During the initial stage of the outlier detection stage, the SMOF algorithm calculates MOF scores of each data point in both windows —the data points that fall in high-mass of both windows are normal data points, while current data points in low-mass of both windows are an outlier. —as new data points

arrive at the current window, which is the so-called current data point, and the old data points are in the reference window, which is the so-called reference data point.

One straightforward solution for detecting outliers would be to measure MOF scores in the current window and return them as outlier scores of data points, the same as the MOF algorithm. However, the MOF score cannot be distinguished between outliers and normal data points well because outliers in streaming data rarely occur. The percentage of outliers in a batch is not constant, while at some period, there is no outlier mixed in them. Therefore, SMOF uses the average MOF score of both windows dotted by the MOF score in its window, resulting in the SMOF score of each data point. The batch, which contains a large percentage of outliers, has a high average MOF score, while the batch, which contains a low percentage of outliers, has a low average MOF score. With this feature, the SMOF scores of outliers are higher than the normal data point. Note: SMOF scores of current data points are based on both current and reference data points, while outlier scores of the HS-Trees algorithm are based only on reference data points.

To better understand the differences between MOF and SMOF scores assigned by the SMOF algorithm. Figure 4.11 shows synthesized data which contains blue normal data points and red outliers generated by gaussian distribution and uniform random, respectively. It is divided into 3 periods (timestamps from 1-1000, 1001-2000, and 2001-3000). The first and the third periods contain only normal data points, while the second period is contaminated by outliers. Figure 4.12 shows the time-series graph of MOF scores and SMOF scores on a synthesized data set with a setting cut-off threshold (red line). The data points above the red line are outliers, while the data points below the red line are normal data points. Figure 4.12 also shows the detection outliers by the MOF scores cause false positive ($FP$: detect a normal data point as an outlier) greater than the detected outliers by the SMOF scores in the same true negative ($TN$: detect an outlier as an outlier) and false negative ($FN$: detect an outlier as a normal data point). Note that this example sets the window size to 250.



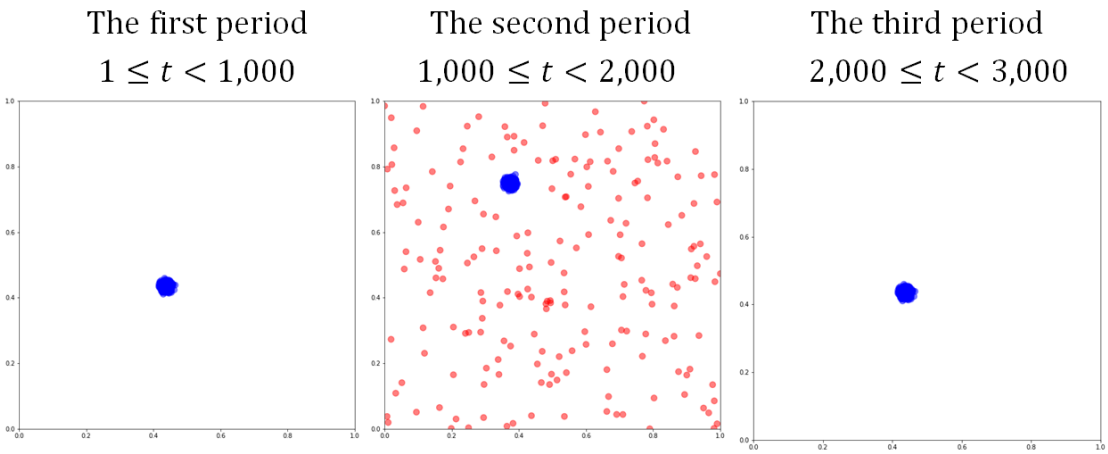



Figure 4.11 2d synthesized streaming data timestamp from 1- 3000

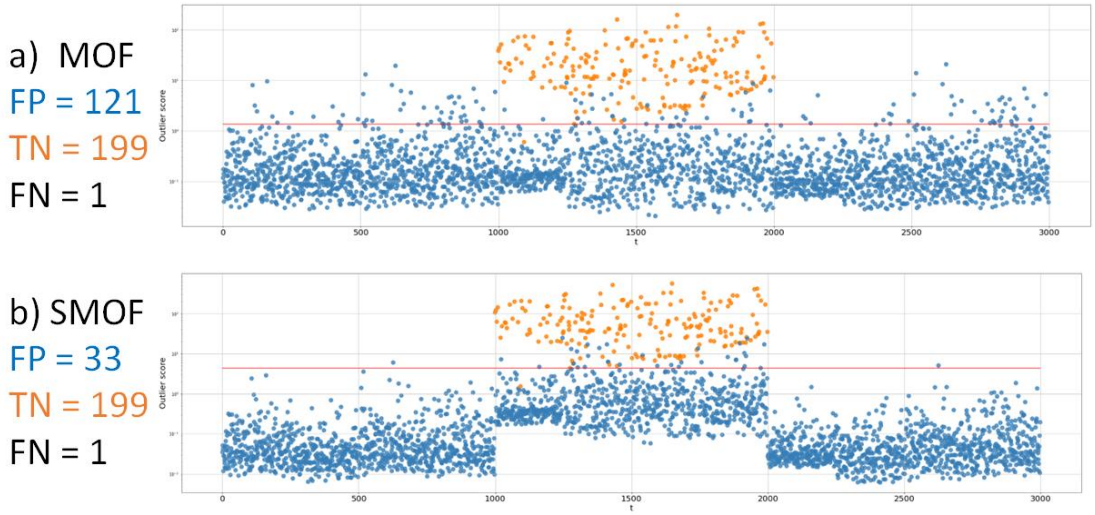



Figure 4.12 Detect outliers on time-series graph of a) MOF and b) SMOF scores

After the current data points in the current window are assigned SMOF scores, they must be removed from the current window and overridden to the reference window. Instead of the SMOF algorithm simply removing and overriding data points by the FIFO method similar to the Streaming HS-Trees algorithm, which does not preserve normal regions in the reference window. To solve this problem, the SMOF algorithm uses WRS to sample half data points from the reference and the current windows which are weighted by MOF scores. The larger MOF scores data points have, the less probability they are selected by WRS. In addition, streaming data treats the current data points as more important than the reference data points according to a characteristic of streaming data (transient). Therefore, the weight of reference data points is increased by their average MOF score. Since the SMOF algorithm collects data points from the largest to smallest weight, the weight must be the opposite value.

To better understand how the WRS algorithm can preserve in normal regions, Figure 4.13 shows the behavior of WRS sampling from four batches and 9 steps. The first batch ($T = 1$) is recorded in the reference window (Step 1), and the second batch ($T = 2$) is recorded in the current window (Step 2). Note that each batch contains 150 data points ($w$). Once the data points in both the current and the reference window are calculated MOF scores, the data points are taken a sample from 300 (150+150) to 150 data points by the WRS algorithm. The results show that the half sample of data points from the reference and the current windows can preserve the normal region and remove outliers (Step 3). This is because outliers have high MOF scores resulting in a low probability of being selected by WRS. After that sample is sent to the next reference window (Step 4) and keep the third batch ($T$ =3) in the current window (Step 5), and both windows are taken a sample again (Step 6). The result shows that the normal region in the sampling, but not in the current window, is faded. Next, the sample from Step 5 sent to the next reference window (Step 7) and the fourth batch ($T$ =4) sent into the current window (Step 8). Finally (Step 9), the WRS algorithms take a sample of both windows. It shows that old normal region is completely faded.

This is because the weight of data points in the reference window is increased by the average MOF scores of its window, resulting in current data points being more important than reference data points.

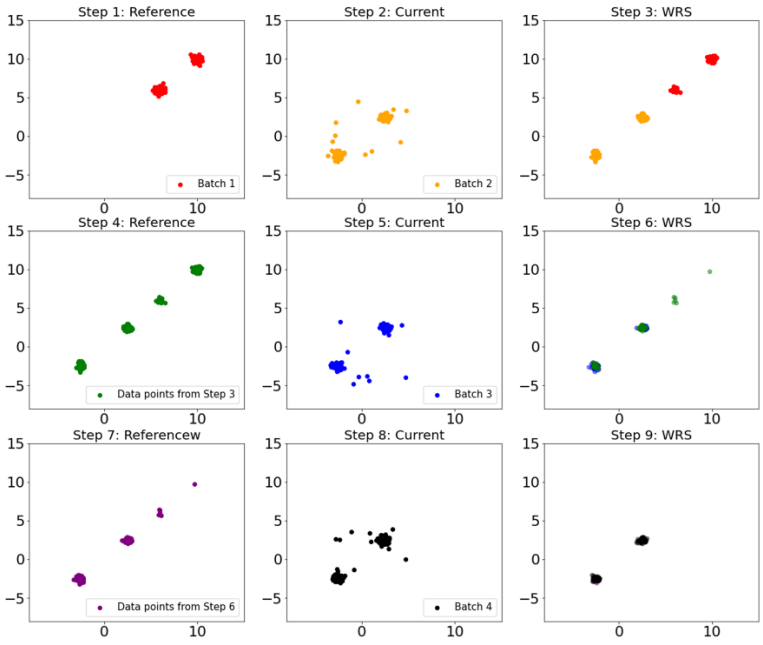



Figure 4.13 The behavior of WRS sampling from batch 1-4

## 4.2.2 The overview of the SMOF algorithm

The SMOF algorithm processes each batch of streaming data according to Figure 4.14. The key notations used to describe the method are listed in Table 4.1.



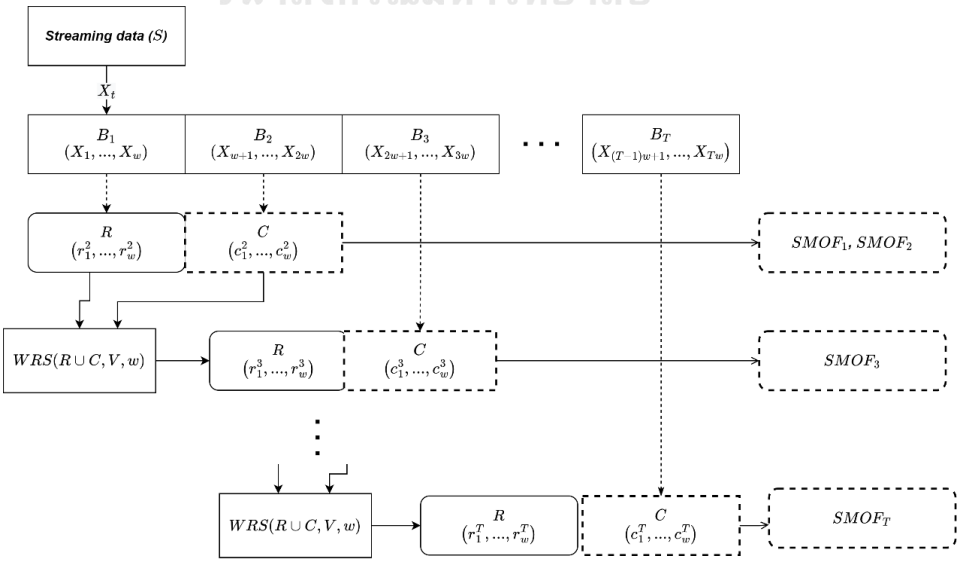



Figure 4.14 The process of the SMOF algorithm

Table 4.1 List of symbols

| | |
|---|---|
| $ProcessMOF(D)$ | Function to calculate MOF scores in $D$ |
| $MOF(p)$ | The MOF score of a data point p |
| $SMOF_T$ | $Streaming\ Mass\ ratio\ variance\ outlier\ factor\ of\ B_T$ |
| $X_t$ | $t^{th}$ data point |
| $B_T$ | $T^{th}$ Batch contain $\left(X_{((T-1)w+1}, \dots X_{(Tw)}\right)$ |
| $w$ | A window size |
| $R$ | The reference window, $r_i$ is $i^{th}$ the reference data point |
| $C$ | The current window, $c_i$ is $i^{th}$ the current data point |
| $V^R$ | The weight of $R$, $v_i^r$ is the weight of $r_i$ |
| $V^c$ | The weight of C, $v_i^c$ is the weight of $c_i$ |
| $WRS(D,V,m)$ | Random sampling $m$ data points of $D$ weighted by $V$ |

During the initial stage (the reference window is null and $T = 1$), the SMOF algorithm keeps batch $B_1$ in the reference window $R$, and also keeps batch $B_2$ in the current window $C$. Then the SMOF algorithm calculates SMOF scores ($SMOF_T$) which is defined as follows:

**Definition 4.4** (Streaming Mass ratio variance outlier factor of $B_T$)

Let $MOF^r$ and $MOF^C$ be MOF scores of data points in the reference and the current windows. $\mu^r$ and $\mu^c$ are the mean of $MOF^r$ and $MOF^C$. The Streaming Mass ratio variance outlier factor of $B_T$ is defined as $SMOF_T$ such that:

$$ProcessMOF(R \cup C) = \left(MOF(r_1), \dots, MOF(r_w), MOF(c_1), \dots, MOF(c_w)\right)$$

$$MOF^r = \left(MOF(r_1), \dots, MOF(r_w)\right)$$

$$MOF^c = \left(MOF(c_1), \dots, MOF(c_w)\right)$$

$$\mu^r = \frac{1}{w}\sum ProcessMOF(R)$$

$$\mu^c = \frac{1}{w}\sum ProcessMOF(C)$$

$$SMOF_T = \begin{cases} MOF^r \cdot \mu^r & T = 1 \\ MOF^c \cdot \mu^c & T > 1 \end{cases}$$

To measure the outlier score of the next batch ($B_{(T+1)}$), SMOF needs to clear the current data points. Thus, the reference and the current data points are sampled by WRS. Weigh of the reference, and current data points are defined as follows:

**Definition 4.5** (Weight of the reference and the current data points)

Let $\mu^{rc}$ be the mean of MOF scores in both windows. $V^R$ is defined as the weight of reference data points. $V^C$ is defined as the weight of current data points. $V$ is the weight of the reference and the current data points such that:

$$\mu^{rc} = \frac{1}{2w}\sum ProcessMOF(R \cup C)$$

$$V^R = (v_1^r, \dots, v_w^r) = -\big(MOF(r_1), \dots, MOF(r_w)\big) + \mu^{rc}$$

$$V^C = (v_1^c, \dots, v_w^c) = -\big(MOF(c_1), \dots, MOF(c_w)\big)$$

$$V = V^R \cup V^C$$

The SMOF algorithm uses WRS ($WRS(R \cup C, V, w)$) and keeps the sampling of $w$ data points from $R$ and $C$ into $R$. Once this is done, all data points in $C$ are discarded and get ready to capture the next batch of newly arriving data ($B_{(T+1)}$). This process continues as long as the stream exists. Note that after initializing stage has already passed ($T = 2$), the data points in $R$ are not from the current batch in streaming data, but from the previous batch, which already is assigned SMOF scores. Therefore, the SMOF algorithm will only report SMOF scores only on $C$.

## 4.2.3 The procedure of the SMOF algorithm

The SMOF flowchart is provided in Figure 4.16 to show and the pseudo code of the SMOF algorithm in Figure 4.15. In addition, the SMOF algorithm consists of two stages: the initialization stage to measure SMOF scores in the first batch and the second batch, and the continuous stage to measure SMOF scores after the second batch. The steps are as follows:



1) Initialization stage:

**Step 1:** Input streaming data $S$ and a window size.

**Step 2:** Initialize $R$ with the first batch $B_1$ and set $T$ to two in line 1.

**Step 3:** Streaming data sends data points $X_t$ to buffer $B_T$ until reaching $w$ data points in line 3-4.

**Step 4:** Buffer sends batch $B_T$ to $C$ in line 4.

**Step 5:** Calculate MOF scores in both $R$ and $C$ in line 7.

**Step 6:** Calculate SMOF scores in both the reference window in line 8-10 and the current window in line 13-15. All of them are returned to the user.

**Step 7:** Calculate the weight of the reference and the current data points $V$ in line 16-17.

**Step 8:** Sampling $w$ data points in $R$ and $C$ using WRS algorithm weighted by $V$ and keeping them to $R$ in line 18.

**Step 9:** Remove all data points in the current window and increase $T$ by one in line 19-20.

**Step 10:** Go back to **Step 2** and add conditions from the continuous stage.

2) Continuous stage:

After the initialization stage is finished, the reference window is filled. The main step is the same as in the initialization stage, but the SMOF score is only needed to report in the current window in **Step 6**. Therefore, line 8–12 is not processed anymore. The remaining data points in the batch can still be measured.

