Outlier Detection in Non-stationary Data Streams

Luan Tran * Liyue Fan † Cyrus Shahabi ‡

Abstract

Continuous outlier detection in data streams has important applications in many domains, such as in climate science, transportation, and energy. 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 metric based on Kullback-Leibler distance, two accurate change detection algorithms as well as new evaluation metrics that quantify the timeliness of the detected changes. We demonstrate the outlier detection framework with two outlier models, i.e., Principal Component Analysis and One-class Support Vector Machine. Our extensive experiments with real-world and synthetic datasets show that our change detection algorithms outperform the state-of-the-art PHDT which employs Page Hinckley test for change detection. Our O-NSD framework offers higher accuracy and requires a much shorter running time than retrain-based and incremental update approaches.

Key words: outlier detection, streaming data, non-stationary

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, as it is hard to get labeled data, most outlier detection techniques are unsupervised. In the literature, researchers introduced many modeling techniques, e.g., proximity-based models [29], linear models such as Principal Component Analysis (PCA) [1], One-class Support Vector Machine [31].

Typically, with data streams, outlier detection is performed over sliding windows. The model which is trained by the first window is only applicable to the data 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 changes. 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 in 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 (which are circled) are likely outliers in the first day and are normal in the second day; the wind speeds higher than 7 m/s are normal in the first day but are outliers in the second day. As a result, the model built from data in the first day is not suitable for the second day. A baseline approach to

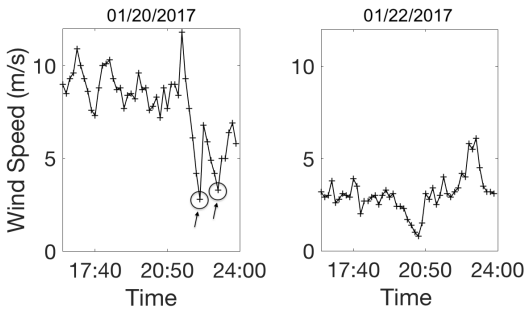


Figure 1: Wind speed data on two days

tackle this challenge is to rebuild or incrementally update [4] the model in every sliding window, which is computationally expensive. Another method is to rebuild the model after a fixed time interval [31], which is hard to optimize because changes are usually unpredictable

*Computer Science Department, University of Southern California, Los Angeles, California. Email: luantran@usc.edu

†University at Albany, SUNY, Albany, New York. Email: liyuefan@albany.edu

‡Computer Science Department, University of Southern California, Los Angeles, California. Email: shahabi@usc.edu

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

in practice. In this paper, we promote the *change detection* approach which rebuilds the model only when the underlying distribution changes in the data streams. Intuitively, as more data points from the new distribution arrive, the difference between the current window and the reference window which is from the old distribution tends to increase. Therefore, we can monitor the sequence of the distance measures 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 [18] typically use multiple outlier classifiers. The contributions of our study are summarized as follows:

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

(2) We propose a distance metric IKL to measure the difference between two distributions. We prove that the distance between the current window and the reference window which is used for model building increases when the underlying distribution changes.

(3) We propose two algorithms for change detection utilizing our proposed distance metric, 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 based on the length of the distance buffer. Our experiments show that Dynamic LIS and AVG are superior to the state-of-the-art approach for change detection.

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

(5) With extensive experiments on 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.

2 Related Work

Outlier Detection Principal Component Analysis [1] is a technique for outlier detection. It has been applied in various domains, such as network intrusion detection [28] and in space craft components [13]. Incremental PCA has been used in visual novelty detection mechanism [20], outlier detection in energy data streams [11], and spatial-temporal data in [6]. One-class SVM is a technique for outlier detection which is widely applied to wireless sensor network [31] and time-series data [17]. To the best of our knowledge, algorithm to incrementally update One-class SVM is not available.

Change Detection Change detection in data streams is first studied for one dimensional data [7, 14, 27, 30]. Kifer et al. [14] proposed a family of distance measures for change detection. In [27], Takeuchi and Yamanishi proposed an incremental AR model to model a sequence of data. In [7], Bifet and Gavalda proposed to use variable length windows to detect changes in distribution. Abrupt change detection in one dimensional data is also referred to quickest change detection as in [30, 5] for sensor data. In this paper, we are interested in detecting a change in the unlabeled multidimensional data streams which were studied in [21, 9, 16]. Dasu et al. [9] detected changes in multidimensional data by computing the 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. Kuncheva in [16] proposed a semi parametric log likelihood detector to measure the difference between windows for detecting changes. Qahtan et al. [21] proposed a PCA-based change detection algorithm, and shows it is superior to the methods in [9, 16]. A change is reported when the distance is much higher than the average distance using Page-Hinckley test [19].

