Outlier Detection in Non-stationary Data Streams

Luan Tran
luantran@usc.edu
University of Southern California
Los Angeles, California

Liyue Fan
liyuefan@albany.edu
University at Albany, SUNY
Albany, New York

Cyrus Shahabi
shahabi@usc.edu
University of Southern California
Los Angeles, California

ABSTRACT

Continuous outlier detection in data streams is an important topic in data mining and has applications in various domains such as fraud detection, weather analysis, and intrusion detection. The non-stationary characteristic of real-world data streams brings the challenge of updating the outlier detection model in a timely and accurate manner. In this paper, we propose a framework for outlier detection in non-stationary data streams (O-NSD) which detects changes in the underlying data distribution to trigger a model update. We propose an improved distance function between sliding windows which offers a monotonicity property; we develop two accurate change detection algorithms, one of which is parameter-free; and we further propose new evaluation measures that quantify the timeliness of the detected changes. Our extensive experiments with real-world and synthetic datasets show that our change detection algorithms outperform the state-of-the-art solution. In addition, we demonstrate our O-NSD framework with two popular unsupervised outlier classifiers. Empirical results show that our framework offers higher accuracy and requires a much lower running time, compared to retrain-based and incremental update approaches.

CCS CONCEPTS

• **Information systems** → **Data streams**; **Data mining**.

KEYWORDS

outlier detection, non-stationary data streams

ACM Reference Format:
Luan Tran, Liyue Fan, and Cyrus Shahabi. 2019. Outlier Detection in Non-stationary Data Streams. In *31st International Conference on Scientific and Statistical Database Management (SSDBM '19), July 23–25, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3335783.3335788

1 INTRODUCTION

More than ever, streaming data is increasing in volume and prevalence. It comes from many sources such as sensor networks, GPS devices, IoT devices, and wearable devices. Outlier detection in data streams is an important data mining task as a pre-processing step, and also has many applications in fraud detection, medical and public health anomaly detection, to name a few. A data object is considered an outlier if it does not conform to the expected behavior. In general, normal data objects, i.e., inliers, conform to a specific model as if that model has generated them, and outliers do not fit that model. In practice, most outlier detection techniques are *unsupervised* because of the difficulty in obtaining labelled data. In the literature, researchers have introduced many modeling techniques, e.g., proximity-based models [44], linear models such as Principal Component Analysis (PCA) [1, 11, 22, 26], One-class Support Vector Machine [47], and non-linear models such as One-class Neural Network [10].

Typically, with data streams, outlier detection is performed over sliding windows. The model trained with the data in the first window can be applied to the data in subsequent windows generated by the same underlying distribution. However, real-world data streams are usually non-stationary in which the underlying distribution changes over time, e.g., mean, variance, and correlation change [9, 34]. Such non-stationary data streams can be observed in many real-world datasets, for example, climate and transportation data streams. Figure 1 depicts the wind speed measured in meter/second at the same location on two different days, collected by the Tropical Atmosphere-Ocean project[1]. As depicted in the figure, the distribution of wind speed changes over hours and days. We observe changes in mean (from 7.62 to 3.95) and variance (from 3.96 to 1.42) between the two days. In particular, the wind speeds below 3 m/s (circled) are likely outliers on the first day and are normal on the second day. As a result, the model built from data on the first day is not suitable for the second day.

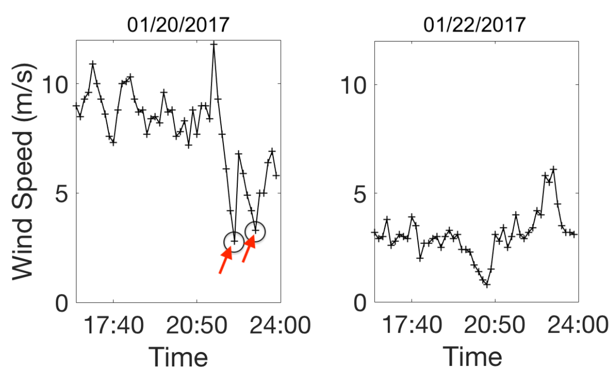


Figure 1: Wind speed data on two days

A baseline approach to tackle the data non-stationarity is to rebuild or incrementally update [2, 4, 8, 21, 35] the model in every sliding window, which is computationally expensive. Another

[1] https://www.pmel.noaa.gov/tao/drupal/disdel/

SSDBM '19, July 23–25, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6216-0/19/07...$15.00
https://doi.org/10.1145/3335783.3335788

method is to rebuild the model after a fixed time interval [47], which is hard to optimize because the timing of changes is usually unpredictable in practice. In this paper, we propose a *change detection* approach which rebuilds the model only when the changes of the underlying distribution are detected in data streams. Intuitively, as more data points from a new distribution arrive, the difference between current window and the previous distribution tends to increase. Therefore, we can monitor the sequence of the aforementioned “differences" continuously to determine whether the underlying distribution is going through changes. One advantage of our framework is that the change detection algorithms do not limit the choice of the outlier classifiers. This feature is crucial as real-world applications [28, 32, 33] typically use multiple outlier classifiers. Specifically, the contributions of our study are summarized as follows:

(1) We propose a framework for outlier detection in non-stationary data streams (O-NSD) which incorporates distribution change detection to trigger model updates. When a change is detected, the outlier detection model is rebuilt with the data from the new distribution.

(2) We propose a distance function IKL to measure the difference between two distributions. We prove that the distance between the current window and the reference window monotonically increases at the beginning of the new distribution.

(3) We propose two algorithms for change detection, i.e., AVG and Dynamic LIS, based on the average distance to the reference window and the length of the longest increasing subsequence of distance values, respectively. The threshold in Dynamic LIS is dynamically computed in a parameter-free manner. Experiments show that our algorithms are superior to the state-of-the-art approach for change detection.

(4) We propose new evaluation metrics for change detection, i.e., $wPrecision$, $wRecall$ and $wF1$, to quantify the timeliness of the detected changes in the context of data streams.

(5) With extensive experiments using synthetic and real-world datasets, we show that our outlier detection framework offers higher accuracy and incurs much less running time than the incremental and retrain-based outlier detection approaches.

The remainder of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we present the fundamental concepts and the problem definition. In Section 4 and Section 5, we propose our change detection, outlier detection solutions and evaluation metrics. In Section 6, we report detailed evaluation results. Finally, in Section 7, we conclude the paper and offer some future research directions.

2 RELATED WORK

Outlier Detection. Most outlier detection techniques are unsupervised because it is hard to get labelled data in practice. Principal Component Analysis (PCA) [1] and One-class SVM [1, 47] are the two most popular models for unsupervised outlier detection. PCA finds the orthogonal dimensions that captures the most variance of data. The data points that have large variances in the dimensions corresponding to the least eigenvalues in which most data have small variances are considered outliers. PCA has been applied in various domains, such as network intrusion detection [42] and in space craft components [17]. Incremental PCA has been used in visual novelty detection mechanism [30], outlier detection in energy data streams [13], and spatial-temporal data in [6]. One-class SVM finds the boundary for the most data with the assumption that outliers are rare. The data points which are outside of the boundary are considered outliers. It has many applications, e.g., in wireless sensor network [47] and time-series data [27]. To the best of our knowledge, the adaption to data streams, i.e., incrementally update One-class SVM, is not available.

Change Detection. Change detection in data streams has been well studied for one-dimensional data, e.g., in [7, 23, 45]. In this paper, we are interested in detecting a change in the unlabeled multidimensional data streams which were studied in [9, 25, 34]. Dasu et al. [9] detected changes in multidimensional data by computing the KL-distance from the estimated distribution of the current sliding window and the reference window, a change is reported when that distance is higher than a fixed threshold in a number of consecutive windows which is manually chosen. The training data is required to compute the threshold. Kuncheva in [25] proposed a semi parametric log likelihood detector to measure the difference between windows for detecting changes. Qahtan et al. [34] proposed a PCA-based change detection algorithm using Page-Hinckley test [29] which is shown to be superior to the methods in [9, 25]. Our solutions aim to address several shortcomings of previous studies, e.g., manually chosen parameters and sensitivity to temporary spikes in the streams.