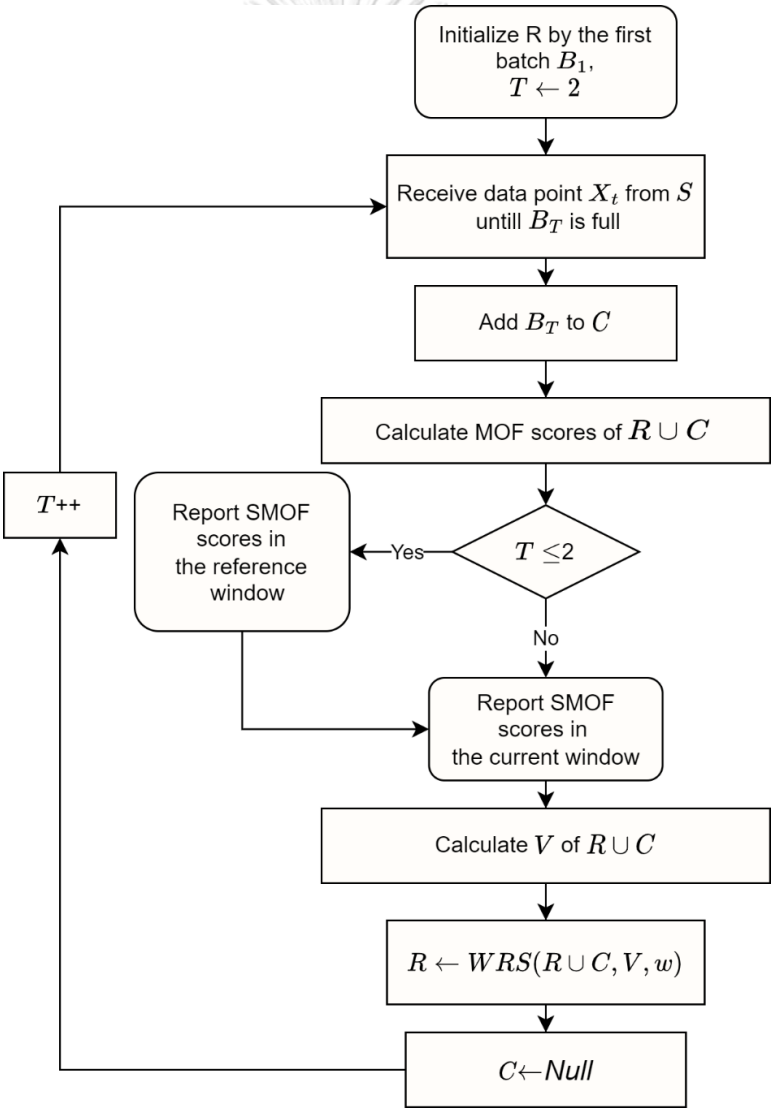



Figure 4.15 The SMOF flowchart

**Algorithm** $ProcessSMOF(w, S)$

**Inputs :** $w - Window\ Size, S - Streaming\ data$

**Outputs :** $SMOF_T - SMOF\ score\ of$ each $T^{th} batch$

**Line:**

1 : $R \leftarrow B_1$
2 : $T \leftarrow 2$
3 : ***while*** $Streaming\ data\ S\ continues\ do$
4 : $B_T \leftarrow B_T \cup X_t$
5 : ***if*** $|B_T| \geq w$ ***then***
6 : $C \leftarrow B_T$
7 : $MOF^r, MOF^c \leftarrow ProcessMOF(R \cup C)$
8 : ***if*** $T \leftarrow 2$ ***then***
9 : $\mu^r \leftarrow \frac{1}{w} \sum ProcessMOF(R)$
10 : $SMOF_1 \leftarrow MOF^r \cdot \mu^r$
11 : $Report\ the\ SMOF_1\ score\ of\ B_1$
12 : ***end if***
13 : $\mu^c \leftarrow \frac{1}{w} \sum ProcessMOF(C)$
14 : $SMOF_T \leftarrow MOF^c \cdot \mu^c$
15 : $Report\ SMOF_T\ score\ of\ B_T$
16 : $\mu^{rc} \leftarrow \frac{1}{2w} \sum ProcessMOF(R \cup C)$
17 : $V \leftarrow V^R \cup V^C$
18 : $R \leftarrow WRS(R \cup C, V, w)$
19 : $C \leftarrow null$
20 : $T++$
21 : ***end if***
22 : ***end while***



Figure 4.16 The pseudo code of the SMOF algorithm

## 4.2.4 The complexity analysis for the SMOF algorithm

The complexity of the SMOF algorithm has been divided into two parts 1) the outlier detection part and 2) the summarization part. The outlier detection part includes everything from the arrival of a data point to the returning outlier scores. The summarization part includes everything starting with calculating $V$ and removing all current data points. The time complexity of each part individually and the total complexity of the SMOF algorithm is the sum of the complexity of these two parts. The complexity of the SMOF algorithm is analyzed based on the amount of time. The SMOF algorithm would take to execute for one data point with respect to window size. Since the SMOF algorithm is executed for every data point and the number of data is potentially infinite, the analysis will not use to the number of data points. That means the time complexity of the SMOF algorithm is related to $w$. In the outlier

detection part, a new data point from streaming data is sent to a buffer until it reaches $w$ data points having an average time complexity $O(1)$. After that buffer sends a batch to $C$ and then calculates MOF scores. The MOF algorithm has time complexity $O(w^2 logw)$ and space complexity $O(w^2)$. This processes $2w$ data points in both windows leads to average time complexity $O(wlogw)$ and space complexity $O(w^2)$. The SMOF algorithm uses MOF scores dotted by the average $MOF$ scores to calculate the SMOF score having an average time complexity $O(1)$. In summary, the space and average time complexity of the outlier detection is $O(w^2)$ and $O(wlogw)$ respectively. In the summarization part, the SMOF algorithm calculates $V$. The MOF scores were already calculated in the first part; therefore, there is only an average time complexity of $O(1)$. After that, the SMOF algorithm uses WRS to sample $w$ data points from both windows according to $V$ by the sorted algorithm, which has time complexity $O(wlogw)$. Thus, it has an average time complexity $O(logw)$. The sampling is sent to the $R$. The current data points are dropped. The space complexity of this part contains $2w$ data points in both windows. In summary, the space and average time complexity of the summarization part is $O(w)$ and $O(logw)$, respectively. In conclusion, the sum of the average time complexity of the SMOF algorithm to measure SMOF scores and update each new data point is $O(wlogw)$, and the sum of space complexity is $O(w^2)$.

# Chapter 5
# Experimental Results

This chapter presents the experimental results of the MOF and the SMOF algorithms, evaluating their performance in terms of accuracy and execution time. The experiments are conducted using both real-world and synthesized data sets. It also presents the details of performance metrics, data sets, simulation models, and competitive algorithms.

## 5.1 Performance metrics

To evaluate the performance outlier detection, the accuracy of each studied algorithm is based on two performance metrics: Precision-Recall Area Under Curve (PR AUC) and Receiver Operator Characteristic Area Under Curve (ROC AUC). All performance metrics are based on the confusion matrix shown in Table 5.1, in which a true positive ($TP$) is a true outlier that is identified as an outlier, and a true negative ($TN$) is a normal data point that is identified as a normal data point, a false positive ($FP$) is a normal data point that is identified as an outlier, and a false negative ($FN$) is an outlier that is identified as a normal data point. A good outlier detection technique maximizes true positives while reducing false negatives and positives.

In the case of outlier detection, true negatives are ignored. The number of normal data points is significantly more than outliers. So, the number of true negatives is very high compared to the number of true positives. All cases of the data set evaluated in this chapter contain a lower 5% outlier.



Table 5.1 The confusion matrix

| | **Actual Outliers** | **Actual normal data points** |
|---|---|---|
| **Predicted Outliers** | TP | FP |
| **Predicted normal data points** | FN | TN |

### 5.1.1 Precision

Precision is defined as the ratio of correct identifications to total identifications. Where there is no mistaken classification, the highest achievable precision is 1. The notation is as follows:

$$Precision = \frac{TP}{TP + FP}$$

### 5.1.2 Recall

Recall is the ratio of the number of detected outliers to the total number of outliers. In reality, precision just demonstrates the correctness of the results; it does not represent the completeness of the results, so precision is always combined with a recall to illustrate the correctness and completeness of an algorithm. The highest achievable recall is 1. The notation is as follows:

$$Recall = \frac{TP}{TP + FN}$$

### 5.1.3 ROC AUC (AUC)

Precision and Recall based on the confusion matrix can only evaluate the performance of a binary score. Although an outlier score can set a threshold to convert to a binary score, it is not comfortable to find the best threshold. AUC ROC curve is a performance measurement for classification problems at various threshold settings. ROC is a probability curve, and AUC represents the degree or measure of separability. It explains how capable the model is capable of distinguishing between outliers and normal data points. An excellent model has an AUC close to one, indicating that it has a high level of separability. A poor model has an AUC close to zero, indicating that it has the weakest measure of separability. In fact, it means that it is reversing the result. Outliers are predicted as normal data points, and normal data points are predicted as outliers. When AUC is 0.5, the model has no class separation capacity at all. The highest AUC is one in which algorithms can identify separability between outliers and normal data points, while the lowest AUC is 0, in which algorithms can reverse identify separability between them. In fact, it means that it is reversing the result. Outliers are predicted as normal data points, and normal data points are predicted as outliers. While the average AUC is 0.5, algorithms cannot identify separability between outliers and normal data points.

Figure 5.1 Receiver operating characteristic curveshows the ROC curve with True positive ($TP$) against the False positive ($FP$) where $FP$ and $TP$ are on-axis $x$ and $y$ respectively. The orange curve is ROC, and the green area under the orange curve is AUC.

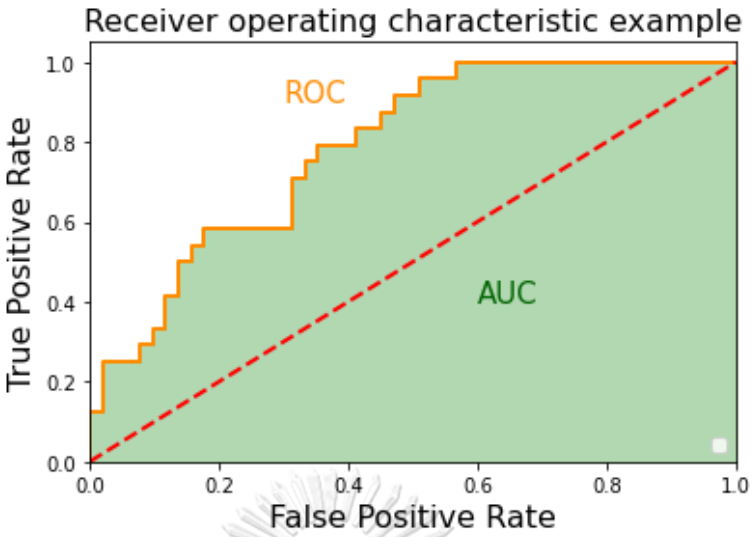



Figure 5.1 Receiver operating characteristic curve

To investigate the use of the area under the ROC curve (AUC) as a measure of classifier performance. When the decision threshold ($i$) is varied, and the number of points on the ROC curve $[TPR = a, FPR = \beta]$ which have been obtained, the simplest way to calculate the area under the ROC curve is to use trapezoidal integration. The notations are as follows.

$$AUC = \sum_{i=1}^{n} \{(\beta \cdot \Delta\alpha) + \frac{1}{2}[\Delta\beta \cdot \alpha]\}$$

$$where\ \Delta\beta = \beta_i - \beta_{i-1}$$

$$\Delta\alpha = \alpha_i - \alpha_{i-1}$$



## 5.1.4 Average Precision (AP)

The Precision-Recall AUC is similar to the ROC AUC in that it summarizes the curve as a single score with a range ($n$) of threshold $i$ values. Thresholds are used to convert outlier scores into precision and recall on-axis $y$ and $x$, according to Figure 5.2. The weighted mean of precision acquired at each threshold is used to describe a precision-recall curve as average precision (AP).

$$AP = \sum_{i \in n} (R_i - R_i - 1)P_i$$

where $P_i$ and $R_i$ are the precision and recall at the $i^{th}$ threshold that has been obtained, the easiest way to calculate the PA is to use the trapezoidal integration.

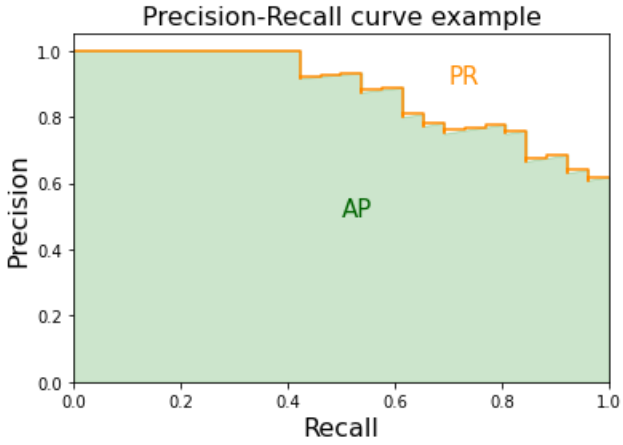



Figure 5.2 Precision-Recall curves

### 5.1.5 A paired t-test

In fact, each outlier detection algorithm has a different idea to detect outliers. To compare the pattern of detecting outliers between the two algorithms. A paired t-test is used when the difference between two algorithms is interested in the same data set. Supposed $n$ data sets were given outlier scores by algorithms A and B. The paired t-test is used to prove that algorithm B leads to improvements or deterioration in detecting outliers compared with algorithm A. The score of each data set can be calculated by recall or precision. Let $X = \{x_1, x_2, \dots, x_n\}$ be a subset of metric scores from algorithm A and $Y = \{y_1, y_2, \dots, y_n\}$ be a subset of metric scores from algorithm B. To test the null hypothesis that the true mean difference is zero, the procedure is as follows:

1. Calculate the difference $\{d_i | \, d_i = y_i - x_i \; where \; i = \{1,2, \dots, n\}\}$
2. Calculate the mean different $\bar{d} = \frac{\sum_{i=1}^{n} d_i}{n}$
3. Calculate the standard deviation of the difference, $s_d$, and use this to calculate the standard error of mean difference, $SE(\bar{d}) = \frac{s_d}{\sqrt{n}}$
4. Calculate the t-statistic, which is given by $T = \frac{\bar{d}}{SE(\bar{d})}$. Under the null hypothesis, this statistic follows a t-distribution with $n - 1$ degree of freedom.
5. Use tables of the t-distribution to compare your value for $T$ to the $n-1$ distribution. This will give the $p$-value for the paired t-test.
6. The $p$-value is the probability of obtaining test results at least as extreme as the results observed, under the assumption that the null hypothesis is correct. A $p$-value larger than a chosen threshold (e.g., 5% or 1%) indicates that our observation is not so unlikely to have occurred by chance. Moreover, it rejects the null hypothesis of equal population means if the $p$-value is smaller than the threshold.

## 5.2 Details and Parallel coordinates plots of benchmark data sets

Benchmark data sets include Shuttle, Glass, Saimage-2, Satellite, [43], HTTP, and SMTP data, which is KDD CUP 99 network intrusion data as used in [44], and Mulcross [45] which all categorical attributes are removed. In addition, this section also shows parallel coordinates plots of each data set in which axis *x* is a dimension, and axis *y* is the value of each dimension. The last dimension is a binary label in which the yellow and blue lines are outliers and normal data points, respectively. Table 5.2 provides the properties of all data sets, where $\#n$ is the number of data points, $\#Dim$ is the number of dimensions whose index of dimensions starts from zero, and the percentage in brackets indicates the percentage of outliers.

Table 5.2 Data properties

| **Data sets** | $\#n$ | $\#Dim$ | **#Outliers (%)** |
|---|---|---|---|
| Glass | 214 | 9 | 9 (4.2%) |
| Musk | 3062 | 166 | 97 (3.2%) |
| Satimage-2 | 5803 | 36 | 71 (1.2%) |
| Satellite | 6435 | 36 | 2036 (32%) |
| MNIST | 7603 | 100 | 700 (9.2%) |
| Ionosphere | 351 | 33 | 126 (36%) |
| Shuttle | 49097 | 9 | 3511 (7%) |
| HTTP | 567479 | 3 | 2211 (0.4%) |
| Satimage-2 | 5803 | 36 | 71 (1.2%) |
| Mulcross | 262144 | 4 | 26214 (10%) |
| SMTP | 95156 | 3 | 30 (0.03%) |



### 5.2.1 Satimage-2 data

Satimage-2 data: Statlog (Landsat Satellite) data set is a multi-class classification data set that contains 7 classes. Class 2 is down sampled to 71 data points as an outlier, and other classes are pooled to produce normal data points. The revised data set has been dubbed Satimage-2. Figure 5.3 shows that the values of all dimensions of most outlier and normal data points are stratified. In addition, they are dense masses that are closely related to each other.