3 Preliminary

Data Stream

DEFINITION 3.1. *[29] A data stream is a possible infinite series of data points ..., o_{n-2}, o_{n-1}, o_n, ..., where data point o_n is received at time $o_n.t$.*

In this definition, a data point o is associated with a time stamp $o.t$ at which it arrives and the stream is ordered by the arrival time. 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* setting.

DEFINITION 3.2. *[29] 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}, ... , o_n.*

With count-based window, every time the window slides, new S data points are incorporated to the window and the oldest S data points are removed from the window. S denotes the slide size which characterizes the speed of the data stream.

Non-stationary Data Streams In real-world data

streams, the changes in the underlying data distribution may come from many sources. They can be due to the nature of data. For example, in the climate data[2], the distributions of wind speed and precipitation change over seasons; in transportation data, the average speed on a highway changes over hours and days. Furthermore, distribution can change if a sensor becomes less accurate gradually over time or when another sensor with a different calibration replaces the faulty sensor.

DEFINITION 3.3. *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 one dimension and correlation between two dimensions [21, 9].

Problem Definition With the above definitions, we now formally define the problem of continuous outlier detection in non-stationary data streams (O-NSD) as follows.

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 model-based approaches 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. Each modeling technique has a different way to compute outlier scores and we will present more details about our specific choice of them in Section 5.

4 Change Detection

In this section, we present our proposed solution for change detection which is a crucial part of the outlier detection framework.

Solution Design Our proposed solution for change detection is presented in Algorithm 1. A change in mean, variance or correlation in the original space is manifested in the transformed space using the Principal Component Analysis (PCA)[1] model which is computed on the first window as discussed in [21]. Therefore, we apply PCA on the window data 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 projected data of the window which is used for model building is called the reference window. The histogram of the projected data is estimated as the counts of data values in bins whose edges are estimated as the maximum of the Sturges [26] and FD estimators [12]. Subsequently, the distance between two windows of multidimensional data is the maximum distance value across all dimensions, line 8 in Algorithm 1. The distance measure is a crucial part for detecting a change.

Algorithm 1 Change Detection

Global variables: $buffer$: The distance buffer, M: The maximum size of the buffer.

```
1: function ChangeDetection(Dt, Dr) ▷ Dt: current window, Dr: reference window
2:     [e, Dr'] = Transform(Dr)
3:     Dt' = Transform(Dt, e)
4:     dis = 0
5:     for i from 1 to d do
6:         f_i, bins = estimateHist(Dr'_i)
7:         g_i = estimateHist(Dt'_i, bins)
8:         dis = max_i(dis, IKL(g_i||f_i))
9:     if len(buffer) ≥ M then
10:        d_expired = buffer.dequeue()
11:    buffer.enqueue(dis)
12:    if IsChange(buffer) then
13:        report a change at Dt
14:        Dr = Dt
15:        buffer.clear()
```

IKL - An Improved Distance Measure Kullback-Leibler distance [15] is commonly used to measure distance between two probability distributions: $KL(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$ with $P(i), Q(i)$ is the probability of data value in bin i. $KL(P||Q) > 0$ if P and Q have different counts over bins. In other words, if D_0 and D_k are 2 sliding windows and are generated by different underlying distributions, then $KL(D_0, D_k) > 0$. However, since a histogram only approximates a distribution, the KL distance between two histograms from one distribution still can be positive. Thus, only using positive value of KL distance is not sufficient to detect a change. A threshold can be used for KL distance to detect a change. However, setting that threshold value which may require knowledge about the magnitude of change a priori is not straightforward. Therefore, in this section, we propose a new distance metric between two estimated distributions which has an important characteristic for detecting changes.

$$IKL(P||Q) = \sum_i \max(P(i) \log \frac{P(i)}{Q(i)}, Q(i) \log \frac{Q(i)}{P(i)})$$

We replace each term $P(i) \log \frac{P(i)}{Q(i)}$ in the original KL divergence formula by $\max(P(i) \log \frac{P(i)}{Q(i)}, Q(i) \log \frac{Q(i)}{P(i)})$ to maximize the distance and this new formula has the following characteristic.