3 PRELIMINARIES

Below we present the fundamental concepts and problem definition.

Definition 1. [44] *A data stream is a possible infinite series of data points* $\ldots, o_{n-2}, o_{n-1}, o_n, \ldots$, *where the data points are sorted by their arrival time.*

In this definition, a data point o is associated with a time stamp $o.t$ at which it arrives. As new data points arrive continuously, data streams are typically processed in *sliding windows*, i.e., sets of active data points. The window size characterizes the volume of the data streams. In this study, we adopt the *count-based window*.

Definition 2. [44] *Given data point* o_n *and a fixed window size* W, *the count-based window* D_n *is the set of* W *data points:* $\{o_{n-W+1}, o_{n-W+2}, \cdots, o_n\}$.

Every time the window slides, S new data points arrive in the window and the oldest S data points are removed. S denotes the slide size which characterizes the speed of the data stream.

In real-world data streams, changes in the underlying data distribution may be inherent due to the nature of data. For example, the distributions of wind speed and precipitation change over seasons [2]; the average speed on a highway changes over hours and days. In addition, changes may happen if a sensor becomes less accurate gradually over time or when another sensor with a different calibration replaces the faulty sensor [18].

[2] https://www.pmel.noaa.gov/tao/drupal/disdel/

Definition 3. *A data stream is non-stationary if the parameters of the underlying distribution change over time.*

In this study, we consider changes in mean and variance in individual dimensions and in correlation between dimensions as in [9, 34]. We now formally define the problem of continuous outlier detection in non-stationary data streams (O-NSD) as follows.

PROBLEM 1 (O-NSD). *Given a non-stationary stream* $\{o\}$*, window size* W*, slide size* S*, the problem is to detect the outliers in every sliding window* $..., D_n, D_{n+S}, ...$

In this paper, we are interested in unsupervised approaches [13, 33, 36] for outlier detection in which each data point is given an outlier score measuring the quality of the fit to the model of normal behavior. We will present more details about outlier scoring in Section 5.

4 CHANGE DETECTION

In this section, we present our proposed solution for change detection which is crucial in the outlier detection framework. The high-level pseudo-code is presented in Algorithm 1. As shown in [34], a change in mean, variance or correlation in the original space is manifested in the transformed space using the Principal Component Analysis (PCA)[43]. Therefore, we apply PCA transformation on the windows as in line 2 in Algorithm 1 and select the first k principal components corresponding to the largest eigenvalues λ_i that satisfy $\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq 0.999$, where d is the number of dimensions. The window used for model building is called the *reference window*, which will be updated once a new distribution is detected. The distribution of the projected data is estimated in line 5 using histograms in each dimension whose edges are estimated as the maximum of the Sturges [41] and FD estimators [16]. Subsequently, the distance between two windows is the maximum distance across all dimensions, as in line 6 in Algorithm 1.

Algorithm 1 Change Detection

Global variables: $buffer$: The distance buffer.

```
1:  function ChangeDetection(D_t, D_r)  ▷ D_t: current window, D_r: reference window
2:      D'_r, D'_t := Apply PCA Transform to D_r, D_t
3:      dis = 0
4:      for i from 1 to d do
5:          f^i, g^i := apply EstimatedHist to D_r^{i'} and D_t^{i'}
6:          dis = max(dis, IKL(g^i || f^i))
7:      buffer.enqueue(dis)
8:      if IsChange(buffer) then
9:          report a change at D_t
10:         buffer.clear()
```

4.1 IKL - An Improved Distance Measure

The distance measure as in line 6 in Algorithm 1 is a crucial part for detecting a change. Kullback-Leibler distance [24] is commonly used to measure distance between two probability distributions:

$$KL(P,Q) = \sum_j P(j) \log \frac{P(j)}{Q(j)} \tag{1}$$

with $P(j)$ and $Q(j)$ are the probabilities of data values in bin j. $KL(P,Q)$ is positive if P and Q have different counts over bins. In other words, if D and D' are 2 sliding windows and are generated by different underlying distributions, we have $KL(D,D') > 0$. However, since a histogram only approximates a distribution, the KL distance between two histograms from the same distribution can be positive. Thus, only using positive value of KL distance is inaccurate for change detection. A threshold can be used for KL distance to detect a change. However, it is not practical as setting the threshold may require knowledge about the magnitude of the change a priori. Therefore, in this section, we propose an improved distance measure between two estimated distributions as follows.

$$IKL(P,Q) = \sum_j \max(P(j) \log \frac{P(j)}{Q(j)}, Q(j) \log \frac{Q(j)}{P(j)}) \tag{2}$$

We replace each term $P(j) \log \frac{P(j)}{Q(j)}$ in the original KL divergence formula by $\max(P(j) \log \frac{P(j)}{Q(j)}, Q(j) \log \frac{Q(j)}{P(j)})$ to maximize the distance. Assuming each slide is generated by the same distribution, the new IKL formula has the following characteristic.

Theorem 1. *Suppose* D_p *is the last sliding window from the previous distribution,* D_l *and* $D_{l'}$ *are sliding windows overlapping the two distributions and containing* l *and* l' *slides from the new distribution, respectively. When* $0 < l < l' < W/S$*, we have:* $IKL(D_p, D_l) < IKL(D_p, D_{l'})$*.*

In other words, at the beginning of a new distribution, the IKL distance to the reference window monotonically increases. Note that the KL distance does not have this characteristic.

Proof of Theorem 1.

Assume the histogram of the previous distribution contains n bins $B^{(p)} = \{b_1^{(p)}, b_2^{(p)}, ..., b_n^{(p)}\}$ with the probability that a data point belongs to bin b_i is $x_i^{(p)}$, $x_1^{(p)} + x_2^{(p)} + ... + x_n^{(p)} = 1$. Assume the probability distribution of the new distribution over the bins $B^{(p)}$ is $\{\gamma_1, \gamma_2, ..., \gamma_n\}$, $\gamma_1 + \gamma_2 + ... + \gamma_n \leq 1$. Here, we assume that for one distribution the data points in every slide are distributed over the bins similarly to the data points in the entire window. When the window receives new l slides from a new distribution and it removes l expired slides, let the probability that one data point belongs to bin b_i be $x_i^{(l)}$. We will compute $x_i^{(l)}$ from $x_i^{(p)}$, γ_i, the window size W, and the slide size S. With l expired slides, the probability of a data point to belong to bin b_i decreases $lS/Wx_i^{(p)}$, and with new l slides from the new distribution, the probability of a data point to belong to bin i gains $l\gamma_i S/W$. Therefore,

$$x_i^{(l)} = x_i^{(p)} - l\frac{S}{W}x_i^{(p)} + l\frac{S}{W}\gamma_i \tag{3}$$

Let $\beta_i = \frac{S}{W}x_i^{(p)} - \frac{S}{W}\gamma_i$, we have $x_i^{(l)} = x_i^{(p)} - l\beta_i$. The IKL distance between D_p and D_l is:

$$
\begin{aligned}
IKL(D_p, D_l) &= \sum_{i=1}^{n} \max(x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(l)}}, x_i^{(l)} \log \frac{x_i^{(l)}}{x_i^{(p)}}) \\
&= \sum_{i=1}^{n} \max(x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l\beta_i}, x_i^{(l)} \log \frac{x_i^{(p)} - l\beta_i}{x_i^{(p)}})
\end{aligned} \tag{4}
$$

Similarly, we have

$$
IKL(D_p, D_{l'}) = \sum_{i=1}^{n} \max(x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l'\beta_i}, x_i^{(l')} \log \frac{x_i^{(p)} - l'\beta_i}{x_i^{(p)}}) \tag{5}
$$