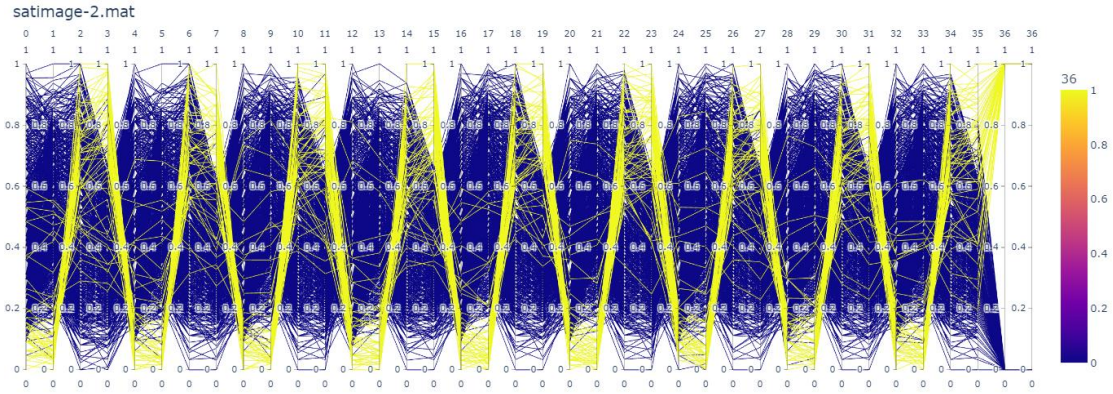



Figure 5.3 Parallel coordinates plot for Satimage-2 data

## 5.2.2 MNIST data

MNIST data: the original data's MNIST database (Modified National Institute of Standards and Technology database) is a massive collection of handwritten digits that is frequently used to train image processing systems that contain 28×28 pixels or 784 features. Seven hundred images are sampled from the digit-six class as the outliers, and all digit-zero classes are considered normal data points; in addition, 100 features are randomly selected from 784 total features. Figure 5.4, Figure 5.5, and Figure 5.6 show that the values of most dimensions of most outlier and normal data points are unclearly stratified.

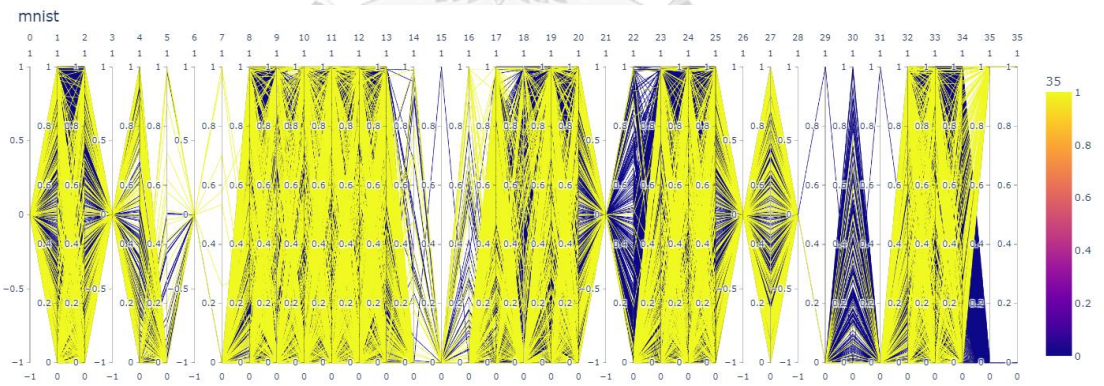



Figure 5.4 Parallel coordinates plot for MNIST data in 0-34th dimensions

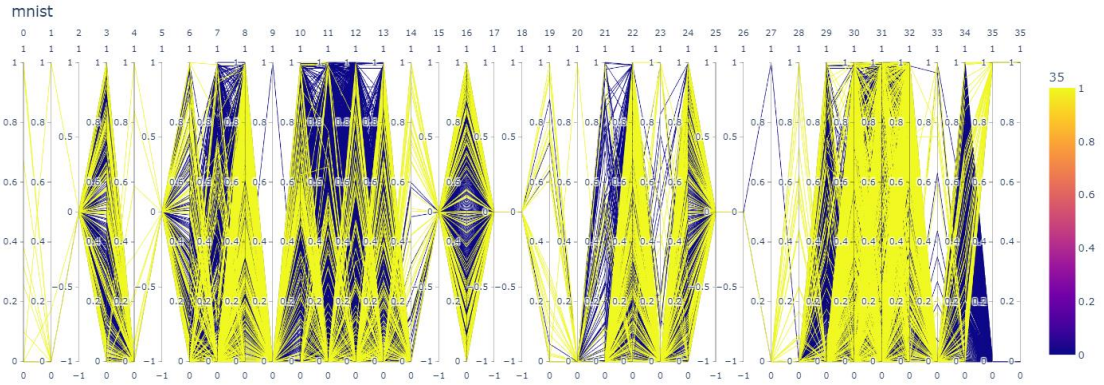



Figure 5.5 Parallel coordinates plot for MNIST data in 35-69th dimensions

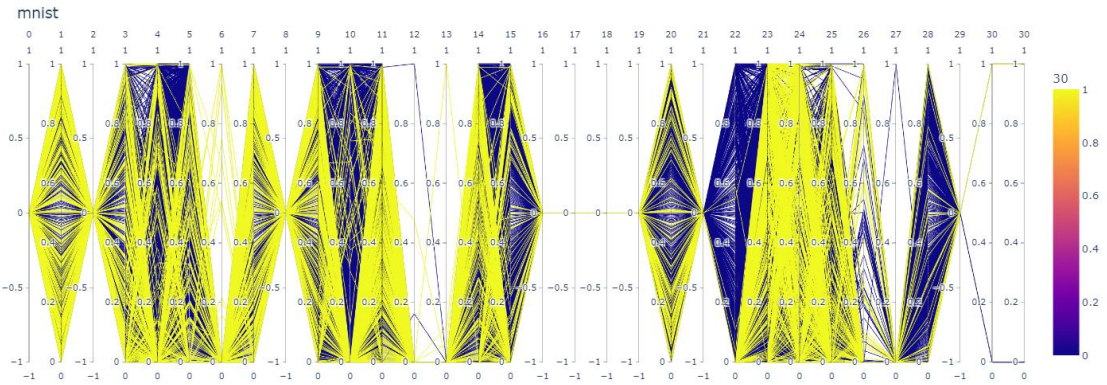



Figure 5.6 Parallel coordinates plot for MNIST data in 70-99th dimensions

## 5.2.3 Ionosphere data

Ionosphere data: The original ionosphere data set was radar data collected by a system in Goose Bay. It has a dimensionality of 34 and is a binary classification data set. A "good" label indicates that radar detects some type of structure in the ionosphere, while a "bad" indicates that radar does not detect any; their signals pass through the ionosphere. The “Good” and “Bad” labels are classified as normal data points and outliers. Moreover, one feature has 0 values and is thus discarded. As a result, the total number of features is 33. Figure 5.7 shows that the values of most dimensions of most outliers and normal data points are unclearly stratified. In addition, outliers are scattered, while the normal data points are grouped together.

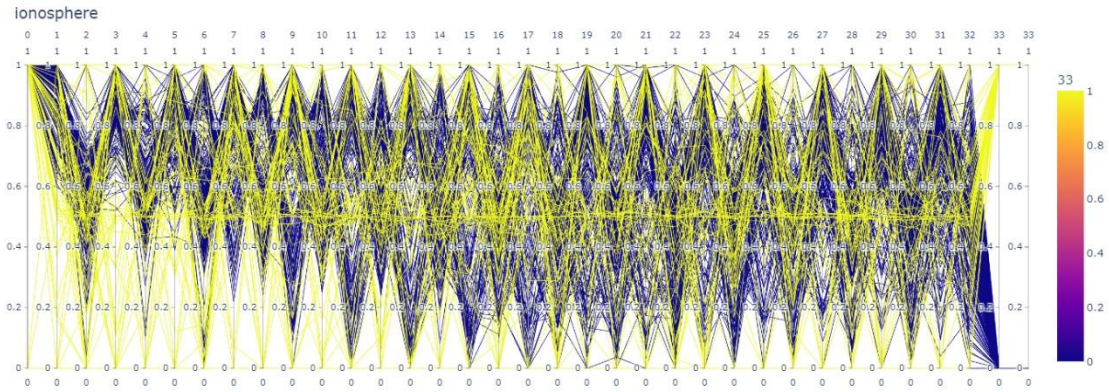



Figure 5.7 Parallel coordinates plot for the Ionosphere data

## 5.2.4 Musk data

Musk data: the data set contains several musk and non-musk classes. No-musk classes j146, j147, and 252 are combined as normal data points, while musk classes 213 and 211 are combined as an outlier. Note those other classes have been dropped. Figure 5.8, Figure 5.9, Figure 5.10, and Figure 5.11 show that the values of most dimensions of most outlier and normal data points are unclearly stratified. In addition, the outlier's values on some dimensions agglomerated.

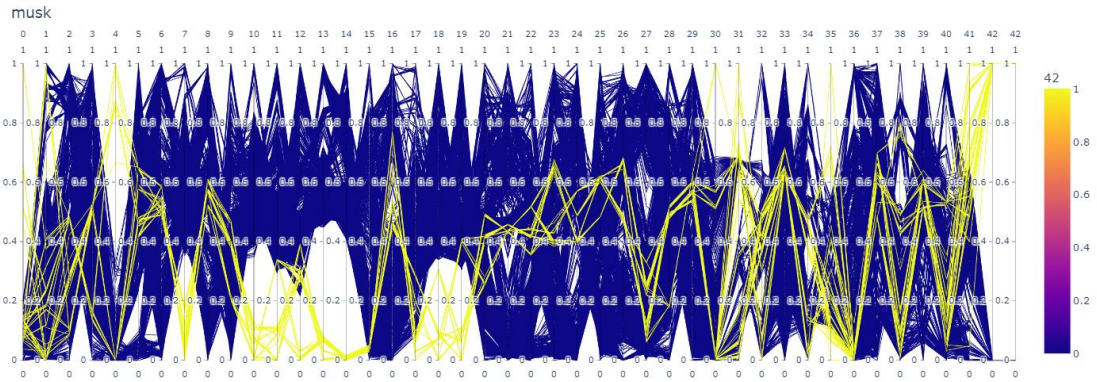



Figure 5.8 Parallel coordinates plot for Musk data in 0-41th dimensions

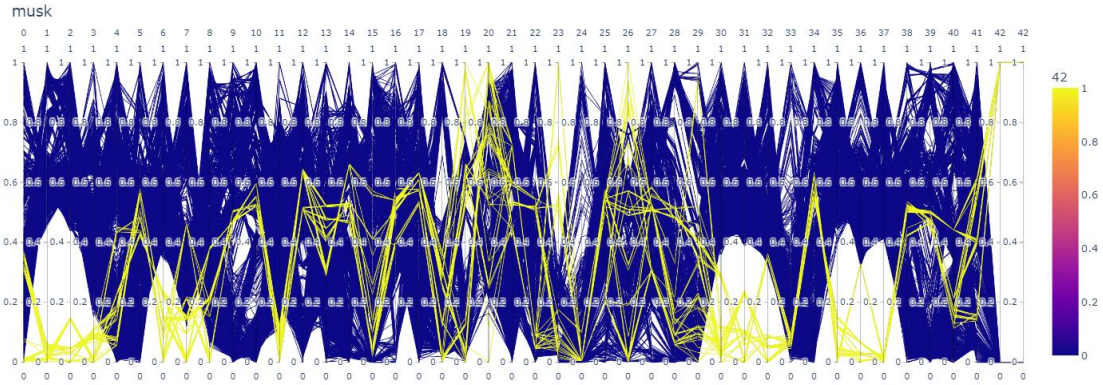



Figure 5.9 Parallel coordinates plot for Musk data in 42-83th dimensions

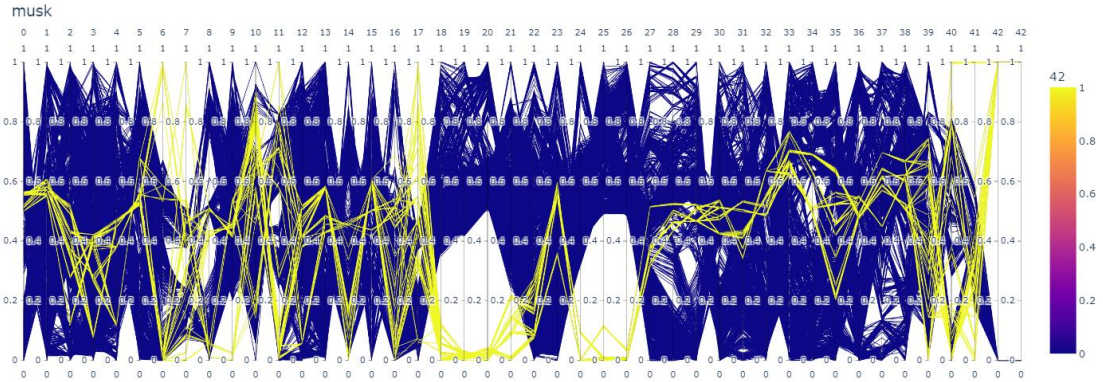



Figure 5.10 Parallel coordinates plot for Musk data in 84-125th dimensions

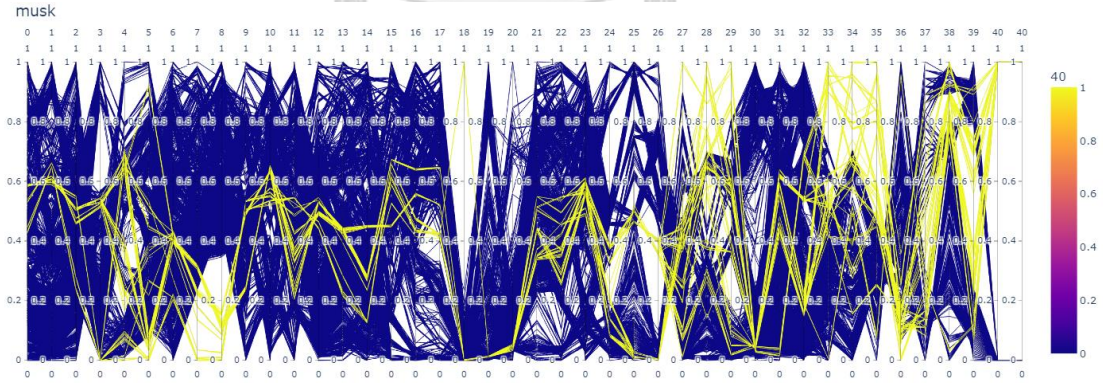



Figure 5.11 Parallel coordinates plot for Musk data in 126-165th dimensions

## 5.2.5 Satellite data

Satellite data: Statlog (Landsat Satellite) data set is a multi-class classification data set that contains 7 classes. Classes 2, 4, and 5 are combined as outliers, while the other classes are combined as normal data points. The revised data set has been

dubbed Satellite. Figure 5.12 shows that the values of the minor dimensions of normal data points and outliers are stratified, while most normal data points and outliers have high overlapping values. In addition, both of them are closely clustered together.

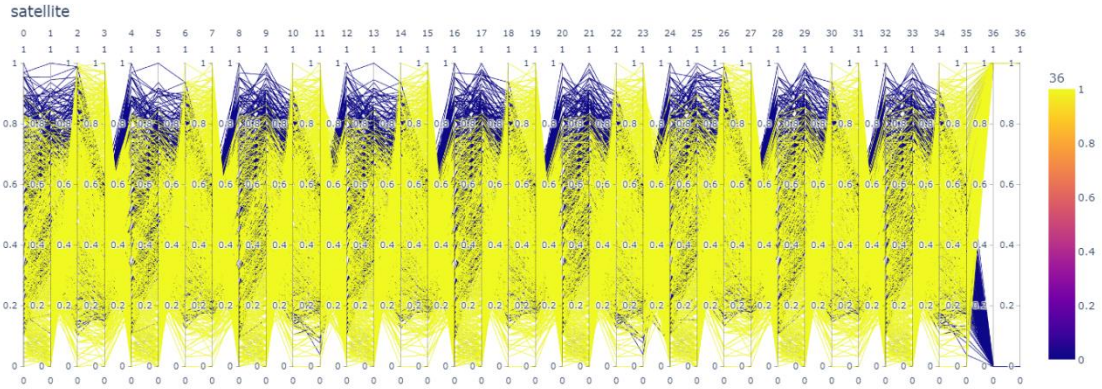



Figure 5.12 Parallel coordinates plot for Satellite data

## 5.2.6 Glass data

Glass data: The data set contains seven types of glass. The number of data points in class 6 is the lowest among other classes, so class 6 is an outlier, while other classes are normal data points. Note each feature of this data includes the ingredient element and the refractive index. Figure 5.13 shows that the values of all dimensions of all normal data points and outliers are not very stratified.