THEOREM 1. *Assume that the probability distribution of data in each slide is the same for the entire window. Suppose D_0 is the last sliding window that contains all data from the first distribution, D_k and D_l are sliding windows that contain k and l slides, respectively, from a new distribution, with $0 < k < l < W/S$, we have: $IKL(D_0, D_k) < IKL(D_0, D_l)$*

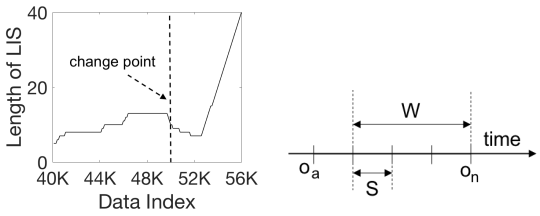
[2] https://www.pmel.noaa.gov/tao/drupal/disdel/

The proof of this theorem is presented in our full report. In other words, if the assumption on the same distribution over slide and window holds, the IKL distance from the windows overlapping the two distributions to the reference window are in an increasing order. Note that the KL distance does not have this characteristic.

Change Detection Algorithms A significant distance to the reference window can be a signal of a change in distribution. However, there are cases of signal spikes, in which the distance is large for a short time and then goes back to normal. To avoid mis-detecting these spikes as distribution changes, Dasu et al. [9] report a change after seeing p consecutive large distances. Qahtan et al. [21] report a change if the current distance value significantly deviates for a reasonable period χ from the history of the distance values. A parameter δ is also pre-defined for the allowable change in distances. However, choosing optimal values for χ, δ in [21] as well as p in [9] is difficult in practice.

When data points from a new distribution arrive, there are W/S windows overlapping the two distributions. The distances from these windows to the reference window are in an upward trend as stated in Theorem 1. We exploit this property by storing M consecutive distances ($M \leq W/S$) in a buffer. Every time the window slides, the distance from the current window to the reference window is appended to the buffer. If the buffer is full with M values, the oldest value is removed before the new value is added. The buffer is cleared after a change is detected. We propose two algorithms, i.e., AVG and Dynamic LIS, to detect a change using the buffer of the maximum size M.

AVG This algorithm is presented as Function IsChangeAVG() 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. Suppose from the beginning of the distribution, there are *count* times the buffer is full. Let $avgSc$ be the overall average distance value. When the window slides, the average value of the distances in the buffer is updated to incorporate the new distance d_{new} and remove the expired value $d_{expired}$ as: $curAvg = \frac{curAvg*M-d_{expired}+d_{new}}{M}$. The *count* and overall average value $avgScore$ is also updated to incorporate the current average value: $avgSc = \frac{avgSc*count+curAvg}{count+1}$. AVG is different from Page-Hinley test [19] which monitors single distances, used in [21] and [24]. In AVG, we monitor the average values of M consecutive distances to reduce the effect of signal spikes. The time complexity for updating the average distance is O(1), adding the new value to the buffer is O(1) and updating the overall average value is also O(1). The AVG computation has a low time complexity. Although AVG requires a pre-defined threshold, i.e., AVG_th, similarly to [21, 9], it is more intuitive for an expert. Our second approach (Section 4) does not require manual threshold setting.



(a) The length of LIS with HPC Dataset

(b) Detected change point o_n has a score of 0.9

Dynamic LIS Besides the increase in the average distance 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 subsequence. LIS has been widely studied and applied in a variety of domains, e.g., the system for aligning entire genomes [10]. Romik et al. [22] presented extensively the mathematics of LIS, e.g., the maximal length of LIS in a random permutation. Figure 2(a) shows the length of LIS of the distance sequence stored in the buffer for HPC sampling dataset (complete description in Section 6). A change happens at the $50K^{th}$ data point. As illustrated in this figure, the length of LIS is stable before the change happens and increases