It is easy to see that: when $\beta_i \geq 0$,

$$
\max(x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l\beta_i}, x_i^{(l)} \log \frac{x_i^{(p)} - l\beta_i}{x_i^{(p)}}) = x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l\beta_i} \tag{6}
$$

and when $\beta_i < 0$,

$$
\max(x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l\beta_i}, x_i^{(l)} \log \frac{x_i^{(p)} - l\beta_i}{x_i^{(p)}}) = x_i^{(l)} \log \frac{x_i^{(p)} - l\beta_i}{x_i^{(p)}}) \tag{7}
$$

Therefore, the difference between these two IKL distances is:

$$
\begin{aligned}
&IKL(D_p, D_{l'}) - IKL(D_p, D_l) \\
&= \sum_{\beta_i \geq 0} (x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l'\beta_i} - x_i^{(p)} \log \frac{x_i^{(p)}}{x_i^{(p)} - l\beta_i}) \\
&+ \sum_{\beta_i < 0} (x_i^{(l')} \log \frac{x_i^{(p)} - l'\beta_i}{x_i^{(p)}} - x_i^{(l)} \log \frac{x_i^{(p)} - l\beta_i}{x_i^{(p)}}) \\
&= \sum_{\beta_i \geq 0} x_i^{(p)} \log(1 + \frac{(l'-l)\beta_i}{x_i^{(p)} - l'\beta_i}) + \sum_{\beta_i < 0} (x_i^{(l)} \log(1 - \frac{(l'-l)\beta_i}{x_i^{(p)} - l\beta_i}) \\
&+ (x_i^{(l')} - x_i^{(l)}) \log \frac{x_i^{(p)} - l'\beta_i}{x_i^{(p)}})
\end{aligned} \tag{8}
$$

Since $l < l' < W/S$, we have $l' - l > 0$ and when $\beta_i < 0$, $x_i^{(p)} - l\beta_i > 0$, $x_i^{(p)} - l'\beta_i > 0$, $x_i^{(l')} - x_i^{(l)} > 0 \Rightarrow IKL(D_p, D_{l'}) - IKL(D_p, D_l) > 0$. This completes the proof of the theorem.

4.2 Change Detection Algorithms

A significant distance to the reference window can signify a change in the distribution. However, there are cases of signal spikes, in which the distance is large for a short time period and then drops to normal. To avoid mistaking those spikes for distribution changes, Dasu et al. [9] report a change after seeing **p** consecutive large distances. Qahtan et al. [34] report a change if the current distance value significantly deviates beyond allowable change δ for a reasonable period χ from the history of the distance values. However, choosing optimal values for χ, δ in [34] and **p** in [9] is difficult in practice.

We observe that when the distribution changes, there are W/S windows overlapping the two distributions. The distances from these windows to the reference window (representative of the old distribution) are in an upward trend if measured by IKL as stated in Theorem 1. We exploit this property by storing M consecutive IKL distances in a buffer which is implemented as a queue as in line 7 in Algorithm 1. Every time the window slides, the distance d_{new} from the current window to the reference window is computed and appended to the buffer. If the buffer is full, the oldest distance $d_{expired}$ in the buffer is removed. The buffer is also cleared after a change is detected. We propose two algorithms, i.e., AVG and Dynamic LIS, to detect a change using the distance values stored in the buffer. We found in our empirical evaluation, a small buffer (i.e., $M = W/S$) is sufficient for both of our algorithms.

Algorithm 2 AVG

Global variables: *count*: Total number of windows from the beginning of distribution, *allAvg*: Overall average score in the buffer

```
function IsChangeAVG(buffer, AVG_th)
    curAvg = Average(buffer)
    allAvg = (allAvg * count + curAvg)/(count + 1)
    count = count + 1
    if curAvg > allAvg * AVG_th then
        count = 0
        return TRUE
    return FALSE
```

AVG. This algorithm is presented in Algorithm 2, which is an instance of Function IsChange() in Algorithm 1. A change is detected if the ratio between the average value of the distances stored in the buffer and the overall average distance is higher than a pre-defined threshold, *AVG_th*. We use *count* to denote the number of sliding windows from the beginning of the current distribution. Let *allAvg* denote the overall average distance value among those windows, and *curAvg* denote the current average distance in the buffer. When the window slides, *curAvg* is updated to incorporate the new distance as in line 2 in Algorithm 2.

$$
curAvg = \frac{curAvg \times M - d_{expired} + d_{new}}{M} \tag{9}
$$

The overall average value *allAvg* is also updated accordingly.

$$
allAvg = \frac{allAvg \times count + curAvg}{count + 1} \tag{10}
$$

Our proposed solution is different from Page-Hinckley test [29] which monitors single distances, used in [34] and [39]. In AVG, we monitor the average value of M consecutive distances to reduce the effect of signal spikes. The time complexity is low, O(1) for each sliding window. Although AVG requires a pre-defined threshold, i.e., *AVG_th*, similarly to [9, 34], it is more intuitive for the user. Our second approach does not require any manual threshold setting.

Dynamic LIS. As more data points arrive from the new distribution, the distance between the current window and the reference window also keeps increasing. As a result, we can measure the longest increasing subsequence (LIS) of distance values stored in the buffer, as an indication of distribution change. In this section, we present Dynamic LIS algorithm using this property. Given a sequence of real values: $arr = \{d_1, d_2, ..., d_K\}$, its subsequence can be formed by removing some elements without re-ordering the remaining ones. LIS is the longest subsequence of arr in which elements are in an increasing order. For example, given a sequence:

$$arr = \{0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15\}$$

Its LIS with the length of 6 is $\{0, 2, 6, 9, 11, 15\}$ because there is no other longer increasing subsequences. LIS has been widely studied and applied in a variety of domains, e.g., the system for aligning entire genomes [12]. Romik et al. [37] presented extensively the mathematics of LIS, e.g., the maximal LIS length in a random permutation.

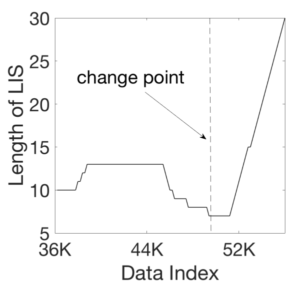


Figure 2: The LIS length with HPC Dataset

Figure 2 shows the LIS length of the distance sequence stored in the buffer using HPC dataset (complete description in Section 6). A change is synthetically introduced at the $50K^{th}$ data point. As illustrated in this figure, the LIS length is stable before the change and increases steadily afterwards. The pseudo code of Dynamic LIS is presented in Algorithm 3 and Function IsChangeDLIS() is an instance of function IsChange() in Algorithm 1. The algorithm relies on the increasing "trend" rather than the absolute distance value. Therefore, it can detect changes with various magnitudes. Furthermore, it does not require consecutive increasing values in the buffer by allowing a subsequence of increasing distance values. This is important because with a small slide size, the assumption on the same probability distribution between a slide and the entire window does not hold, the distance does not increase strictly when the window slides. A change is detected if the LIS length of the distances in the buffer is higher than a threshold, LIS_th.

Manually setting LIS_th can be dataset dependent and requires the domain knowledge of human experts. Therefore, we are motivated to design a dynamic thresholding strategy based on well-known LIS properties that have been extensively studied in the past [37]. For any n different numbers, $n \geq 1$, we denote δ_n to be a uniformly random permutation of those numbers, $L(\delta_n)$ to be the LIS length of δ_n, and $E(L(\delta_n))$ to be the expected value of $L(\delta_n)$.