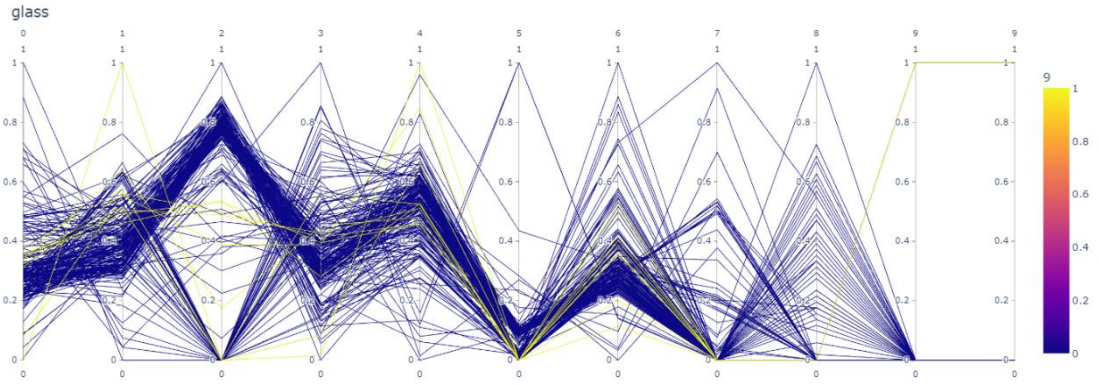



Figure 5.13 Parallel coordinates plot for Glass data

## 5.2.7 HTTP and SMTP data

HTTP, SMTP, and SMTP+HTTP data: The original KDD Cup 1999 data set contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. It contains 3,925,651 attacks (80.1%) out of 4,898,431 data points, and 41 attributes (34 continuous, and 7

categorical). However, this data set keeps only the attribute “logged_in” positive, resulting in 3,377 attacks (0.35%) of 976,157 data points. However, there are still categorical and useless attributes; therefore, they are reduced to 4 attributes: Duration, src_bytes, and dst_bytes continuous attributes which are concentrated around 0. Therefore, they transform each value into a log scale by $y = \log(x + 0.1)$. After that, three smaller data sets are forged according to the service attribute. Firstly, SMTP data contains only the "SMTP" service attribute. Secondly, HTTP data contains only the "HTTP" service attribute. Lastly, SMTP+HTTP data contains sequence data of SMTP and HTTP data, respectively. Figure 5.14 and Figure 5.15 show that the values of the first dimension (src_bytes) of all normal data points and outliers are stratified. The zero and the second dimensions are noise dimensions.

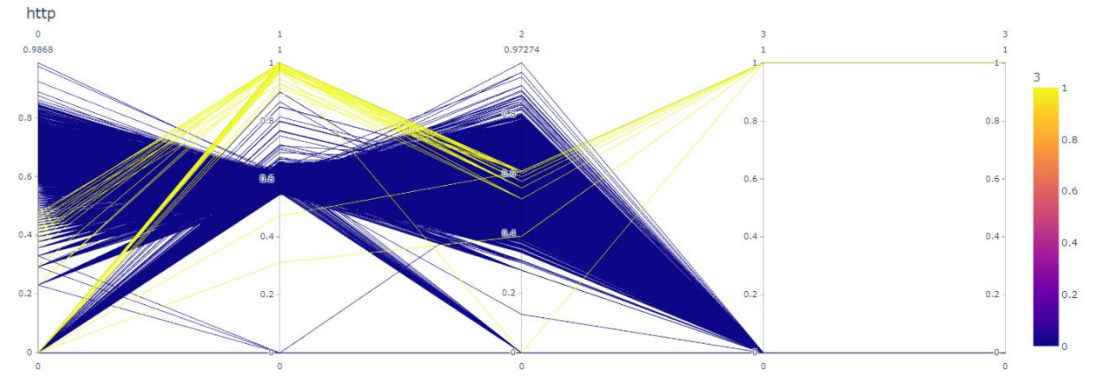



Figure 5.14 Parallel coordinates plot for HTTP data

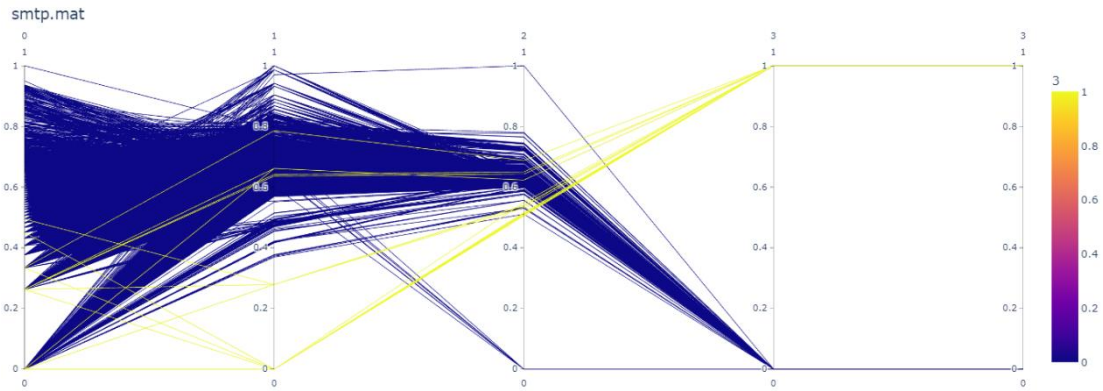



Figure 5.15 Parallel coordinates plot for SMTP data

## 5.2.8 Shuttle data

Shuttle data: Statlog (Landsat Satellite) data set contains 7 classes. Classes 2, 3, 5, 6, and 7 are combined as outliers, while class 1 is a normal data point. Class 4 is discarded. The revised data set has been dubbed Shuttle data. Figure 5.16 shows that the values of the zero and first dimensions of the outlier are spread, while normal data

points are dense. The values of other dimensions of outliers and normal data points have high overlapping values. In addition, both of them are closely clustered together.

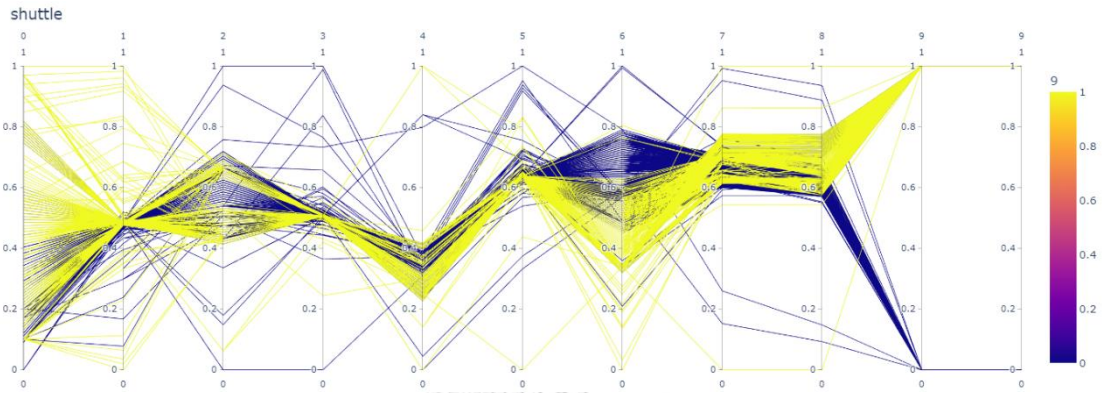



Figure 5.16 Parallel coordinates plot for the Shuttle data

### 5.2.9 Mulcross data

Mulcross data: synthesize data is generated by a multidimensional normal distribution that contains two clusters of outliers that are far from the center of the normal cluster by 2 distance factors. Figure 5.17 shows that the values of the zero and first dimensions are noise, while the values of other dimensions of outlier and normal data points are stratified. In addition, outlier and normal data points are dense masses that are closely related to each other.

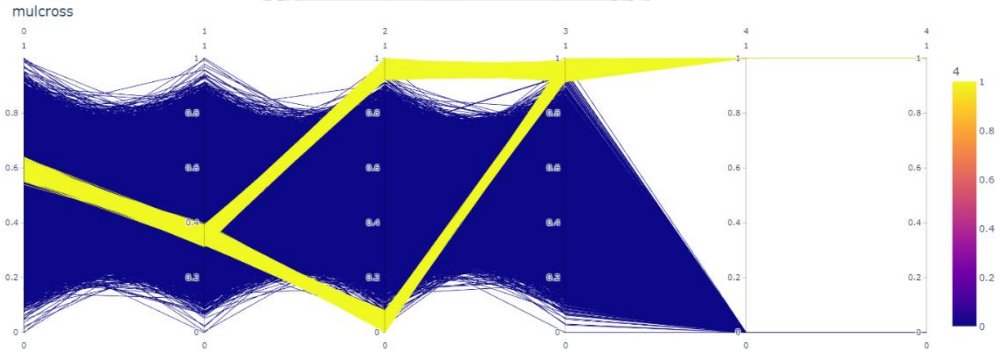



Figure 5.17 Parallel coordinates plot for the Mulcross data

## 5.3 Simulation Model

The simulation goal is to demonstrate the effectiveness of SMOF, MOF, and other state-of-the-art algorithms. Static data is used to calculate their outlier scores, whereas streaming data is imitated. It generates each data point incrementally to the model over a fixed time interval. The entire simulation model is built on Google Colab and runs in Python. Note that all data sets are scaled by min-max scale before

they are processed. The SMOF and the HS-Trees algorithms use random methods. Therefore, the performance results are from an average of 10 trials.

## 5.4 Competitive algorithms

The MOF algorithm is compared to six other algorithms: LOF [41], OOF [22], AOF[24], FastABOD [25], and kNN, all of which are discussed in detail in Chapter 2. LOF, kNN, and FastABOD have a single parameter ($k$-nearest neighbors), whereas OOF and AOF are parameter-free. The LOF, kNN, and FastABOD algorithms are imported from Pyod [46] while the OOF, AOF, and MOF algorithms are coded by the author and accelerated by Numba [47] which converts Python functions to optimized machine code that can approach C speeds.

The SMOF algorithm is compared to the HS-Trees algorithm, the details of which are already presented in Chapter 2. The HS-Trees parameters include the number of HS-Trees in an ensemble ($\mathcal{E}$), the maximum depth ($h$), $sizeLimit$, and window sizes ($w$).

Note: HS-Trees and FastABOD assign low outlier scores as an outlier. The outlier score of them is inverted for comparing AUC and AP metrics with other state-of-the-art outlier detection techniques.

## 5.5 Experiments and results of the MOF algorithm

This section evaluates the MOF algorithm with extensive experiments. All examples are used to illustrate some specific data points that appear to be meaningful, but cannot be identified by other methods. The first example starts with 2d synthesized data sets, for which the outlier scores are colored. The second example uses the paired t-test to evaluate the detection pattern. The third example is from real-world data sets and evaluated performance by AUC, AP, and execute time metrics. In the last experiment, the top-10 outliers in modified real-world data sets are extracted.

### 5.5.1 Visualization of outlier scores

The purpose of this section is to compare the distribution of MOF scores to $OOF$ and AOF scores on six 2d synthesized data sets. The OOF and the AOF algorithms are also unsupervised parameter-free outliers scoring the same as MOF. Figure 5.18 depicts that they are on a scatter plot. The color is the degree of outlier score by each algorithm, with blue indicating a low outlier score and red indicating a high outlier score (according to rainbow colors). The experiments yielded the following results:

- The data sets 1) and 2) contain local and global outliers. MOF, OOF, and AOF can assign outlier scores to global outliers far from normal regions, significantly greater than normal data points. However, OOF and AOF cannot detect local

outliers at the center as normal data points, while the MOF algorithm can detect them correctly.

- The data set 3) contains three normal region clusters generated by a gaussian distribution with a difference in mean and variance. The MOF algorithm balances MOF scores across all clusters, whereas the OOF and the AOF algorithms assign the highest outlier scores to the highest sparse cluster (middle cluster).
- The data set 4) is made up of three ellipse clusters generated by a multidimensional gaussian distribution, each with a different mean but the same variance. The MOF algorithm can detect outliers correctly. In contrast, OOF and AOF fail to detect outliers around clusters
- The data sets 5) and 6) only contain global outliers from the normal region. OOF, AOF, and MOF can significantly assign outlier scores to outliers greater than all normal regions.

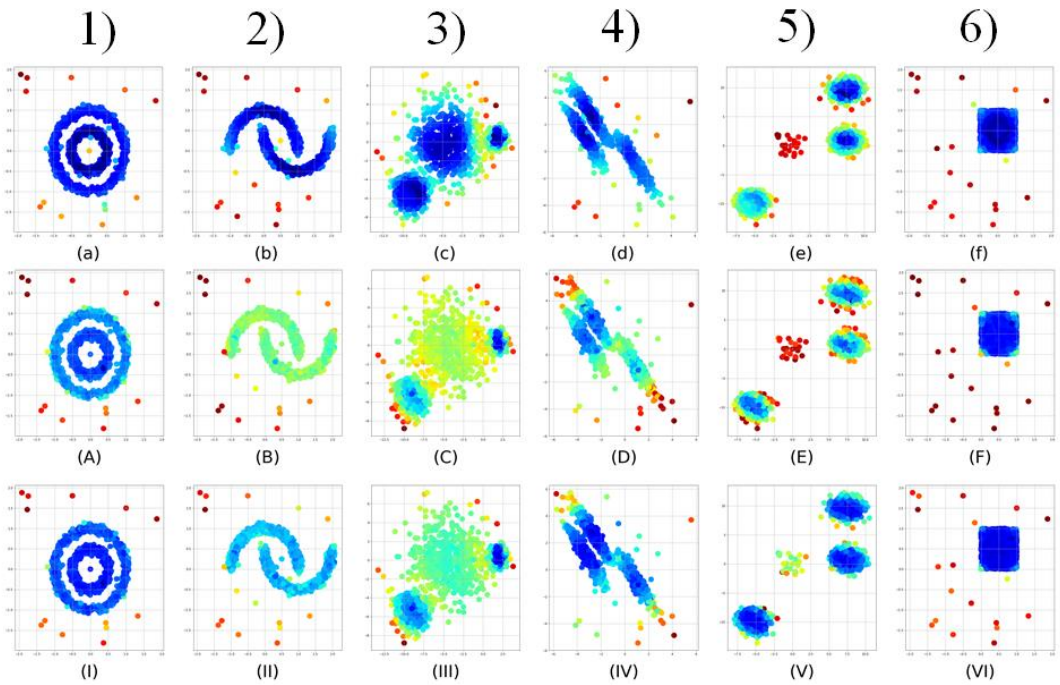



Figure 5.18 Assigning the log-scale on MOF (a – f), OOF (A-F), and (I-VI) scores to the six 2d synthesized data sets (1-6)

The results of these experiments show that the MOF algorithm can detect local and global outliers in single or multiple normal region clusters with different densities, whereas AOF and OOF cannot. However, OOF and AOF can still detect global outliers in single or multiple normal regional clusters with the same densities. This is because the above data set necessitates the principle of data density-based rather than distance-based.

### 5.5.2 Comparison of detection patterns by paired t-test

The MOF algorithm is inspired by many other algorithmic results to cover conceptual bases such as LOF is the density-based outlier detection, AOF assigns an outlier score based on other data points, and FastABOD uses variance to summarize the outlier score. Therefore, this experiment aims to compare detection patterns of MOF to LOF, AOF, and FastABOD by paired t-test of precision and recall metrics. To implement, the data sets will be generated to have 1-cluster, 2-cluster, and 3-cluster patterns. Each cluster pattern will be generated for 100 data sets, and each data set will have two numeric attributes with 1,000 data points. From 1,000 data points, 990 data points will follow the gaussian distribution having random centroids and variances. Note centroids of the gaussian distribution and the other 10 data points, which are generated randomly within a box [-10,10] × [-10,10]. The example 2 data set from each cluster pattern is shown in Figure 5.19.