Algorithm 2 AVG

Global variables: *count*: Total number of windows from the beginning of distribution, $avgSc$: Overall average scores in the buffer, M: The maximum size of the buffer.

```
1: function IsChangeAVG(buffer, AVG_th)
2:     if buffer.length < M then return FALSE
3:     curAvg = Average(buffer)
4:     avgSc = (avgSc * count + curAvg)/(count + 1)
5:     count = count + 1
6:     if curAvg > avgSc * AVG_th then
7:         return TRUE
8:     return FALSE
```

steadily afterwards. The pseudo code of Dynamic LIS is presented in Algorithm 3. Function IsChangeDLIS()

Algorithm 3 Dynamic LIS

Global variables: $possibleLIS$: Maximum possible length of LIS so far, initially set to 0.

```
1: function IsChangeDLIS(buffer, LIS_th)
2:     possibleLIS = possibleLIS + 1
3:     if possibleLIS ≤ LIS_th then
4:         return FALSE
5:     lis = GETLIS(buffer)
6:     possibleLIS = lis
7:     if lis > LIS_th then
8:         return TRUE
9:     return FALSE
```

is an instance of function IsChange() in Algorithm 1. A change is detected if the length of LIS of the distances in the buffer is higher than a threshold, LIS_th. The algorithm relies on the "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. Intuitively, we can manually set LIS_th. However, it is 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. For any n different numbers, $n \geq 1$, we denote δ_n to be a uniformly random permutation of order n, $L(\delta_n)$ to be the length of the longest increasing subsequence of δ_n, and $E(L(\delta_n))$ to be the expected value of $L(\delta_n)$.

Theorem 2. (Lower bound of LIS length) *[22] For all $n \geq 1$, we have: $E(L(\delta_n)) \geq \sqrt{n}$*

Theorem 3. (Limit of the 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 these properties, to detect a change using distances in the buffer, we set $LIS\_th = 2\sqrt{|buffer|}$. For higher $wPrecision$, LIS_th should be set higher and for higher $wRecall$, LIS_th should be set lower.

Efficient LIS Computation For a long sequence, the computation of LIS can be complex and time-consuming. Therefore, with a buffer size M, the time complexity to get LIS is $O(MlogM)$. To reduce the time for computing LIS for every window, we use a variable $possibleLIS$ to track the maximal possible length of LIS. The actual length of LIS 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 detected. If $possibleLIS$ is higher than LIS_th, the actual LIS is computed. If the actual LIS is longer than LIS_th, then a change is detected, otherwise, $possibleLIS$ is updated by the length of the actual LIS.

5 Outlier Detection Framework

Framework Design We first present our framework design for outlier detection in data streams utilizing change detection to trigger the model update. The pseudo code of our framework are presented in Algorithm 4. A model is first built from the initial window data, Function Train(), and it is used to detect outliers (function OutlierDetection()) for future windows if there is not a change in the distribution. This reduces the number of model updates and re-evaluations for existing data points. When the window slides, if a change is detected by change detection (Function ChangeDetection()) then the model is rebuilt with the data of the next window starting from the change point. This ensures that the model is built from only the new distribution. In addition, the reference window for change detection is also updated.

Outlier Detection Algorithm. This framework is applicable to various model-based outlier detection algorithms. In this study, to demonstrate the framework, we use PCA and One-class SVM technique considering their prevalence in practice. In general, after the outlier scores of data points are computed, some criteria can be used to report outliers. 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.

Algorithm 4 Outlier Detection Framework

Input: Stream x, Window Size W, Slide Size S
Output: Outliers in every sliding window
Procedure:

```
1: Dr = {x_1, x_2, ..., x_W}
2: model = Train(Dr)
3: yield OutlierDetection(model, Dr)
4: Dt = Dr                                  ▷ Initialize sliding window Dt
5: while a new slide S arrives do
6:     Dt = Dt \ S_expired                  ▷ Remove expired slide
7:     Dt = Dt ∪ S                          ▷ Incorporate new slide
8:     if ChangeDetection(Dt, Dr) then
9:         model = Train(D_{t+W-S})
10:        yield OutlierDetection(model, D_{t+W-S})
11:    else
12:        yield OutlierDetection(model, S)
```