Theorem 2 (Lower bound of LIS length). [37] *For all $n \geq 1$, we have: $E(L(\delta_n)) \geq \sqrt{n}$.*

Algorithm 3 Dynamic LIS

Global variables: *possibleLIS*: Upper bound of LIS length so far, initially set to 0.

```
function IsChangeDLIS(buffer, LIS_th)
    possibleLIS = possibleLIS + 1
    if possibleLIS ≤ LIS_th then return FALSE
    possibleLIS, lis = GETLIS(buffer)
    if lis > LIS_th then
        possibleLIS = 0; return TRUE
    return FALSE
```

```
function GETLIS(arr)
    n = arr.length
    vector   tail(n, 0)                    ▷ Initialized with 0
    vector   prev(n, −1)                   ▷ Initialized with -1
    len = 1
    for i = 1 to n − 1 do
        if arr[i] < arr[tail[0]] then
            tail[0] = i                    ▷ new smallest value
        else if arr[i] > arr[tail[len − 1]] then
                        ▷ arr[i] can extend largest subsequence
            prev[i] = tail[len − 1]
            tail[len + +] = i
        else
            pos =FindPosition(arr, tail, −1, len − 1, arr[i])  ▷
Use binary search to find the right position
            prev[i] = tail[pos − 1]
            tail[pos] = i
    return len
```

Theorem 3 (Limit of LIS length). [3] *As $n \to \infty$, we have: $E(L(\delta_n)) = 2\sqrt{n} + cn^{\frac{1}{6}} + o(n^{\frac{1}{6}})$ with $c \approx -1.77108$ and $\frac{E(L(\delta_n))}{\sqrt{(n)}} \to 2$.*

According to the above theorems, the expectation of the LIS length of a sequence of length n is asymptotically $2\sqrt{n}$. Therefore, it is unexpetected to observe a longer LIS. In our solution, to detect a change using distances in the buffer, we set $LIS\_th = 2\sqrt{M}$ accordingly.

Efficient LIS Computation. For a long sequence, the computation of LIS can be complex and time consuming. In [15], Fredman presented an algorithm to directly compute the LIS length of a sequence of numbers. For each element $arr[i]$, the algorithm finds the longest increasing subsequence ending at $arr[i]$ by using a binary search on the current LIS with the time complexity $O(logM)$. Therefore, with a buffer size M, the time complexity to compute LIS is $O(MlogM)$. To reduce the time for computing LIS for every sliding window, we use a variable *possibleLIS* to track the upper bound of LIS length. The actual LIS length is always smaller than or equal to *possibleLIS*. When the window slides, *possibleLIS* is incremented by 1. If *possibleLIS* is lower than LIS_th, no change is present. Only if *possibleLIS* is higher than LIS_th, the actual LIS is computed and then *possibleLIS* is updated accordingly.

5 OUTLIER DETECTION SOLUTION

Framework Design. We first present our framework design for outlier detection in non-stationary data streams. The pseudo code of our framework are presented in Algorithm 4. A model is first built with the initial window (Function TRAIN()), and it is used to detect outliers (Function OUTLIERDETECTION()) for subsequent windows until a change is detected. This reduces the number of model updates and re-evaluations for existing data points. When the window slides, and a change is detected (Function CHANGEDETECTION()), the model is rebuilt with the window starting from the change point. This ensures that the model is built to reflect only the new distribution. The reference window for change detection is also updated after a detected change.

Algorithm 4 Outlier Detection Solution

Input: Stream $\{o\}$, Window Size W, Slide Size S
Output: Outliers in every sliding window
Procedure:

```
1: D_r = {o_1, o_2, ..., o_W}; model = TRAIN(D_r)
2: OUTLIERDETECTION(model, D_r)   ▷ Detect outliers in the first window
3: for every sliding window D_t do
4:     if CHANGEDETECTION(D_t, D_r) then
5:         D_r = D_{t+W-S}            ▷ Update reference window
6:         model = TRAIN(D_r)    ▷ Build model with data of new distribution
7:         OUTLIERDETECTION(model, D_r)    ▷ Detect outliers in new distribution
8:     else
9:         OUTLIERDETECTION(model, {o_{t-S+1}, ..., o_t})   ▷ Detect outliers in the new slide
```

Outlier Detection Algorithms. This framework is applicable to various outlier detection algorithms. In general, after the outlier scores of data points are computed, outliers can be reported using different methods. If a threshold is used, the data points with a score higher than the threshold are reported as outliers [4]. Another approach is to report top $m\%$ of data points with the highest scores as outliers [1]. We adopt the latter approach to control outlier rate as well as to get an unbiased comparison between methods. In this study, to demonstrate the framework, we use PCA [1, 42] and One-class SVM technique [33, 36] considering their prevalence in practice. Now we describe how the outlier scores of data points are computed in PCA and One-class SVM.

PCA-based Outlier Detection. Assume data points have d dimensions. There are $k < d$ dimensions corresponding to the largest eigenvalues retaining the most data variance. Thus, the data points that have high deviation on the $d - k$ dimensions with small eigenvalues, can be considered as outliers. Let x'_{ij} be the projected value of data point x_i on the eigenvector e_j which has a small eigenvalue. The large deviation of x'_{ij} as compared to that of other data points suggests that x_i is an outlier. The outlier score of a data point x is measured by the normalized distance to the centroid μ along the principal components. That distance is weighted based on the eigenvalues as follows:

$$score(x) = \sum_{i=1}^{d} \frac{|(x-\mu)e_i|^2}{\lambda_i} \tag{11}$$

where d is the number of dimensions, e_i and λ_i are the i^{th} eigenvector and the corresponding i^{th} largest eigenvalue, respectively.

One-class SVM Outlier Detection. One-class SVM [1, 20] can be used for outlier detection – given a set of samples, it will detect the soft boundary of that set to classify data points to normal or outlier class. One-class SVM can be viewed as a regular two-class SVM [46] where all the training data lies in the first class, and the origin is taken as the only member of the second class. Thus, the linear decision boundary corresponds to the classification rule:

$$f(x) = \langle w, x \rangle + b \tag{12}$$

where w is the normal vector and b is a bias term. One-class SVM solves an optimization problem to find the rule f with maximal geometric margin. An outlier corresponds to $f(x) < 0$. To allow a nonlinear decision function, we can use a kernel function [20] such as linear, polynomial and Gaussian kernels to project input data into a feature space.

Timeliness Evaluation for Change Detection. Precision, recall, and F1-score are commonly used metrics for generic classification tasks. In a streaming setting, prompt detection of changes is crucial as the outlier detection model can be updated for the new distribution. For the O-NSD problem, the detected change point should be close to the actual change point. However, the standard metrics such as precision, recall and F1-score, do not quantify how timely the changes are detected. In [9, 34], the authors consider a detection accurate if the last data point of the detected window is less than two windows away from to the actual change point. This approach uses a hard-coded threshold, i.e., 2 windows length, the choice of which is not intuitive. Furthermore, this approach does not quantify the timeliness smoothly. In the studies of the quickest change detection problem [5, 45], the average delay of the detected change point is used. However, it does not distinguish true and false alarms. In this study, we propose three *weighted* measures, i.e., $wPrecision$, $wRecall$ and $wF1$ which incorporate the timeliness of change detection into the commonly used precision, recall and F1-score metrics, respectively. In order to define the new measures, we first define the timeliness score of one detected change point which is considered as the last data point of the detected window. If there are more than one detected change points for the same actual change point, the closest change point has a positive score, and the others are considered false positive detections with the score of 0. Assume a change is detected at window D_n, and the actual change point is o_a. The score of D_n depends on the number of disjoint windows passed after o_a:

$$score(D_n) = e^{-\lambda \lfloor \frac{n-a}{W} \rfloor} \tag{13}$$

where λ is a decay factor, $0 < \lambda < 1$. The parameter λ can be set by the users to control how fast the score decays with time. The scores represent the utility of change detection and can be used to measure the sensitivity of algorithms to various slide sizes. Furthermore, we have $0 < score(D_n) \leq 1$ for $\forall n \geq a$. The score decreases as the distance from the detected window to the actual change point increases. For example, with $\lambda = 0.1$, $D_n = o_a + 4W/3$ as in Figure 3, D_n has score 0.9.