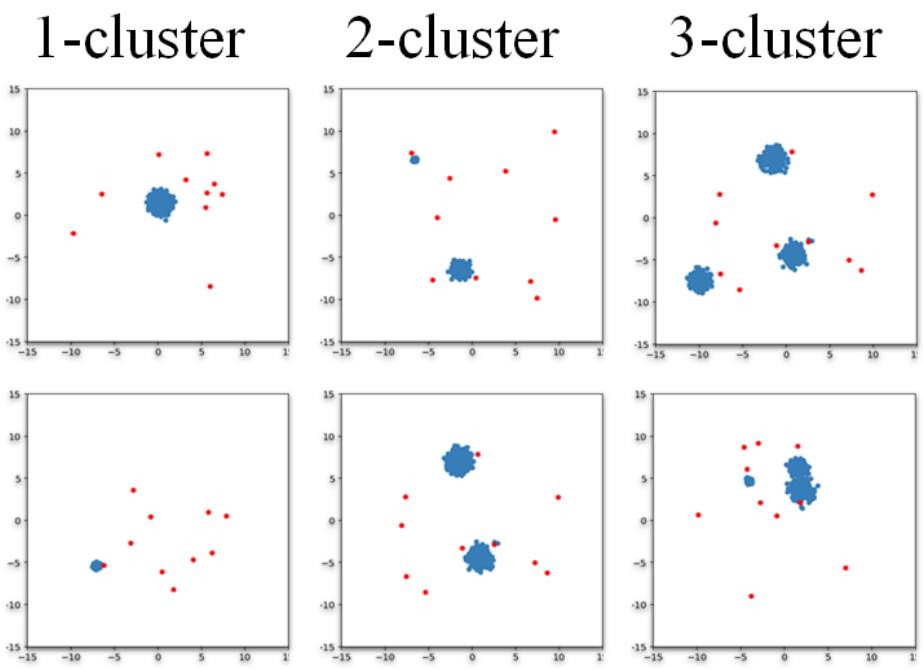



Figure 5.19 One, two, and three clusters pattern of 2d synthesized data sets

After that, the MOF and other algorithms measure outlier scores on all 100 data sets and return the top-10 highest scores from each algorithm. They will change to precision and recall metrics which is the perfect score of 1. It intends to utilize the paired t-test to establish the statistical significance of precision and recall from all data sets of the MOF to AOF, FastABOD, and LOF. The significance level in this experiment is set to 0.05. If the $p$-value is less than or equal to 0.05, then the null hypothesis will be rejected. In contrast, if the $p$-value is greater than 0.05, then the null hypothesis cannot be rejected. This means that the MOF algorithm obtains a

similar performance to the considered algorithm. The results of $p$-values are shown in Table 5.1. The underlined numbers are marked for a $p$-value lower than 0.05.

Table 5.3 shows that the MOF algorithm has significantly different recall and precision compared with AOF since all cluster patterns have $p$-values less than 0.05. In contrast, The MOF algorithm does not show significantly different recall and precision compared with FastABOD and LOF. All cluster patterns have $p$-values greater than 0.05.

Table 5.3 $p$-value for each cluster pattern and algorithm

| **Cluster** | ***p*-value** | | | | | |
|---|---|---|---|---|---|---|
| **patterns** | **AOF** | | **FastABOD** | | **LOF** | |
| | **Recall** | **Precision** | **Recall** | **Precision** | **Recall** | **Precision** |
| 1 | **<u>0.045</u>** | **<u>0.045</u>** | 0.320 | 0.320 | 0.566 | 0.566 |
| 2 | **<u>0.000</u>** | **<u>0.000</u>** | 0.103 | 0.103 | 0.083 | 0.083 |
| 3 | **<u>0.000</u>** | **<u>0.000</u>** | 0.241 | 0.241 | 0.250 | 0.250 |

The results of these experiments show that the detection pattern of the MOF algorithm is the same as the FastABOD and LOF algorithm but different from the AOF algorithm. The main reasons are the following:

- Both the MOF and the LOF algorithms can detect outliers in multiple or single clusters (Gaussian distribution) with different densities (variance).
- Both MOF and FastABOD use the variance of contribution data points to summarize the outlier score, which can distinguish between outlier and normal data points.
- Although both MOF and AOF are parameters-free, AOF is distance-based, order difference distance which has an outlier scoring idea that is very different from MOF.



## 5.5.3 Experiments on real-world data sets

From the past two experiments, the MOF algorithm can detect outliers in the synthesized data set. Therefore, it has the potential to detect outliers in real-world data sets. This experiment aims to compare performance among MOF, LOF, OOF, and kNN in real-world data sets such as Glass, Ionosphere, MNIST, Musk, Satellite, and Satimage-2 (detail in section 5.2). Since it is hard to know the real outliers in the data set, all parameters of all algorithms must be set to the default value to detect outliers in all data sets. The MOF and the OOF algorithms have no parameters to configure, whereas LOF and kNN have $k$-nearest neighbors set to 20 and 5, respectively. The performance metrics to evaluate are AUC, AP, and execution time. The experimental

results are shown in Table 5.4, Table 5.5, and Table 5.6 ordered by rank. Rank 1 shows the best and rank 4 shows the worst. If the algorithm's performance is the same as the other model, ranks are shared. The underline in Table 5.4 and Table 5.5 indicated that they had the highest score, while Table 5.6 indicated that they had the shortest execution time. In addition, all tables show the average value and rank of performance for a summary. The following are the performance results.

Table 5.4 and Table 5.5 show that the MOF algorithm outperforms other algorithms in terms of AUC and AP scores on the Glass, MNIST, Musk, and Satimage-2 data sets. This is because these data sets contain dense outliers according to the parallel coordinates plot in section 5.2, in which most dimensions of the outlier are dense, which is larger than $k$ data points in the parameters of the LOF and the kNN algorithms. In addition, the AOF algorithm cannot detect micro-outliers.

However, parallel coordinates plots of Satellite and Ionosphere are shown that each dimension of the outlier is high spread. Therefore, the kNN algorithm outperforms in Ionosphere data, and OOF outperforms in Satellite. The outlier score in this data set should be from the distance of the nearest neighbors.

One interesting point is that the LOF algorithm has the worst performance accuracy. This is because the $k$-nearest neighbors are very sensitive. The $k$-nearest neighbors must be based on the number of micro-clusters to detect outliers. You can see that AUC and AP of LOF are worse than kNN in the Satimage-2 data set, in which outliers are very dense compared to other data sets. Although $k$-nearest neighbors in LOF are greater than $k$-nearest neighbors in kNN, AP of LOF is lower than six times AP of kNN.

Table 5.6 shows that the MOF algorithm takes execution time (seconds) longer than LOF and kNN but faster than OOF because LOF and kNN use $k$-nearest neighborhoods to measure outlier scores in the local area ($k$ data points). In contrast, both the OOF and MOF algorithms measure outlier scores in relation to all data points ($n$ data points) in a data set. Moreover, kNN and LOF have a time complexity of $O(n^2)$, whereas the OOF and MOF algorithms share a time complexity of $O(n^2 logn)$. The OOF algorithm should be the same as the MOF algorithm because it has steps to calculate lower than MOF. The source code of the OOF algorithm has shared some parts with the AOF algorithm. Note: All of these algorithms use the same Euclidean distance measurement; number dimensions do not play a role in determining time complexity.

Overall performance by average performance and rank can be summarized as follows:

- They showed that the accuracy of the MOF algorithm outperforms other algorithms while the LOF underperforms other algorithms.
- For execution time, the LOF and the kNN algorithms are faster execution times than the MOF and OOF algorithms.

Table 5.4 AUC scores for static data

| Data sets | MOF | | OOF | | LOF | | kNN | |
|---|---|---|---|---|---|---|---|---|
| | Rank | AUC | Rank | AUC | Rank | AUC | Rank | AUC |
| Glass | **1** | **0.850** | 4 | 0.824 | 3 | 0.831 | 2 | 0.84 |
| Ionosphere | 2 | 0.920 | 4 | 0.718 | 3 | 0.895 | **1** | **0.933** |
| MNIST | **1** | **0.871** | 3 | 0.771 | 4 | 0.679 | 2 | 0.841 |
| Musk | **1** | **1.000** | 2 | 0.811 | 4 | 0.637 | 3 | 0.759 |
| Satellite | 2 | 0.696 | **1** | **0.782** | 4 | 0.546 | 3 | 0.672 |
| Satimage-2 | **1** | **0.997** | 2 | 0.965 | 4 | 0.542 | 3 | 0.938 |
| **Average** | **1.33** | **0.889** | 2.67 | 0.812 | 3.67 | 0.688 | 2.33 | 0.831 |

Table 5.5 AP scores for static data

| Data sets | MOF | | OOF | | LOF | | kNN | |
|---|---|---|---|---|---|---|---|---|
| | Rank | AP | Rank | AP | Rank | AP | Rank | AP |
| Glass | **1** | **0.222** | 2 | 0.111 | 2 | 0.111 | 2 | 0.111 |
| Ionosphere | 2 | 0.810 | 4 | 0.5 | 3 | 0.770 | **1** | **0.881** |
| MNIST | **1** | **0.429** | 3 | 0.39 | 4 | 0.294 | 2 | 0.424 |
| Musk | **1** | **1.000** | 2 | 0.598 | 3 | 0.258 | 4 | 0.237 |
| Satellite | 2 | 0.552 | **1** | **0.61** | 4 | 0.376 | 3 | 0.492 |
| Satimage-2 | **1** | **0.747** | 2 | 0.732 | 4 | 0.07 | 3 | 0.394 |
| **Average** | **1.33** | **0.627** | 2.33 | 0.49 | 3.33 | 0.313 | 2.5 | 0.423 |



Table 5.6 Execution time for static data

| Data sets | MOF | | OOF | | LOF | | kNN | |
|---|---|---|---|---|---|---|---|---|
| | Rank | Time | Rank | Time | Rank | Time | Rank | Time |
| Glass | 2 | 0.009 | 4 | 1.849 | 3 | 0.009 | **1** | **0.004** |
| Ionosphere | 4 | 0.035 | 3 | 0.031 | 2 | 0.018 | **1** | **0.008** |
| MNIST | 3 | 21.547 | 4 | 21.950 | **1** | **1.979** | 2 | 1.989 |
| Musk | 3 | 2.828 | 4 | 3.346 | **1** | **0.315** | 2 | 0.346 |
| Satellite | 3 | 10.919 | 4 | 14.523 | **1** | **1.043** | 2 | 1.072 |
| Satimage-2 | 3 | 8.406 | 4 | 10.444 | 4 | 0.89 | **1** | **0.879** |
| **Average** | 3 | 7.290 | 3.83 | 8.691 | 1.67 | 0.709 | **1.5** | **0.716** |

### 5.5.4 The top-10 outliers

This section aims to evaluate outlier detections such as MOF, OOF, FastABOD, and LOF on three real-world data sets, Ionosphere, Musk, and Satimage-2 data set, comparing only the top-10 highest scores extracted with their indices. From the previous section, the MOF algorithm can detect dense outliers perfectly. Therefore, the outliers of each data set are reduced to just ten data points randomly to evaluate real-world data sets which contain few outliers. In addition, the parameters for LOF vary from 10 to 40 and return the best tune by grid search. The AOF and the MOF algorithms do not require any parameter, while the parameter of the FastABOD algorithm is set to 0.1 of the real-world data sizes. The reported results are the ones with the maximum number of large indices from each real-world data set and the number of outliers that are detected.

Three real-world data sets are the Ionosphere data set, the Musk data set, and the Satimage-2 data set, as shown in Table 5.7. First, the Ionosphere data set has 225 normal data points and 126 outliers. It has 33 dimensions. Second, the Musk data set contains several musk and non-musk classes. This data set contains 97 musk and 2,965 non-musk classes with 166 dimensions. Lastly, the Satimage-2 contains class 2 as 71 outliers and 5,732 as normal classes with 36 dimensions. In the experiment, only the first 10 outliers were kept.

Table 5.7 Three real-world data sets

| Data sets | Non-outlier Indices | Outlier Indices | #Dim |
|---|---|---|---|
| Ionosphere | 0 - 234 | 235 - 244 | 33 |
| Musk | 0 - 2,964 | 2,965 - 2,974 | 166 |
| Satimage-2 | 0 - 5,731 | 5,732 - 5,741 | 36 |



The results shown in Table 5.8 are the top-10 ranks and the number of each algorithm that detects outliers correctly. The parameter $k$ of LOF is determined to be 28, 21, and 22 for the Ionosphere data set, the Musk data set, and the Satimage-2 data set, respectively. In the Ionosphere data set, the MOF algorithm can detect more outliers than AOF but detect the same number of outliers as FastABOD and LOF. In addition, No. 61 and No. 17 are detected by MOF, FastABOD, and LOF as outliers. In the Musk data set, the MOF and the LOF algorithms can detect all outliers, but the FastABOD algorithm can only detect No. 2965 while the AOF algorithm can detect No. 2965 and No. 2967. The last experiment was on the Satimage-2 data set. Both the MOF and the FastABOD algorithms can detect the top-8 outliers correctly and identify No. 3366 as an outlier.

Table 5.8 The outlier scores of each algorithm for three real-world data sets

| **Ionosphere data set** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Top-10** | **MOF** | | **AOF** | | **FastABOD** | | **LOF** | |
| | **Index** | **Score** | **Index** | **Score** | **Index** | **Score** | **Index** | **Score** |
| 1 | **233** | 125.12 | **233** | 0.54 | **233** | -5.42E-06 | **230** | 3.69 |
| 2 | **234** | 104.49 | 26 | 0.17 | **234** | -1.30E-05 | **231** | 3.39 |
| 3 | **226** | 71.57 | 94 | 0.16 | **230** | -3.46E-05 | **234** | 3.07 |
| 4 | **228** | 65.93 | **230** | 0.15 | **232** | -7.88E-05 | **228** | 2.96 |
| 5 | **230** | 41.50 | 181 | 0.13 | **228** | -7.98E-05 | 61 | 2.93 |
| 6 | **231** | 29.05 | 171 | 0.12 | **231** | -1.10E-04 | **233** | 2.82 |
| 7 | 61 | 24.65 | 55 | 0.11 | **226** | -1.30E-04 | **232** | 2.61 |
| 8 | **232** | 21.02 | **234** | 0.09 | 17 | -2.50E-04 | 17 | 2.55 |
| 9 | 25 | 18.73 | 39 | 0.08 | 61 | -3.00E-04 | **226** | 2.41 |
| 10 | 17 | 17.88 | 14 | 0.08 | 25 | -5.60E-04 | 95 | 2.37 |
| Correct | **8** | - | 3 | | **8** | - | 7 | |
| **Musk data set** | | | | | | | | |
| **Top-10** | **MOF** | | **AOF** | | **FastABOD** | | **LOF** | |
| | **Index** | **Score** | **Index** | **Score** | **Index** | **Score** | **Index** | **Score** |
| 1 | **2,972** | 57.95 | 1,647 | 11.44 | **2,965** | -8.59E-15 | **2,972** | 1.77 |
| 2 | **2,973** | 57.86 | **2,965** | 9.56 | 393 | -3.65E-14 | **2,973** | 1.77 |
| 3 | **2,969** | 53.93 | 1,835 | 7.97 | 126 | -3.88E-14 | **2,969** | 1.76 |
| 4 | **2,968** | 53.83 | 1,889 | 6.73 | 354 | -4.30E-14 | **2,968** | 1.76 |
| 5 | **2,966** | 53.63 | 1,668 | 5.08 | 83 | -4.45E-14 | **2,966** | 1.76 |
| 6 | **2,974** | 49.50 | 1,649 | 4.65 | 359 | -4.53E-14 | **2,965** | 1.76 |
| 7 | **2,971** | 49.46 | 501 | 4.64 | 224 | -4.56E-14 | **2,974** | 1.76 |
| 8 | **2,965** | 48.99 | **2,967** | 4.63 | 769 | -4.57E-14 | **2,971** | 1.76 |
| 9 | **2,970** | 44.98 | 1,898 | 4.46 | 320 | -4.67E-14 | **2,970** | 1.74 |
| 10 | **2,967** | 44.85 | 1,827 | 3.94 | 627 | -4.72E-14 | **2,967** | 1.74 |
| Correct | **10** | - | 2 | | 1 | - | **10** | |
| **Satimage-2 data set** | | | | | | | | |
| **Top-10** | **MOF** | | **AOF** | | **FastABOD** | | **LOF** | |
| | **Index** | **Score** | **Index** | **Score** | **Index** | **Score** | **Index** | **Score** |
| 1 | **5,738** | 512.69 | **5,738** | 21.36 | **5,738** | -2.9E-11 | **5,738** | 2.75 |
| 2 | **5,732** | 411.45 | **5,737** | 14.07 | **5,736** | -3.8E-11 | **5,732** | 2.46 |
| 3 | **5,739** | 388.40 | **5,739** | 6.56 | **5,735** | -3.8E-11 | **5,737** | 2.45 |
| 4 | **5,737** | 345.91 | **5,741** | 4.69 | **5,734** | -4.3E-11 | **5,739** | 2.43 |
| 5 | **5,734** | 185.57 | **5,740** | 3.85 | **5,732** | -4.5E-11 | 4,335 | 2.27 |
| 6 | **5,741** | 181.76 | **5,732** | 3.54 | **5,741** | -4.6E-11 | **5,741** | 2.25 |
| 7 | **5,735** | 114.32 | 4,241 | 3.31 | **5,739** | -4.7E-11 | **3,302** | 2.22 |
| 8 | **5,736** | 114.32 | 4,242 | 2.80 | **5,737** | -6.2E-11 | **5,740** | 2.17 |
| 9 | 936 | 93.79 | 404 | 1.60 | 3,366 | -8.1E-11 | **5,734** | 2.14 |
| 10 | 3,366 | 89.73 | **5,734** | 1.53 | 72 | -1.1E-10 | 964 | 2.12 |
| Correct | **8** | - | 7 | | **8** | - | **8** | |

The results of these experiments show that the MOF algorithm still can detect a few outliers in data sets better than other algorithms. For the LOF algorithm, if it can use a grid search for the best tune, the number of outliers detected by the LOF algorithm is near to the MOF algorithm.