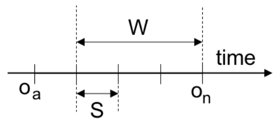


Figure 3: Detected change point o_n has a score of 0.9

Specifically, assume there are A actual change points and K detected windows with scores: $score_1, score_2, ..., score_K$. We define $wPrecision$, $wRecall$, $wF1$ as follows:

$$wPrecision = \frac{\sum_{i=1}^{K} score_i}{K}, \ wRecall = \frac{\sum_{i=1}^{K} score_i}{A} \quad (14)$$

The metrics $wPrecision$ and $wRecall$ measure the correctness and the sufficiency of detected changes, respectively. The metric $wF1$ is their harmonic mean.

$$wF1 = \frac{2 \times wPrecision \times wRecall}{wPrecision + wRecall} \quad (15)$$

The values of weighted measures, $wPrecision, wRecall$ and $wF1$ range from 0 to 1, similarly to the standard metrics. The standard metrics are upper bounds of the weighted measures.

Time and Space Analysis. We analyze the time and space complexities of our framework with the aforementioned outlier classifiers, i.e., PCA and One-class SVM using a linear kernel. The time complexity of our overall framework depends on the following three main procedures. For each window W of d-dimensional data: 1) outlier detection model building costs $O(d^2W + d^3)$ for PCA and $O(dW^3)$ for One-class SVM [31], 2) outlier score computation for one slide and sorting for an entire sliding window cost $O(d^2S + W \log W)$ for PCA and $O(dS + W \log W)$ for One-class SVM, and 3) change detection time includes applying PCA for the reference window that costs $O(d^2W)$, incremental update of estimated histogram that costs constant time, and LIS computation that costs $O(M \log M) = O(\frac{W}{S} \log(\frac{W}{S}))$ as we set $M = W/S$ in our evaluation. In summary, suppose that there is one change after N sliding windows, $N > 1$, the average time complexity for one window is $O(\frac{d^2W+d^3}{N} + d^2S + W \log W)$ for PCA and $O(\frac{dW^3+d^2W}{N} + dS + W \log W)$ for One-class SVM. This shows that the change detection module does not incur much overhead compared to outlier detection module. The space cost of our framework depends on the following four main components: 1) the original and transformed data in the current window, that cost $O(dW)$, 2) the reference window that costs $O(dW)$, 3) the distances in the buffer that cost $O(\frac{W}{S})$, and 4) the estimated histograms that cost $O(dW)$. In total, the framework requires $O(d + \frac{1}{S})W$ space.

6 EXPERIMENTS

6.1 Experimental Methodology

Our change detection methods, i.e., Dynamic LIS and AVG are compared with the method in [9] which we refer as KL-Bootstrap and the state of the art change detection method PHDT [34]. The outlier detection framework is compared with the incremental approach [2, 8] which updates the model after every new slide arrives in an approximate manner, retrain based approach which updates model periodically, and static approach which does not update the model. The algorithms are implemented in Python with Scikit-learn package [31] and our source codes are available online[3]. Experiments are conducted on a Linux machine with 4 cores 2.7GHz and 24GB memory.

Synthetic Datasets. We generate synthetic 2 dimensional datasets with changes in mean, standard deviation or correlation. In these datasets, for the first distribution, the mean, standard deviation, correlation coefficient are set to 0.01, 0.2, 0.5, respectively. After each l samples, we create a change in 1 randomly selected dimension for changing mean or standard deviation and in 2 randomly selected dimensions for changing correlation by adding ϵ to the distribution's parameters. Our synthetic datasets are generated similarly as in [9, 34]. We control the length of distributions l to be 50000 in the change detection experiments and draw a random number between 25000 to 100000 in the outlier detection experiments.

Real-world Datasets. We use 4 real-world datasets: 1) *TAO* has 3 attributes and is available at Tropical Atmosphere Ocean project [19], 2) *Forest Cover* (FC) has 55 attributes and we use the first 10 continuous attributes, 3) *HPC* has 7 attributes, extracted from the Household Electric Power Consumption dataset, 4) *EM* has 16 attributes from Gas Sensor Array dataset. The last three datasets are available at the UCI KDD Archive [14]. Because the ground-truth of distribution changes is not available, we simulate artificial changes as in [40]. Specifically, for each dataset, we sample batches of 50000 data points. For each batch, one random dimension is selected to apply one of two change types: 1) *Gauss-1D*: the batch is added a random Gaussian variable with 10% of mean and variance of the batch, respectively; 2) *Scale-1D*: each value of one randomly selected dimension is doubled. We append "G" to the dataset name to indicate added Gauss-1D changes and "S" for Scale-1D changes.

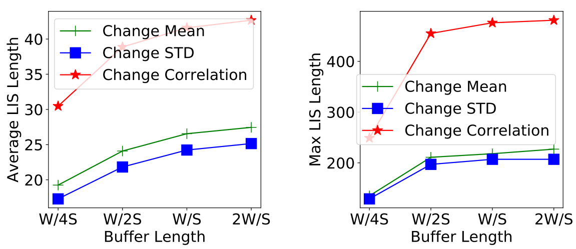
Default Parameters. The window size W and slide size S are set to 10000 and 20, respectively. The default change ϵ is 0.03 for mean, 0.2 for standard deviation and 0.1 for correlation when creating changes for synthetic datasets. The default values of χ and δ for PHDT are set to 500 and 5.10^{-3}, as in [34]. For KL-Bootstrap [9], the number of consecutive high distances **p** is set to $\max(5, 0.05 * W/S)$. We use 500 bootstrap samples to get high threshold corresponding to each distribution similar to [9]. The buffer size M is set to W/S and AVG_th is set to 1.5. The default outlier rate is 0.05. The decay factor λ in the evaluation metric is set to 0.1.

6.2 Change Detection

Buffer Length Selection. In this experiment, we examine the trend of LIS length in the buffer when varying the buffer length. After a change, the reference window is updated when the current window entirely is in a new distribution for the first time. With

[3]https://goo.gl/Vq4kgu

more distance values in the buffer, the number of values which are in an increasing order can be larger. Figure 4 shows the average and maximal LIS length in the buffer when varying the buffer length from $\frac{W}{4S}$ to $\frac{2W}{S}$ by changing mean, standard deviation, and correlation. As we can see in the figure, the average and maximal LIS length increase when the buffer length is increased, however not linearly with the buffer size. With $M = \frac{W}{S}$, we can get high average LIS length and maximal LIS length, comparable to $M = \frac{2W}{S}$. It is because most of the high distances in the buffer are the distances from the sliding windows overlapping the two distributions and there are $\frac{W}{S}$ such windows for each change. It confirms our choice of buffer length which is $\frac{W}{S}$.



(a) Average LIS Length (b) Maximum LIS Length

Figure 4: Varying Buffer Length

Parameter Studying. In many applications, early and accurate change detection is preferable. We vary AVG_th from 1.1 to 2 to examine the utility of the detected changes. When AVG_th increases, the criteria for detecting a change is stricter. Table 1 shows the $wPrecision - wRecall - wF1$ of AVG for different change types with the synthetic datasets. As we can see in this table, when AVG_th increases, the $wPrecision$ first increases as the condition is stricter and there are less false detections. With $AVG\_th = 1.5$, AVG offers the highest $wF1$ in most cases. When AVG_th is further increased, although the detected changes are more precise, the delay of the detected change points compared to the actual changes is larger, $wPrecision$ decreases. Also, since there are fewer detected changes, $wRecall$ decreases. As we can see from the table, when $AVG\_th \geq 1.7$, $wF1$ decreases.

AVG_th	Mean	STD	Correlation
1.1	0.76-0.93-0.84	0.69-0.94-0.80	0.64-0.96-0.76
1.3	0.91-0.95-0.93	0.99-0.99-0.99	**0.98-0.98-0.98**
1.5	**0.95-0.95-0.95**	**1.0-1.0-1.0**	0.96-0.93-0.95
1.7	0.98-0.50-0.66	0.92-0.92-0.92	0.98-0.78-0.87
2	0.90-0.45-0.60	0.92-0.29-0.44	0.94-0.74-0.83

Table 1: Varying AVG_th with Synthetic Datasets

Change Magnitude Sensitivity. We vary the change magnitude ϵ from 0.01 to 0.05 for mean, from 0.05 to 0.5 for standard deviation and from 0.01 to 0.2 for correlation coefficient. Table 2 shows the $wF1$s of PHDT, KL-Bootstrap, AVG and Dynamic LIS. As can be seen in this table, when ϵ increases, $wF1$s of all the algorithms increase because of higher difference between distributions. Dynamic LIS and AVG offer the highest $wF1$s in most cases. When $\epsilon \geq 0.04$ for changing mean, $\epsilon \geq 0.3$ for changing standard deviation and $\epsilon \geq 0.15$ for changing correlation, AVG and Dynamic LIS detect changes perfectly. Especially, Dynamic LIS performs better than the others with small ϵ because it does not rely on the absolute increase in distances. KL-Bootstrap which utilizes KL distance does not perform well with changing correlation in most the cases because it performs detection on original data in which a change in correlation may not result in a high KL distance.

Change	ϵ	0.01	0.02	0.03	0.04	0.05
Mean	PHDT	0.38	0.58	0.67	0.87	0.9
	KL-Bootstrap	0.54	0.84	0.89	0.95	0.94
	AVG	**0.58**	0.85	0.94	**1.0**	**1.0**
	DLIS	0.56	**0.96**	**1.0**	**1.0**	**1.0**
STD	ϵ	0.05	0.1	0.2	0.3	0.5
	PHDT	0.21	0.86	0.93	0.98	**1.0**
	KL-Bootstrap	0.44	0.87	0.9	0.95	**1.0**
	AVG	0.44	**0.90**	**1.0**	**1.0**	**1.0**
	DLIS	**0.47**	0.86	0.9	0.99	**1.0**
Corr	ϵ	0.01	0.05	0.1	0.15	0.2
	PHDT	0.15	0.71	0.84	0.96	**1.0**
	KL-Bootstrap	0.05	0.40	0.41	0.43	0.46
	AVG	0.28	0.85	**0.97**	**1.0**	**1.0**
	DLIS	**0.45**	**0.89**	0.93	**1.0**	**1.0**

Table 2: Varying Change Magnitude - $wF1$

Slide Size Sensitivity. We vary the slide size from 1 to 400. When the slide size increases, the difference between sliding windows is more significant, however the number of slides overlapping two distributions (W/S) decreases. Therefore, there is a smaller number of significant increases in distance to the reference window when the distribution is changing. Table 3 shows the $wF1$s of PHDT, KL-Bootstrap, AVG, and Dynamic LIS. As can be seen in this table, when the slide size increases from 1 to 400, AVG and Dynamic LIS offer better performance than PHDT and KL-Bootstrap in all cases. AVG offers the most stable $wF1$s because it is less affected by the reduction of the buffer size. When the slide size increases, the $wF1$ of Dynamic LIS decreases slightly for most cases because the expected LIS length holds when $n \to \infty$ (Theorem 3).

Change Detection with Real-world Datasets. Table 4 shows the $wF1$ of PHDT, AVG and Dynamic LIS with the 4 real-world datasets and 2 types of change, i.e., Gauss-1D and Scale-1D. As can be seen in this table, Dynamic LIS and AVG offer the highest $wF1$s in most cases. Especially, Dynamic LIS shows robust performance in all cases.

Scalability Comparison. Figure 5 shows the running time of the

Change	Slide Size	1	20	100	200	400
Mean	PHDT	0.67	0.68	0.66	0.64	0.69
	KL-Bootstrap	0.90	0.89	0.88	0.88	0.87
	AVG	**0.97**	0.96	0.94	**0.97**	0.96
	DLIS	0.95	**0.97**	**0.97**	**0.97**	**0.97**
STD	PHDT	0.92	0.90	0.91	0.90	0.89
	KL-Bootstrap	0.95	**1.0**	0.91	0.91	0.88
	AVG	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	DLIS	0.96	0.94	0.92	0.91	0.89
Corr	PHDT	0.84	0.83	0.85	0.87	0.84
	KL-Bootstrap	0.42	0.35	0.45	0.42	0.39
	AVG	**0.98**	**0.97**	**0.97**	**0.98**	**0.96**
	DLIS	0.89	0.91	0.88	0.88	0.87

Table 3: Varying Slide Size - $wF1$

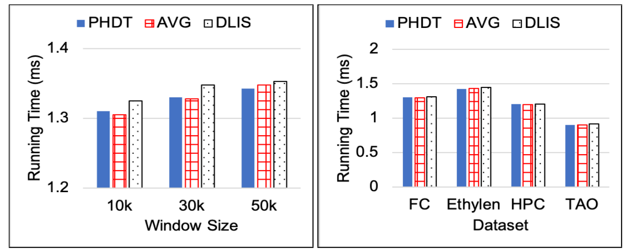
Change	Dataset	TAO	Ethylen	HPC	FC
Scale 1D	PHDT	0.88	0.88	0.88	**0.91**
	KL-Bootstrap	0.58	0.89	0.75	0.78
	AVG	0.88	**0.91**	0.89	**0.91**
	DLIS	**0.89**	**0.91**	**0.94**	**0.91**
Gauss 1D	PHDT	0.78	0.96	0.70	0.72
	KL-Bootstrap	0.82	0.90	0.55	0.74
	AVG	0.82	0.96	0.73	0.80
	DLIS	**0.85**	**0.97**	**0.89**	**0.89**

Table 4: Change Detection with Real-world Datasets - $wF1$

algorithms with different window sizes with FC dataset and different dimensions by using different datasets. We do not include KL-Bootstrap here because it requires training data and the sampling process to compute the high threshold ($\approx 10ms$ for one sliding window), thus the running time of KL-Bootstrap is much larger than the others. When the window size increases, the time for creating the reference window increases and the time for updating histogram is not change. As can be seen in this figure, when the window size increases from 10k to 50k, the running time increases slightly because the number of reference window re-computations is not significant. When the dimensions of the data increases, the PCA transforming operation takes more time. As we can see in the figure, with the Ethylen dataset, the running time of all the algorithms is the highest since it has the highest number of dimensions. In both cases, all three algorithms have comparable running time, showing our DLIS which requires LIS computation does not incur much overhead.

6.3 Outlier Detection

Since Dynamic LIS and AVG show robust performance across datasets and does not require any threshold setting, we adopt these methods in the outlier detection framework. We demonstrate the efficiency of our proposed framework by using PCA-based and One-class SVM classifers. Our methods including DLIS SVM, DLIS PCA, AVG SVM and AVG PCA are compared with the static approaches including Static SVM, Static PCA which do not update the model at all and re-train based approaches including Retrain SVM,



(a) Varying Window Size (b) Varying Dimensions

Figure 5: Change Detection Running Time Per Window (ms)

Retrain PCA which update the model periodically after every t data points. For PCA, we also compare with Incremental PCA [38] which updates the PCA model in every slide. Incremental approach for One-class SVM is not available, however. We compared the results of the methods with the ground truth retrieved by applying the algorithms, i.e., PCA and One-class SVM to the entire ground truth distributions. The overall F1-score is averaged over all windows. In this experiment, we generate synthetic datasets with 5 dimensions and each distribution has a length which is randomly generated in the range from 25000 to 100000.