### 5.5.5 Conclusions on Experimental Results for the MOF algorithm

The experimental evaluations presented in the previous sections for the MOF algorithm are as follows:

- The AOF and the OOF algorithms are not applicable to detect local outliers which are near the normal region and micro-outlier clusters that consist of a small group of outliers.
- The AOF and the OOF algorithms are also inapplicable for a data set consisting of different densities of the normal region.
- The LOF and the kNN algorithms are not applicable to detect outliers in data that is the unknown distribution of data. The parameter values affect the detection of abnormal data, especially the LOF algorithm.
- The MOF algorithm is applicable and effective in all cases. Moreover, it does not assume the density of outlier and normal data points. Hence, the applicability of the MOF algorithm is much broader compared to the LOF and the kNN algorithms, which must be set by some parameters.
- The MOF algorithm is perfectly applicable for data sets that contain dense outliers, such as the Musk data set, where the existing outlier detection technique fails to work.
- Although the execution time of the MOF algorithm is more than that of the LOF and the kNN algorithms, the accuracy of MOF is still better than theirs.
- The accuracy (AUC and AP) of the MOF algorithm is not based on any user parameter. Thus, the user can use it without prior knowledge.



## 5.6 Experiments and results of the SMOF algorithm

This section evaluates the SMOF algorithm with extensive experiments. They confirm that the SMOF algorithm can be used successfully to identify outliers that appear to be meaningful but cannot be identified by other algorithms. The first example starts with 2d synthesized data sets. The second example assigns outlier scores to 2d synthesized data sets which are evaluated performance by AUC, AP, and execute time metrics. In the third example, benchmark data sets are used and evaluated performance by AUC and AP. The latest experiment deals with the meaningful impact of window size on benchmark data sets.

### 5.6.1 Visualization of comparison sampling data sets

Since the purpose of WRS in the SMOF algorithm is to select half of the data points to preserve the normal data points and remove outliers throughout the processing of streaming data, WRS can be regarded as a random sampling algorithm weighted by the MOF score.

Figure 5.20 and Figure 5.21 show the sampling result of random sampling and WRS by MOF score. For the synthetized normal data set, WRS takes samples successfully in which high-density regions are preserved (inside the normal region) while low-density regions (normal border region) fade, but the random sampling distorts the shape of data distribution since it selects samples just uniformly regardless of the characteristics of data distribution. In the case of synthesized noise, which contains many outliers or noise, random sampling selects data points just uniformly in all regions. In contrast, WRS intensively selects data points in high-density regions and removes outliers in Figure 5.21(A), (B), (D), and (E). This result in Figure 5.21 (C) shows that WRS makes the boundaries between the high-density regions and the low-density regions clearer. Due to such an effect of WRS, outlier detection algorithms can easily distinguish outliers even if these data sets contain many noisy data points. However, Figure 5.21(F) shows that some edges of normal regions are removed.

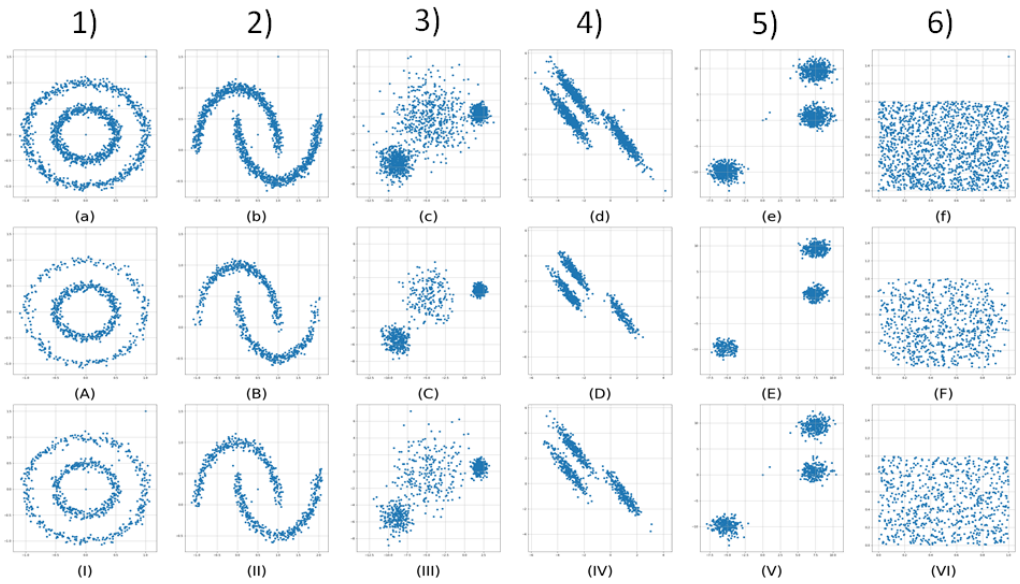



Figure 5.20 Original six synthesized normal data sets (a-f), half sampling results of them by weighted random sampling (A-F) and random sampling (I-VI)

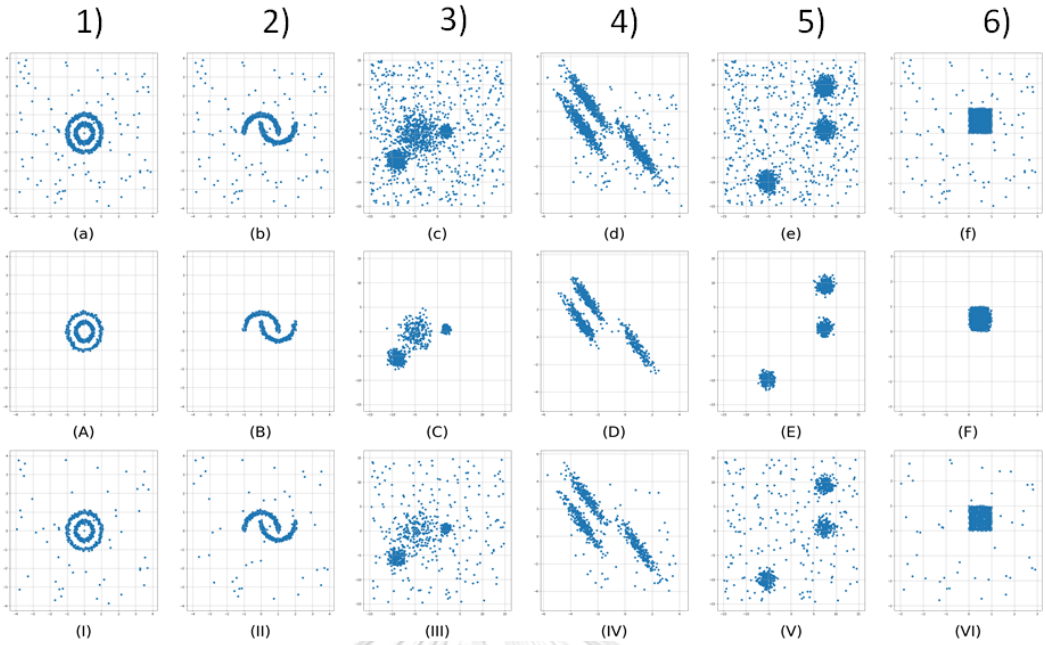



Figure 5.21 Original six synthetized-noisy data set data sets (a-f), half sampling results of them by Weighted random Sampling (A-F) and Random Sampling (I-VI)

In conclusion, the experiment shows that the SMOF algorithm can preserve normal data points and remove outliers every time sampling data points in reference and the current windows. This result leads to normal data points in the reference window, which make the MOF algorithm assign high outlier scores to data points in the current window which are not in the normal region.

## 5.6.2 Outlier scoring on 3d synthesized data



Since the SMOF algorithm detects outliers in each batch of streaming data, there will be a problem if the window size is too small to distinguish between normal data points and outliers. Furthermore, if concept drift occurs, normal data points from new unseen normal regions in the current window that differ from normal regions in the reference window may be misidentified as outliers. This section evaluates the SMOF algorithm compared to the HS-Trees algorithm on 3d synthesized data.

To understand how the HS-Trees and the SMOF algorithms assign outlier scores to this data set, Figure 5.22 depicts generated 3d synthesized data according to a timestamp in Figure 5.23. Before the normal region changes to the new normal region, sequence outliers occur between them. This situation is the so-called concept drift. The red, green, and purple data points are normal, while the red and orange data points are outliers. These scores are processed in the HS-Trees and the SMOF algorithms and return outlier scores to evaluate performance. The window sizes are set to range from 100 to 500. Other hyperparameters of the HS-Trees algorithm are default values according to Table 5.9.

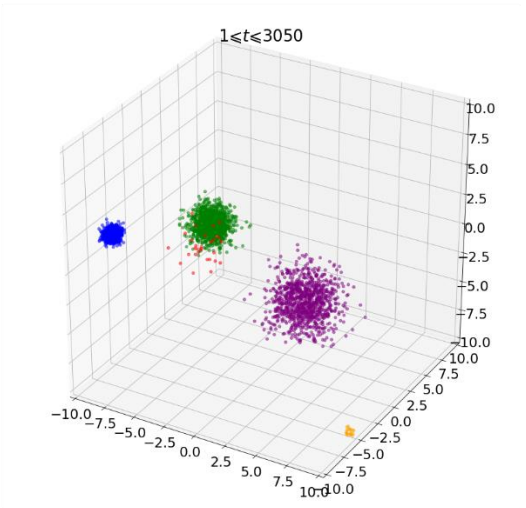



Figure 5.22 3d synthesized data

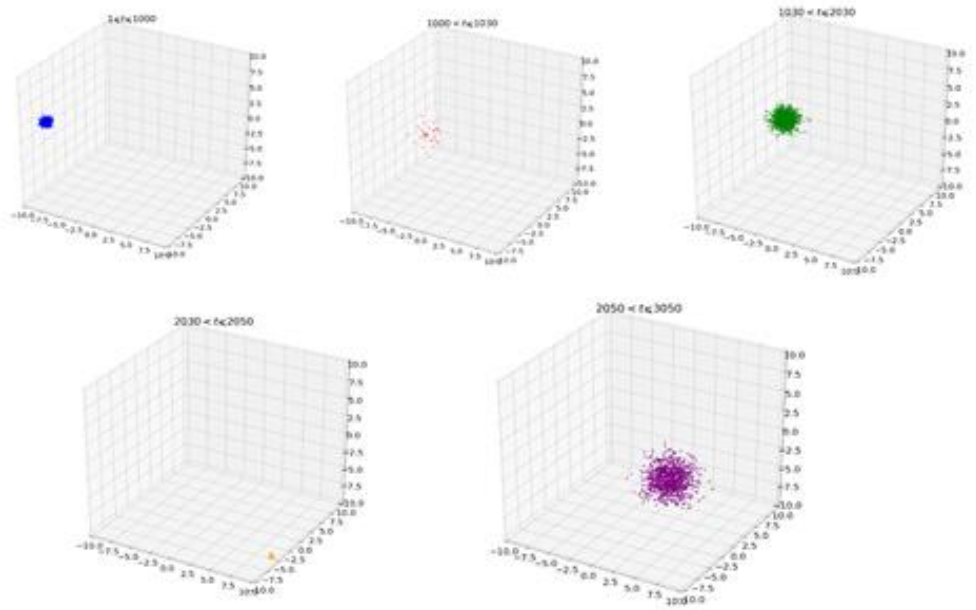


Figure 5.23 Timestamp (t) 0 to 4000 for a 3d synthesized data

Figure 5.24 depicts AUC and AP metrics of the SMOF and HS-Trees algorithms varying with a window size. It shows that the larger the window size, the accuracy of the SMOF algorithm tends to increase, while the accuracy of the HS-Trees algorithm tends to be stable. This is because when the concept drift occurs, normal regions in the reference window are different from normal regions in the current window. HS-Trees algorithm assigns outlier scores to current data points based only on the reference window leading to detecting them as an outlier. In contrast, the SMOF algorithm passes outlier scores to current data points based on both the reference and the current windows, leading to detecting them as normal data points. To better understand, Figure 5.25 shows the outlier score assigned by the

SMOF and the HS-Trees algorithms at the period time stamp ($w = 350$). It shows that when the normal region changes ($Batch\ A\ to\ B\ or\ C\ to\ D$), the current normal region ($Batch\ B\ and\ D$) is assigned a high outlier score by HS-Trees algorithm, while a low outlier score by the SMOF algorithm compares to the outlier score of the current normal region ($Batch\ A\ and\ C$). In addition, the execution time of the SMOF algorithm is faster than the HS-Trees algorithm. This is because the time complexity is based on the number of trees and deep trees. When the window size is large enough, the execution time tends to be stable. Although the time complexity of the SMOF algorithm is based on window size.

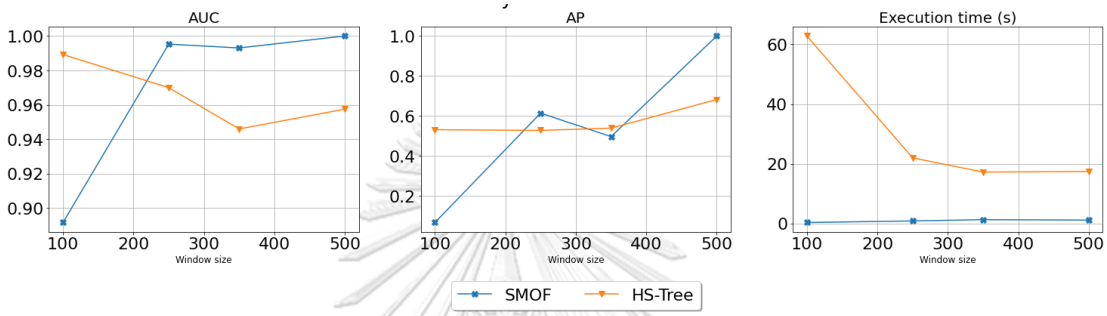



Figure 5.24 The impact of window size on 3d synthesized data set

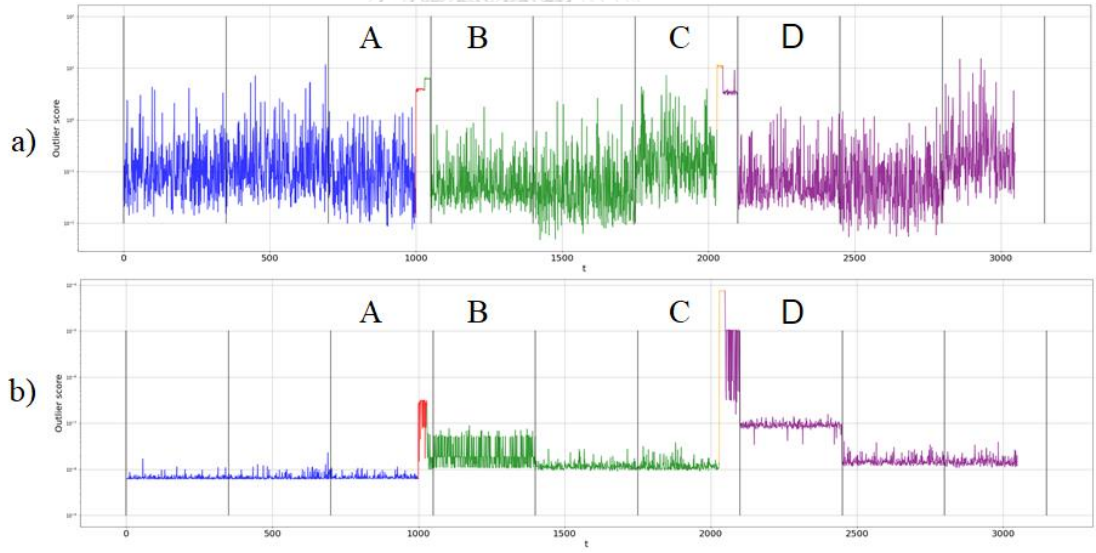



Figure 5.25 a) SMOF and b) HS-Trees scores of each data point in 3d synthesized data set

In conclusion, both the SMOF and the HS-Trees algorithms are passing high outlier scores to the new normal region. However, their outlier scores assigned by the HS-Trees algorithm are more sensitive than the SMOF algorithm.