F1-score. Tables 5 and 6 show the F1-score of One-class SVM and PCA approaches, respectively, with the real-world and synthetic datasets. As can be seen from these tables, the change detection approach offers the highest F1-scores. DLIS SVM and DLIS PCA have comparable F1-scores with AVG SVM and AVG PCA, respectively. Static One-class SVM and Static PCA offer the lowest F1-scores because the models are outdated when the distribution changes. Incremental PCA incrementally updates the model. However, it does not adapt quickly to the change of distribution and the model is built with data from multiple distributions. Incremental PCA offers higher F1-scores than Static PCA but lower than the other approaches. In the retrain-based approach, the data used for model building can overlap two distributions and it can result in a wrong detection. As we can see from these tables, in Retrain PCA and

Dataset	DLIS SVM	AVG SVM	Static SVM	Retrain SVM		
				5k	10k	20k
Mean	**0.95**	0.94	0.01	0.87	0.85	0.79
STD	**0.96**	0.93	0.69	0.95	0.94	0.92
Corr	**0.97**	0.96	0.83	0.96	0.95	0.93
FC-G	**0.93**	0.92	0.73	**0.93**	0.92	0.89
Ethyl-G	**0.94**	0.93	0.52	0.93	0.93	0.89
HPC-G	**0.94**	0.93	0.62	0.93	0.92	0.90
TAO-G	0.90	**0.91**	0.42	0.83	0.81	0.74
FC-S	**0.91**	**0.91**	0.48	0.90	0.88	0.85
Ethyl-S	0.95	**0.96**	0.49	0.93	0.92	0.87
HPC-S	0.89	**0.94**	0.78	0.90	0.89	0.87
TAO-S	0.87	**0.90**	0.45	0.84	0.82	0.76

Table 5: One-class SVM Outlier Detection - F1-score

Dataset	DLIS PCA	AVG PCA	Static PCA	Retrain PCA 5k	Retrain PCA 10k	Retrain PCA 20k	Incr. PCA
Mean	**0.94**	0.92	0.87	0.92	0.91	0.90	0.89
STD	**0.93**	0.91	0.86	0.89	0.88	0.87	0.82
Corr	**0.90**	0.89	0.64	0.88	0.87	0.85	0.70
FC-G	**0.91**	0.88	0.48	0.89	0.87	0.82	0.72
Ethyl-G	**0.94**	0.90	0.46	0.92	0.89	0.83	0.72
HPC-G	**0.95**	0.87	0.55	0.92	0.90	0.86	0.78
TAO-G	**0.92**	0.89	0.62	0.89	0.87	0.84	0.79
FC-S	0.92	**0.93**	0.43	0.88	0.86	0.85	0.84
Ethyl-S	**0.90**	0.89	0.19	0.87	0.84	0.75	0.62
HPC-S	0.91	**0.92**	0.70	0.89	0.88	0.86	0.82
TAO-S	0.91	**0.92**	0.78	0.88	0.86	0.85	0.88

Table 6: PCA Outlier Detection - F1-score

Retrain SVM, when t is increased from 5k to 20k, the F1-score decreases because the model is updated less frequently.

Stability with Outlier Rate. We compare the approaches when varying the outlier rate r from 0.01 to 0.1. Tables 7 and 8 show the F1-score of all the approaches with the FC dataset and Gaussian-1D change injection. As can be seen in these tables, the F1-scores of all the methods increase when the outlier rate increases because there is higher chance a local outlier, i.e., the outlier in a sliding window, can be a global outlier which is an outlier in the entire distribution. DLIS PCA and DLIS SVM offer the highest F1-scores in all cases.

r	DLIS PCA	AVG PCA	Static PCA	Retrain PCA 5k	Retrain PCA 10k	Retrain PCA 20k	Incr. PCA
0.01	**0.86**	**0.86**	0.41	0.83	0.81	0.74	0.77
0.05	**0.91**	0.90	0.43	0.87	0.85	0.78	0.84
0.1	**0.92**	0.91	0.55	0.89	0.87	0.82	0.87

Table 7: PCA-based Outlier Detection with FC-G Dataset - F1-score - Varying Outlier Rate

r	DLIS SVM	AVG SVM	Static SVM	Retrain SVM 5k	Retrain SVM 10k	Retrain SVM 20k
0.01	**0.90**	0.89	0.5	0.84	0.83	0.81
0.05	**0.93**	0.91	0.67	0.86	0.85	0.84
0.1	**0.93**	0.9	0.69	0.87	0.86	0.85

Table 8: One-class SVM Outlier Detection with FC-G Dataset - F1-score - Varying Outlier Rate

Running Time. Tables 9 and 10 show the average running time for each sliding window in milliseconds of the approaches with the synthetic and real-word datasets. As depicted in these tables, in the retrain-based approach, with higher t, it incurs less running time because it rebuilds the model less frequently. With $t = 5k$, it requires the highest running time. Incremental PCA updates the model in every slide and the entire window is re-evaluated, thus requires the highest running time. AVG PCA and AVG SVM incur the smallest running time. It is because the change detection module does not incur much running time compared to model building and outlier detection and the number of model building is less than the other approaches. DLIS SVM and DLIS PCA require slightly higher running time than AVG SVM and AVM PCA, respectively, because of the LIS computation. With datasets TAO-G and TAO-S which have the fewest dimensions, all the methods incur the least running time.

Dataset	DLIS SVM	AVG SVM	Retrain SVM 5k	Retrain SVM 10k	Retrain SVM 20k
Mean	1.96	**1.90**	5.20	3.08	2.00
STD	1.88	**1.87**	4.92	3.08	2.04
Corr	1.88	**1.75**	4.80	2.96	2.12
FC-G	2.52	**2.45**	6.68	3.88	3.00
Ethyl-G	2.72	**2.50**	6.92	4.24	2.80
HPC-G	2.56	2.45	5.60	3.24	**2.08**
TAO-G	1.64	**1.60**	4.36	2.56	1.76
FC-S	2.16	**2.12**	4.84	2.60	2.40
Ethyl-S	2.40	**2.35**	6.48	3.72	2.40
HPC-S	1.84	**1.75**	5.40	3.20	2.04
TAO-S	1.60	**1.44**	4.40	2.56	1.68

Table 9: One-class SVM Outlier Detection - Running Time Per Window(ms)

Dataset	DLIS PCA	AVG PCA	Retrain PCA 5k	Retrain PCA 10k	Retrain PCA 20k	Incr. PCA
Mean	5.72	**5.53**	8.40	7.20	6.00	35.44
STD	5.64	**5.55**	8.20	7.40	6.12	35.64
Corr	5.20	**5.17**	7.40	7.00	5.88	29.84
FC-G	4.36	**4.33**	6.28	5.60	4.56	23.48
Ethyl-G	4.40	**4.38**	6.40	5.48	4.72	23.96
HPC-G	4.64	**4.60**	6.48	5.60	4.84	26.00
TAO-G	4.00	**3.77**	5.20	4.84	4.28	24.80
FC-S	6.12	**6.01**	10.68	9.64	7.44	42.00
Ethyl-S	4.80	**4.72**	7.24	6.20	5.24	27.08
HPC-S	4.60	**4.55**	6.40	5.64	4.88	27.56
TAO-S	4.52	**4.40**	5.32	4.76	4.60	25.52

Table 10: PCA Outlier Detection - Running Time Per Window(ms)

7 CONCLUSIONS

In this paper, we introduced a framework for outlier detection in non-stationary data streams by incorporating distribution change detection to trigger model updates. To detect changes in distribution accurately, we proposed to compute the distance of the current window to the reference window using a new IKL measure, and two change detection algorithms which monitor the distance values. Our AVG method exploits the increase in the average distance to overcome temporary spikes, and Dynamic LIS method exploits the increasing trend of the distance values and is parameter-free. We demonstrated our outlier detection framework with two popular classifiers, i.e., PCA and One-class SVM. Our time and space analysis showed that the incorporated change detection does not incur much overhead. The experiment results showed that AVG and Dynamic LIS offer robust performance for change detection . Our proposed framework provides highly accurate outlier detection, requires significantly less running time than the incremental and retrain based approaches. The combination of AVG and Dynamic LIS can be exploited in the future work to achieve a higher accuracy in change detection.