### 5.6.3 Experiments on benchmark data sets

Since the previous experiment showed that the SMOF algorithm was able to detect synthesized multidimensional streaming data, it has the potential to detect outliers in streaming data. However, a small number of data points (only 3,050 data points) cannot be used to evaluate outlier detection in streaming data which is a large number of data points.

In this section, the SMOF algorithm is evaluated by synthesized and real-world data sets to compare performance among outlier detection in static (the LOF and the MOF algorithms) and streaming data (the HS-Trees algorithm). In fact, the SMOF algorithm does not require any parameter, for equal, each algorithm uses grid search to best tune performance with only one parameter. The range of values of the default values of each parameter is presented in Table 5.9. The default values are chosen based on the characteristics of existing literature. Although the HS-Trees algorithm has other parameters that are set to a default value, the SMOF algorithm has time complexity related to the window size while the HS-Trees algorithm does not.

Table 5.10 and Table 5.11 show the average AUC and AP of four algorithms for five data sets from 10 repeated trials. The underlying value indicated that they had the best accuracy. Moreover, the bottom of the tables has a summary of performance by average rank and performance accuracy. Since the number of data points in all data sets is enormous, the MOF algorithm does not deal with the large data sets, resulting in a memory leak in Google Colab; thus, the MOF algorithm cannot report any result.

The SMTP data set contains a few outliers (only 30 data points) that are far from the normal region. It is assumed to contain an outlier of type I. the LOF algorithm is acceptably designed for this situation where outliers have few and are isolated, although it is streaming data.



Other data sets have characteristics of streaming data. Moreover, some outliers are type I or II. Thus, LOF tries to detect outliers based on the assumption that they are locally dense according to the range of $k$-nearest neighbors. It causes numerous errors and misidentifies many normal data points as outliers, resulting in poor AUC and AP.

In contrast, AUC and AP of the SMOF and the HS-Trees algorithms are acceptable. Both of them can detect outliers with greater than 80% AUC for all data sets. These prove that the HS-Trees and the SMOF algorithms can detect outliers in streaming data.

Overall performance by average performance and rank can be summarized as follows:

- The MOF algorithm cannot process large data sets.
- The LOF algorithm has the worst accuracy, with an average ranking of 2.8 and 2.6 for AUC and AP.

The SMOF algorithm has the best performance accuracy with an average ranking of 1 and 1.4 for AUC and AP.

Table 5.9 Parameters setting

| Parameters | SMOF | HS-Trees | LOF | MOF |
|---|---|---|---|---|
| $k$ | N/A | N/A | 5- 300 | N/A |
| $w$ | 100 - 500 | 100 - 4000 | N/A | N/A |
| $h$ | N/A | 10 | N/A | N/A |
| $\mathcal{E}$ | N/A | 15 | N/A | N/A |
| $sizeLimit$ | N/A | $0.1w$ | N/A | N/A |

Table 5.10 Average AUC scores for streaming data

| Data sets | SMOF | | HS-Trees | | LOF | | MOF | |
|---|---|---|---|---|---|---|---|---|
| | Rank | AUC | Rank | AUC | Rank | AUC | Rank | AUC |
| SMTP+HTTP | 1 | 0.998 | 2 | 0.986 | 3 | 0.378 | N/A | N/A |
| HTTP | 1 | 0.999 | 2 | 0.996 | 3 | 0.369 | N/A | N/A |
| Mulcross | 1 | 0.998 | 2 | 0.971 | 3 | 0.606 | N/A | N/A |
| SMTP | 1 | 0.941 | 3 | 0.873 | 2 | 0.933 | N/A | N/A |
| Shuttle | 1 | 0.989 | 2 | 0.970 | 3 | 0.583 | N/A | N/A |
| **Average** | 1 | 0.985 | 2.2 | 0.959 | 2.8 | 0.574 | N/A | N/A |

Table 5.11 Average AP scores for streaming data

| Data sets | SMOF | | HS-Trees | | LOF | | MOF | |
|---|---|---|---|---|---|---|---|---|
| | Rank | AP | Rank | AP | Rank | AP | Rank | AP |
| SMTP+HTTP | 1 | 0.950 | 2 | 0.262 | 3 | 0.033 | N/A | N/A |
| HTTP | 1 | 0.954 | 2 | 0.475 | 3 | 0.046 | N/A | N/A |
| Mulcross | 1 | 0.966 | 2 | 0.782 | 3 | 0.177 | N/A | N/A |
| SMTP | 3 | 0.202 | 3 | 0.258 | 1 | 0.309 | N/A | N/A |
| Shuttle | 1 | 0.837 | 2 | 0.829 | 3 | 0.167 | N/A | N/A |
| **Average** | 1.4 | 0.782 | 2.2 | 0.521 | 2.6 | 0.146 | N/A | N/A |



## 5.6.4 Impact of the window size

This section aims to study the impact of the window size on the performance of the SMOF and the HS-Trees algorithms. Not only setting parameters are the same as in the previous section but also the data sets to evaluate are the same. This section also adds execution time to evaluate the time complexity of an algorithm. The results

of performance metrics in this section can be divided into accuracy (AUC and AP) and execution time, and it will analyze the experimental results of each data set as follows:

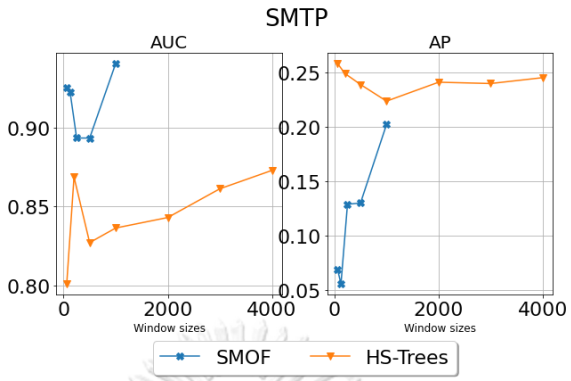



Figure 5.26 The impact of window size on SMTP data

Figure 5.26 shows the impact of window size on the performance of the SMOF and the HS-Trees algorithms. SMTP data is streaming data involving intrusions. SMTP data does not have surges of outliers, but possibly exhibits some distribution changes within the short streaming sequence.

To detect these few outliers, the window size of the SMOF algorithm must be large enough to identify outliers and normal data points. Therefore, the increased window size leads to increasing AUC and AP. However, an increased window size of the HS-Trees algorithm slightly affects accuracy.

One interesting in SMTP data at window size 400, AUC in the SMOF algorithm decreases while in the HS-Trees algorithm increases. This is because few sequence outliers occur between the joints of the reference and the current windows. This problem occurs the same as the 3d synthesized data set in Figure 5.25 in which the outlier scores normal data points (green points) after detecting the first sequence outlier (red points). The SMOF algorithm can decrease this problem by increasing the window size. However, both algorithms have a low AP because this data set is very unbalanced (0.03%), and some data points have all values in dimensions in the normal region.

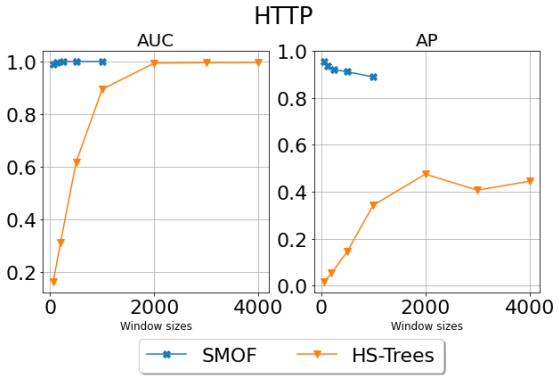



Figure 5.27 The impact of window size on HTTP data

Figure 5.27 shows that AUC and AP of the SMOF algorithm are stable while they increase in the HS-Trees algorithm with the increase of the window size in HTTP data. This is because HTTP data is streaming data involving network intrusions that are characterized by sudden outliers in some streaming segments. HTTP data also has a long sequence of dense duplicate outliers and distribution changes within the streaming sequence. The HS-Trees algorithm uses the mass profile in the reference window to measure the outlier score in the next window, which is also used to measure outliers in the window. If the next window contains all outliers, the HS-Trees algorithm will use the mass profile of the outlier to measure outlier scores leading to failure. Therefore, the increasing window will be the window that contains more normal data points than an outlier. However, the SMOF algorithm detects a long sequence of dense duplicate outliers. This is because the MOF algorithm has the function of duplicating outlier scoring. The more data redundancy, the higher the outlier score. Moreover, the MOF score is doted by the average MOF scores result which assigns a very high score to sequence outliers. To better understand, Figure 5.28 shows the outlier score of sequence outlier (orange data points) by a) the SMOF and b) the HS-Trees algorithm. They show that SMOF can assign outlier scores to most of the outliers more than normal data points (blue). Although the HS-Trees algorithm can assign outlier scores more than normal data points in the first period, the second period is less than normal data points.

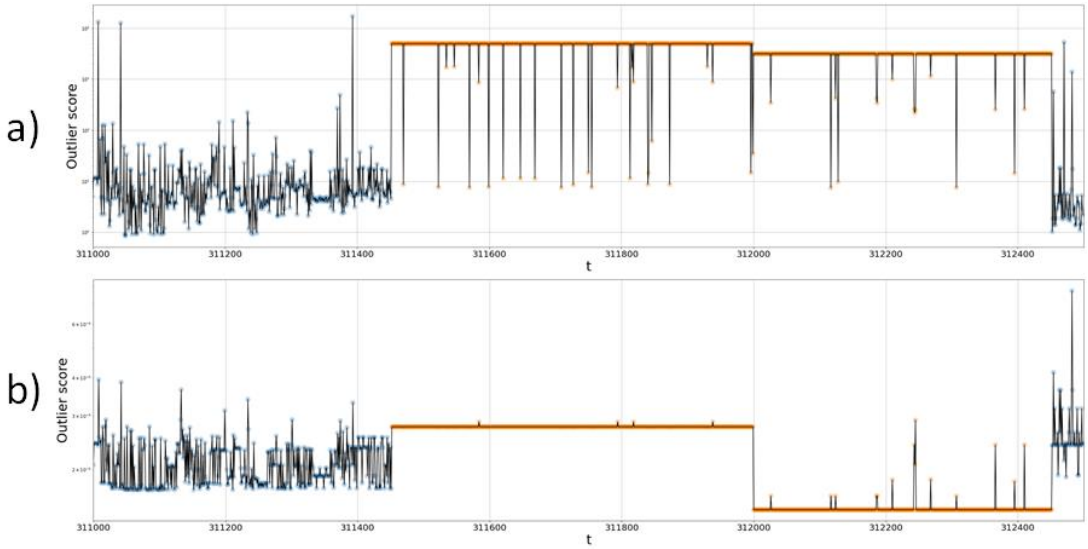



Figure 5.28 The outlier scores of sequence outliers (the orange data points) by a) the SMOF and b) the HS-Trees algorithms (window size is 1000).

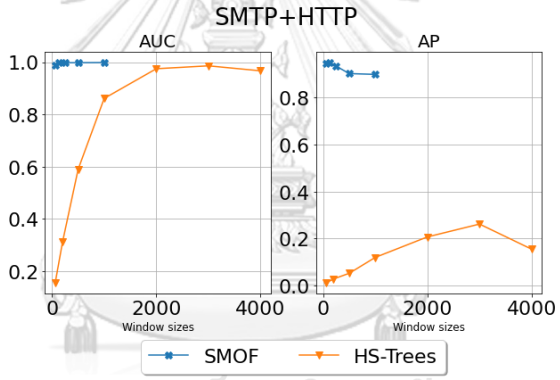





Figure 5.29 The impact of window size on SMTP+HTTP data

Figure 5.29 shows the impact of the window size on the performance of the SMOF and the HS-Trees algorithms. SMTP + HTTP data is a combination of SMTP data followed by HTTP data. When the communication protocol is switched from SMTP domain to HTTP domain, this data reflects the scenario distribution change that will occur. In fact, the results of AUC and AP in this data set are near to the ones in HTTP. This is because HTTP contains more data than SMTP, around 6 times, and both algorithms can deal with concept drift. Therefore, switching from SMTP to HTTP data does slightly affect AP and AUC. However, it still affects the HS-Trees algorithm. This is because HS-Trees structures binary trees are created based on SMTP data. Therefore, AUC and AP decrease.

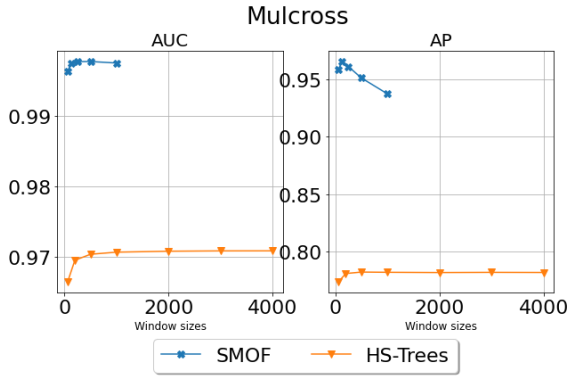



Figure 5.30 The impact of the window size on Mulcross data

Figure 5.30 shows the impact of the window size on the performance of the SMOF and the HS-Trees algorithms. Mulcross data distributions with little or no variation. Mulcross data, on the other hand, contains dense clusters of outliers that are more difficult to detect than scattered outliers. This data set contains small normal regions. AUC in both algorithms quickly converges to a stable state which is the narrow window size. In fact, the outlier is dense clusters, and the result of WRS sampling always contains outliers. Hence, the wider the window size, AUC of the SMOF algorithm decreases. However, the HS-Trees algorithm is not affected by this situation.

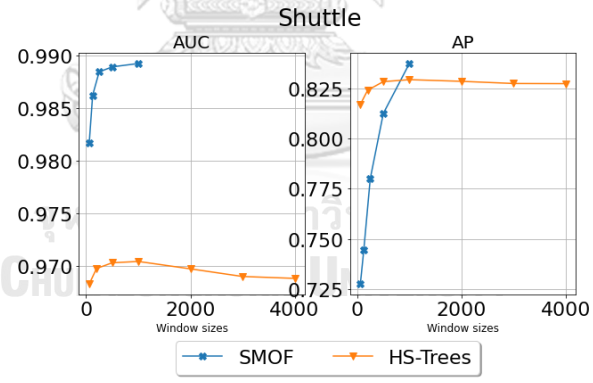



Figure 5.31 The impact of the window size on Shuttle data

Figure 5.31 shows the impact of window size on the performance of the SMOF and the HS-Trees algorithms. Shuttle data is a data set with little or no distribution changes and a high number of class outliers. In terms of AUC and AP, SMOF outperforms the HS-trees algorithm. There are a number of outliers in this data set. Increasing the window size has no effect on AUC score when window sizes are large enough to include both normal and outlier data points. In contrast, increasing the window size causes the outlier score of normal data points to increase and outlier clusters of data points to increase slightly, causing AP score tends to fall.

In summary, AUC increases with the increase of the window sizes in the SMOF and the HS-Trees algorithms from all data sets. This is because data sets contain a large normal region to preserve in the reference window. When the window size is large enough, both algorithms can preserve all normal regions to predict the current data points correctly, leading to AUC being stable. In addition, the SMOF algorithm can deal with duplicate outliers. The overall AUC and AP of the SMOF algorithm are better than the HS-trees algorithm.