8 ACKNOWLEDGMENTS

This work has been supported in part by NSF CNS-1915828, the USC Integrated Media Systems Center, and unrestricted cash gifts from Oracle and Google. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Charu C Aggarwal. 2015. Outlier analysis. In *Data mining*. Springer, 237–263.

[2] R. Arora, A. Cotter, K. Livescu, and N. Srebro. 2012. Stochastic optimization for PCA and PLS. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 861–868. https://doi.org/10.1109/Allerton.2012.6483308

[3] Jinho Baik, Percy Deift, and Kurt Johansson. 1999. On the distribution of the length of the longest increasing subsequence of random permutations. *Journal of the American Mathematical Society* 12, 4 (1999), 1119–1178.

[4] Azzeddine Bakdi and Abdelmalek Kouadri. 2017. A new adaptive PCA based thresholding scheme for fault detection in complex systems. *Chemometrics and Intelligent Laboratory Systems* 162 (2017), 83–93.

[5] Taposh Banerjee and Venugopal V Veeravalli. 2013. Data-efficient quickest change detection in minimax settings. *IEEE Transactions on Information Theory* 59, 10 (2013), 6917–6931.

[6] Alka Bhushan, Monir H Sharker, and Hassan A Karimi. 2015. Incremental principal component analysis based outlier detection methods for spatiotemporal data streams. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 2 (2015), 67–71.

[7] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 443–448.

[8] Matthew Brand. 2002. Incremental Singular Value Decomposition of Uncertain Data with Missing Values. In *Proceedings of the 7th European Conference on Computer Vision-Part I (ECCV '02)*. Springer-Verlag, London, UK, UK, 707–720. http://dl.acm.org/citation.cfm?id=645315.649157

[9] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. 2006. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*.

[10] Anh Hoang Dau, Victor Ciesielski, and Andy Song. 2014. Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class.. In *SEAL*. 311–322.

[11] Nicolas Delannay, Cédric Archambeau, and Michel Verleysen. 2008. Improving the robustness to outliers of mixtures of probabilistic PCAs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 527–535.

[12] Arthur L. Delcher, Simon Kasif, Robert D. Fleischmann, Jeremy Peterson, Owen White, and Steven L. Salzberg. 1999. Alignment of whole genomes. *Nucleic Acids Research* 27, 11 (1999), 2369–2376. https://doi.org/10.1093/nar/27.11.2369

[13] Jeremiah D Deng. 2016. Online outlier detection of energy data streams using incremental and kernel PCA algorithms. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE, 390–397.

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[15] Michael L Fredman. 1975. On computing the length of longest increasing subsequences. *Discrete Mathematics* 11, 1 (1975), 29–35.

[16] David Freedman and Persi Diaconis. 1981. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 57, 4 (1981), 453–476.

[17] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. 2005. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 401–410.

[18] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 44.

[19] SP Hayes, LJ Mangum, Joël Picaut, A Sumi, and K Takeuchi. 1991. TOGA-TAO: A moored array for real-time measurements in the tropical Pacific Ocean. *Bulletin of the American Meteorological Society* 72, 3 (1991), 339–347.

[20] Katherine Heller, Krysta Svore, Angelos D Keromytis, and Salvatore Stolfo. 2003. One class support vector machines for detecting anomalous windows registry accesses. (2003).

[21] S. Y. Huang, J. W. Lin, and R. H. Tsaih. 2016. Outlier detection in the concept drifting environment. In *2016 International Joint Conference on Neural Networks (IJCNN)*. 31–37. https://doi.org/10.1109/IJCNN.2016.7727177

[22] Ruoyi Jiang, Hongliang Fei, and Jun Huan. 2011. Anomaly Localization for Network Data Streams with Graph Joint Sparse PCA. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. ACM, New York, NY, USA, 886–894. https://doi.org/10.1145/2020408.2020557

[23] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 180–191.

[24] Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

[25] L. I. Kuncheva. 2013. Change Detection in Streaming Multivariate Data Using Likelihood Detectors. *IEEE Transactions on Knowledge and Data Engineering* 25, 5 (May 2013), 1175–1180.

https://doi.org/10.1109/TKDE.2011.226
[26] R. Lasaponara. 2006. On the use of principal component analysis (PCA) for evaluating interannual vegetation anomalies from SPOT/VEGETATION {NDVI} temporal series. *Ecological Modelling* 194, 4 (2006), 429 – 434. https://doi.org/10.1016/j.ecolmodel.2005.10.035
[27] Junshui Ma and Simon Perkins. 2003. Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, Vol. 3. IEEE, 1741–1745.
[28] Paulo M Mafra, Vinicius Moll, Joni da Silva Fraga, and Altair Olivo Santin. 2010. Octopus-IIDS: An anomaly based intelligent intrusion detection system. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*. IEEE, 405–410.
[29] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi. 2004. Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system. In *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*, Vol. 2. 815–818 Vol.2.
[30] H Vieira Neto and Ulrich Nehmzow. 2005. Incremental PCA: An alternative approach for novelty detection. *Towards Autonomous Robotic Systems* (2005).
[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[32] Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. 2009. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer networks* 53, 6 (2009), 864–881.
[33] Roberto Perdisci, Guofei Gu, and Wenke Lee. 2006. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 488–498.
[34] Abdulhakim A. Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. 2015. A PCA-Based Change Detection Framework for Multidimensional Data Streams: Change Detection in Multidimensional Data Streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 935–944. https://doi.org/10.1145/2783258.2783359
[35] Marcos Quiñones-Grueiro and Cristina Verde. 2017. Comments on the applicability of âĂIJAn improved weighted recursive PCA algorithm for adaptive fault detectionâĂİ. *Control Engineering Practice* 58 (2017), 254–255.
[36] S. Rajasegarar, C. Leckie, J. C. Bezdek, and M. Palaniswami. 2010. Centered Hyperspherical and Hyperellipsoidal One-Class Support Vector Machines for Anomaly Detection in Sensor Networks. *IEEE Transactions on Information Forensics and Security* 5, 3 (Sept 2010), 518–533. https://doi.org/10.1109/TIFS.2010.2051543
[37] Dan Romik. 2015. *The surprising mathematics of longest increasing subsequences*. Vol. 4. Cambridge University Press.
[38] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. 2008. Incremental learning for robust visual tracking. *International journal of computer vision* 77, 1-3 (2008), 125–141.
[39] Raquel Sebastiao and Joao Gama. 2009. A study on change detection methods. In *Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA*. 12–15.
[40] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. 2007. Statistical Change Detection for Multidimensional Data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. ACM, New York, NY, USA, 667–676. https://doi.org/10.1145/1281192.1281264
[41] Herbert A Sturges. 1926. The choice of a class interval. *Journal of the american statistical association* 21, 153 (1926), 65–66.
[42] Marina Thottan and Chuanyi Ji. 2003. Anomaly detection in IP networks. *IEEE Transactions on signal processing* 51, 8 (2003), 2191–2204.
[43] Michael E Tipping and Christopher M Bishop. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61, 3 (1999), 611–622.
[44] Luan Tran, Liyue Fan, and Cyrus Shahabi. 2016. Distance-based Outlier Detection in Data Streams. *Proc. VLDB Endow.* 9, 12 (Aug. 2016), 1089–1100. https://doi.org/10.14778/2994509.2994526
[45] Venugopal V Veeravalli and Taposh Banerjee. 2014. Quickest change detection. In *Academic Press Library in Signal Processing*. Vol. 3. Elsevier, 209–255.
[46] Jason Weston and Chris Watkins. 1998. *Multi-class support vector machines*. Technical Report. Citeseer.
[47] Yang Zhang, Nirvana Meratnia, and Paul J. M. Havinga. 2010. Ensuring High Sensor Data Quality Through Use of Online Outlier Detection Techniques. *Int. J. Sen. Netw.* 7, 3 (May 2010), 141–151. https://doi.org/10.1504/IJSNET.2010.033116