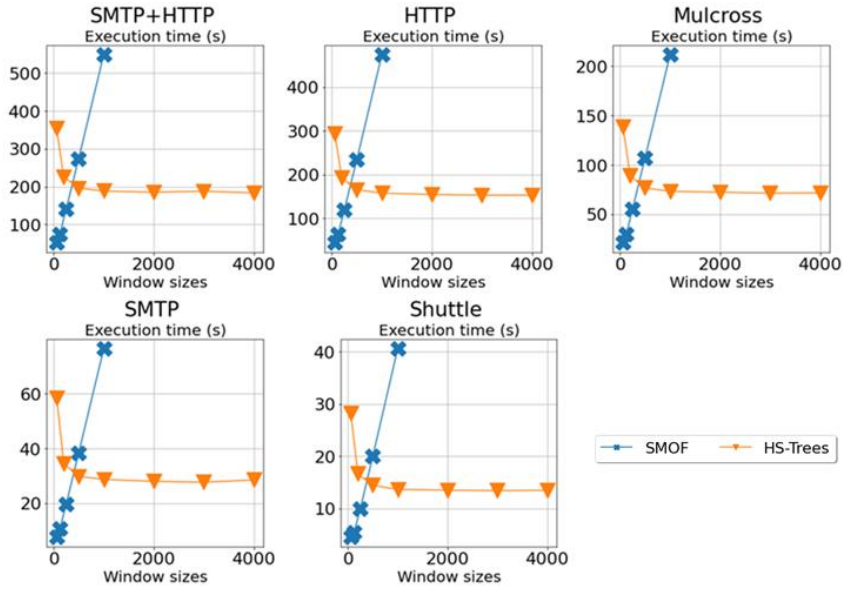



Figure 5.32 The impact of the window size on benchmark data

The experimental results in Figure 5.32 show that executive time increases in the SMOF algorithm while it converges to a stable state in the HS-Trees algorithm with the increase of the window size in all data sets. This is because the time complexity of the HS-Trees algorithm is $O(t(h))$ which does not affect the size of the window, while the time complexity of HS-Trees is $O(wlogw)$.

### 5.6.5 Conclusions on experimental results for the SMOF algorithm

The conclusions from the experimental results can be summarized as follows:

- Weighted random sampling based on MOF scores can pick normal data points and remove outliers better than simple random sampling. In addition, normal regions are still preserved.
- The SMOF algorithm performs better than the HS-Trees algorithm for real applications like network intrusion detection or NASA's space shuttle classification. The diversity of the data sets shows the applicability of SMOF for a wide range of applications.

- Although the SMOF algorithm processes with better accuracy compared to the HS-Trees algorithm, the SMOF algorithm has a worse competitive execution time.
- The impact of window size on accuracy is very small when it is large enough. Thus, the user has much liberty in choosing an appropriate value for this parameter. Even if the user chooses a value that is not optimal, the SMOF algorithm is still capable of detecting outliers.
- The execution time of the SMOF algorithm is also insensitive to the window size, which trades off accuracy. Thus, the user should choose the window size according to arrival time.
- The execution time of the SMOF algorithm increases with the log-linear increase in the window size.
- The SMOF algorithm has a function to deal with long sequences and duplicate outliers, while the HS-Trees algorithm does not.

# Chapter 6
# Conclusions and Future works

This thesis proposes two outlier detection techniques for static and streaming data called MOF and SMOF, respectively. The first algorithm, the MOF algorithm, unsupervised parameter-free outlier scoring, has been designed to detect outliers in multidimensional numeric data. The MOF algorithm processes all data points in a data set and returns the MOF score, which is a variance of the mass-ratio distribution. The large variance is associated with outliers, while the small variance is associated with normal data points.

The second algorithm, the SMOF algorithm, unsupervised outlier scoring, has been designed to measure outlier score outliers in numeric multidimensional streaming data. The SMOF algorithm is based on MOF with a detected outlier based on a non-overlapping sliding window. SMOF assigns SMOF scores and summarizes reference data points. SMOF addresses the following characteristics of streaming data: transiency, the notion of time, the notion of infinity, and concept drift.

The complexity analysis has been conducted to evaluate the time and space complexity of the MOF and the SMOF algorithms. By means of simulation and using data sets, comprehensive experiments are performed to compare the MOF algorithm with the four existing algorithms: LOF, AOF, OOF, and FastABOD, for static data and compare SMOF with the HS-Trees algorithm for streaming data. In the next section, the performance evaluation results and the future research are discussed.

## 6.1 Conclusions of the MOF algorithm



Outlier detection in static data is most important for data presenting to detect or remove outliers. The simple and effective outlier detection is a proximity model, which makes it easy to understand how an outlier is different from a normal data point. Their main idea is that the outliers are far from the rest of the normal data points in a data set. In addition, most of them can detect outliers by measuring outliers that allow a user to choose the top-$n$ outliers. However, most of them also require parameters for identifying outliers, such as $k$-nearest neighbors, number of clusters, or radius distance. The wrong parameter results in failure to detect. Hence, detecting or measuring outliers with parameter-free outlier detection is very challenging.

This thesis proposed novel outlier detection with a parameter-free technique for static data. The MOF algorithm uses a variance of mass ratio to measure the outlier score of data points. The conclusion of the MOF algorithm is as follows:

- The MOF algorithm is the only parameter-free outlier scoring technique for static data based on the density concept that does not assume any distribution of the data set.

- The time complexity of the MOF algorithm is $O(n^2 logn)$ with respect to the number of data points in the data set. Most of the time complexity is due to the sorting step that sort $n$ data points.
- The space complexity of the MOF algorithm is $O(n^2)$ with respect to the number of data points in the data set. Most of the space complexity involves collecting all pairwise distances of data points in a data set.
- The LOF and the kNN algorithms are needed $k$-nearest neighbors' parameter that is applicable to detect outliers with knowledge of data sets due to the sensitivity of the parameter.
- The AOF and the OOF algorithms are based on the first nearest neighbors, so when the data set consists of the micro-outlier detection, local outlier, or different density normal regions, it cannot detect them.
- The MOF algorithm with some threshold can detect outliers of synthesized data sets by generating many types of distributions such as gaussian distribution, half-moon, or circle according to the density concept.
- The MOF algorithm has performance no different compared to the LOF and the OOF algorithms, while different compared with the AOF algorithm.
- The MOF algorithm has AUC and AP scores that are better than LOF, kNN, and OOF on most of the benchmark data sets. However, execution times are always more than LOF and kNN and less than OOF.
- The MOF algorithm can detect more outliers than AOF, FastABOD, and LOF with the best tune parameter on real-world data sets.

## 6.2 Conclusions of the SMOF algorithm

The SMOF algorithm is designed for multidimensional streaming data non-overlapping on sliding windows. Unlike other approaches, the SMOF algorithm does not use user-parameter settings to assume any distribution of data. the SMOF algorithm tries to keep the present normal data points in the reference window based on the MOF scores and measure SMOF scores on the data points of each batch with respect to the reference and the current data points. The summary of the SMOF algorithm is as follows:

- The SMOF algorithm is flexible and easy to use for detecting outliers in multidimensional streaming data.
- Although the MOF algorithm is parameter-free, the SMOF algorithm still needs the window size, even though it is not a critical parameter as long as the window contains more normal data points than outliers.
- Neither time nor space is sensitive to the number of data points. The memory usage of the SMOF algorithm is limited, although the number of data points is infinite.
- The SMOF algorithm attempts to keep the current normal data points and discard old reference data points in the reference window by using weighted random sampling.

- The SMOF algorithm can adapt window size to adapt execution time, according to the arrival rate of seaming data.
- Experiments studying the impacts of the concept drift show that the SMOF algorithm can handle concept drift in SMTP+HTTP data effectively without affecting its accuracy while the HS-Trees algorithm is not applicable to detect outliers that occur in concept drift because the binary trees construct only at the start of streaming data.
- Experiments studying the impacts of a long sequence of outliers show that the SMOF algorithm can handle them in HTTP data effectively without affecting its accuracy, while the HS-Trees algorithm fails.
- AUC of the SMOF algorithm increases with the increase in the window sizes, even if the data set has a large number of outliers in batches. However, AP of SMOF slightly decreases with the increasing of window sizes if the outlier is densely likely to have normal data points but a few. Normal data points and outliers also increase the outlier score.
- AUC of the SMOF algorithm is stable when the window size is large enough to preserve the normal region of data points. The recommended window size setting is 500 according to the experimental results.
- The execution time of the SMOF algorithm increases with the increase of window size. After calculating MOF scores with respect to the window size, the user should consider the execution time.
- The average time complexity of the SMOF algorithm to detect a data point is $O(wlogw)$ to detect a data point with respect to window size $w$.
- The space complexity of SMOF is $O(w^2)$ with respect to the window size.

## 6.3 Future works

- The MOF and the SMOF algorithms are very flexible outlier detection techniques for multidimensional streaming and static data. Although it is very scalable in terms of accuracy, its execution time grows as the number of window sizes and data points increases. The fast and scalable version of the MOF and the SMOF algorithms in terms of execution time should be developed in the future.
- The MOF and the SMOF algorithms have just measured the outlier score of the data points, and that each attribute is of the same importance, there are no explored cross-correlations among the data points. Therefore, it will be more interesting to study the effect of complex cross-correlations among the data points from multidimensional streaming data to the SMOF algorithms.

# REFERENCES


1. Zimek, A. and E. Schubert, *Outlier Detection*, in *Encyclopedia of Database Systems*, L. Liu and M.T. Özsu, Editors. 2017, Springer New York: New York, NY. p. 1-5.

2. Aggarwal, C.C., *An Introduction to Outlier Analysis*, in *Outlier Analysis*. 2017, Springer International Publishing: Cham. p. 1-34.

3. Ahmed, M., A.N. Mahmood, and J. Hu, *A survey of network anomaly detection techniques.* Journal of Network and Computer Applications, 2016. 60: p. 19-31.

4. Aleskerov, E., B. Freisleben, and B. Rao. *Cardwatch: A neural network based database mining system for credit card fraud detection*. in *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr)*. 1997. IEEE.

5. Nasi, J., A. Sorsa, and K. Leiviska. *Sensor validation and outlier detection using fuzzy limits*. in *Proceedings of the 44th IEEE Conference on Decision and Control*. 2005. IEEE.

6. Spence, C., L. Parra, and P. Sajda. *Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model*. in *Proceedings IEEE workshop on mathematical methods in biomedical image analysis (MMBIA 2001)*. 2001. IEEE.

7. Chandola, V., A. Banerjee, and V. Kumar, *Anomaly detection: A survey.* ACM computing surveys (CSUR), 2009. 41(3): p. 1-58.

8. Hodge, V. and J. Austin, *A survey of outlier detection methodologies.* Artificial intelligence review, 2004. 22(2): p. 85-126.

9. Goldberger, A.L., et al., *PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.* circulation, 2000. 101(23): p. e215-e220.

10. Ester, M., et al. *A density-based algorithm for discovering clusters in large spatial databases with noise*. in *kdd*. 1996.

11. Ertöz, L., M. Steinbach, and V. Kumar, *Finding topics in collections of documents: A shared nearest neighbor approach*, in *Clustering and information retrieval*. 2004, Springer. p. 83-103.

12. Kohonen, T. *Exploration of very large databases by self-organizing maps*. in *Proceedings of international conference on neural networks (icnn'97)*. 1997. IEEE.

13. MacQueen, J. *Some methods for classification and analysis of multivariate observations*. in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. 1967. Oakland, CA, USA.

14. Dempster, A.P., N.M. Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society: Series B (Methodological), 1977. 39(1): p. 1-22.

15. Smith, R., et al., *Clustering approaches for anomaly based intrusion detection.* Proceedings of intelligent engineering systems through artificial neural networks, 2002. 9.

16. He, Z., X. Xu, and S. Deng, *Discovering cluster-based local outliers.* Pattern recognition letters, 2003. 24(9-10): p. 1641-1650.

17. Zhang, J. and H. Wang, *Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance.* Knowledge and information systems, 2006. 10(3): p. 333-355.

18. Angiulli, F. and C. Pizzuti. *Fast outlier detection in high dimensional spaces*. in *European conference on principles of data mining and knowledge discovery*. 2002. Springer.

19. Eskin, E., et al., *A geometric framework for unsupervised anomaly detection*, in *Applications of data mining in computer security*. 2002, Springer. p. 77-101.

20. Knorr, E.M. and R.T. Ng. *Finding intensional knowledge of distance-based outliers*. in *Vldb*. 1999. Citeseer.

21. Din, S.U., et al., *Data stream classification with novel class detection: a review, comparison and challenges.* Knowledge and Information Systems, 2021. 63(9): p. 2231-2276.

22. Buthong, N., A. Luangsodsai, and K. Sinapiromsaran. *Outlier detection score based on ordered distance difference*. in *2013 International Computer Science and Engineering Conference (ICSEC)*. 2013. IEEE.

23. Kiangia, W., A. Luangsodsai, and K. Sinapiromsaran. *Weighted minimum consecutive pair of the extreme pole outlier factor*. in *2016 International Computer Science and Engineering Conference (ICSEC)*. 2016. IEEE.

24. Pumruckthum, P., S. Boonsiri, and K. Sinapiromsaran. *Parameter-Free Outlier Scoring Algorithm Using the Acute Angle Order Difference Distance*. in *International Conference on Computing and Information Technology*. 2019. Springer.

25. Kriegel, H.-P., M. Schubert, and A. Zimek. *Angle-based outlier detection in high-dimensional data*. in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008.

26. Barnett, V. and T. Lewis, *Outliers in statistical data.* Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, 1984.

27. Thakkar, P., J. Vala, and V. Prajapati, *Survey on outlier detection in data stream.* Int. J. Comput. Appl, 2016. 136(2): p. 13-16.

28. Zubaroglu, A. and V. Atalay, *Data stream clustering: a review.* Artificial Intelligence Review, 2021. 54(2): p. 1201-1236.

29. Angiulli, F. and F. Fassetti. *Detecting distance-based outliers in streams of data*. in

*Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 2007.

30. Yang, D., E.A. Rundensteiner, and M.O. Ward. *Neighbor-based pattern detection for windows over streaming data*. in *Proceedings of the 12th international conference on extending database technology: advances in database technology*. 2009.

31. Cao, L., et al. *Scalable distance-based outlier detection over high-volume data streams*. in *2014 IEEE 30th international conference on data engineering*. 2014. IEEE.

32. Kontaki, M., et al. *Continuous monitoring of distance-based outliers over data streams*. in *2011 IEEE 27th International Conference on Data Engineering*. 2011. IEEE.

33. Papadimitriou, S., et al. *Loci: Fast outlier detection using the local correlation integral*. in *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*. 2003. IEEE.

34. Zhang, K., M. Hutter, and H. Jin. *A new local distance-based outlier detection approach for scattered real-world data*. in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 2009. Springer.

35. Kriegel, H.-P., et al. *LoOP: local outlier probabilities*. in *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009.

36. Pokrajac, D., A. Lazarevic, and L.J. Latecki. *Incremental local outlier detection for data streams*. in *2007 IEEE symposium on computational intelligence and data mining*. 2007. IEEE.

37. Salehi, M., et al., *Fast memory efficient local outlier detection in data streams.* IEEE Transactions on Knowledge and Data Engineering, 2016. 28(12): p. 3246-3260.

38. Na, G.S., D. Kim, and H. Yu. *Dilof: Effective and memory efficient local outlier detection in data streams*. in *Proceedings of the 24th ACM SIGKDD International Conference on*

*Knowledge Discovery & Data Mining*. 2018.

39. Cordeiro, M., et al., *Evolving networks and social network analysis methods and techniques.* Social media and journalism-trends, connections, implications, 2018: p. 101-134.

40. Efraimidis, P. and P. Spirakis, *Weighted Random Sampling*, in *Encyclopedia of Algorithms*, M.-Y. Kao, Editor. 2008, Springer US: Boston, MA. p. 1024-1027.

41. Breunig, M.M., et al. *LOF: identifying density-based local outliers*. in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000.

42. Bolton, R.J. and D.J. Hand, *Unsupervised profiling methods for fraud detection.* Credit scoring and credit control VII, 2001: p. 235-255.

43. Asuncion, A. and D. Newman, *UCI machine learning repository*. 2007, Irvine, CA, USA.

44. Yamanishi, K., et al., *On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms.* Data Mining and Knowledge Discovery, 2004. 8(3): p. 275-300.

45. Rocke, D.M. and D.L. Woodruff, *Identification of outliers in multivariate data.* Journal of the American Statistical Association, 1996. 91(435): p. 1047-1061.

46. Zhao, Y., Z. Nasrullah, and Z. Li, *Pyod: A python toolbox for scalable outlier detection.* arXiv preprint arXiv:1901.01588, 2019.

47. Lam, S.K., A. Pitrou, and S. Seibert. *Numba: A llvm-based python jit compiler*. in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015.

# VITA

| | |
|---|---|
| **NAME** | Phichapop Changsakul |
| **DATE OF BIRTH** | 9 August 1996 |
| **PLACE OF BIRTH** | Bangkok, Thailand |
| **INSTITUTIONS ATTENDED** | Bachelor of Electrical Engineering Programs, Kasetsart University |