5.1 Evaluation Metric

Change Detection Since the outlier detection model needs to be updated as a change occurs, prompt detection of changes is crucial. For the O-NSD problem, the detected change point should be close to the actual change point. Therefore, there is a need for a metric to quantify how timely the changes are detected. The metric should satisfy the following conditions: 1) the score of a detected change is higher or equal to zero, and 2) the score of a detected change decreases as its distance to the actual change point increases. In [21, 9], the authors consider a detection late if the last data point of the detected window is two windows away from to the actual change point. This approach uses a hard cutoff, i.e., 2, and cannot quantify the timeliness of detection smoothly. In this study, we propose three *weighted* measures, i.e., $wPrecision, wRecall$ and $wF1$ to quantify the timeliness of change detection. In order to define the measures, we define the score of a detected window. The detected change point is considered as the last data point of the detected window. To define a score of detected change points, we align them with the actual change points. If there are more than one detected change points which are matched with the same actual change point, the closest change point has a positive score, and the others are considered false positive detection with the score of 0. Assume D_n is detected as a change occurs, and the actual change point is o_a. The score of D_n depends on the number of disjoint windows passed after o_a: $score(D_n) = e^{-\lambda \left\lfloor \frac{n-a}{W} \right\rfloor}$ where λ is a decay factor, which can be a small positive number, $0 < \lambda < 1$. 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 when the distance from the detected windows to the actual change point increases. The parameter λ can be set by the users to control how fast the score decays with time. Now, we are ready to present the metrics to evaluate the change detection results. Assume there are A actual change points and K detected windows with scores $score_1$, $score_2$, ..., $score_K$. $wPrecision$, $wRecall$, $wF1$ are defined as follows: $wPrecision = \frac{\sum_{i=1}^{K} score_i}{K}$, $wRecall = \frac{\sum_{i=1}^{K} score_i}{A}$. $wPrecision$ and $wRecall$ measures the correctness and sufficiency of detected changes, respectively, $wF1$ is their harmonic mean.

Outlier Detection To evaluate the performance of the proposed framework, we compared the results 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.

Time and Space Analysis The time complexity of our framework depends on the following three main procedures. For each window W of d-dimensional data: 1) outlier detection model building costs $O(d^2W + d^3)$ and $O(dW^3)$ in PCA and One-class SVM using a linear kernel, respectively, and 2) outlier score computation and sorting cost $O(Wd + WlogW)$, $O(dW)$ for a sliding window in PCA and One-class SVM, respectively, and 3) change detection time includes applying PCA for the reference window that costs $O(Wd^2)$, incremental update of estimated histogram which costs constant time, and LIS computation that costs $O(MlogM) = O(W/Slog(W/S)$. In summary, suppose that there is one change after N disjoint windows, $N > 1$, the average total time complexity of PCA and One-class SVM for one window is $O(dW(logW + \frac{d}{N}))$ and $O(dW(1 + \frac{W^2}{N}))$, respectively. This shows that the change detection module does not incur much overhead.

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 $O(Wd)$, and 2) the reference window that costs $O(Wd)$, and 3) the distances in the buffer that cost $O(M)$, and 4) the estimated histograms that cost $O(Wd)$. In total, the framework requires $O(Wd)$ space.

6 Experiments

6.1 Experimental Methodology Our change detection methods, i.e., Dynamic LIS and AVG are compared with the state of the art PHDT [21]. The distance between windows are rounded to three digits after the decimal point. 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 one-time approach which does not update the model. The algorithms are implemented in Python and our source codes are available online[3]. Experiments are conducted on a Linux machine with 4 cores 2.7GHz, 24GB memory.

Synthetic Datasets We generate synthetic datasets with changes in one parameter with 2 dimensions for change detection and 5 dimensions for outlier detection experiments. Our synthetic datasets are generated similarly as in [21, 9]. 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 2 randomly selected dimensions. The length of distributions l is 50000 in the change detection experiments and a

[3] https://github.com/tranvanluan2/ONSD

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[4] 2) **Forest Cover** (FC) has 55 attributes, we select 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[5]. Because the ground-truth of distribution changes is not available, we simulate artificial changes as in [25]. 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 mean and variance is equal to 10% of mean and variance of the batch, respectively 2) **Scale-1D**: each value of one randomly selected dimension is doubled. Specifically, we have FC-G, Ethyl-G, HPC-G, TAO-G are the real-world datasets applied Gaussian-1D change and FC-S, Ethyl-S, HPC-S, TAO-S are the real-world datasets applied Scale-1D change.

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 changing mean, 0.2 for standard deviation and 0.1 for changing correlation. The default values of χ and δ for PHDT are set to 500 and 5.10^{-3}, respectively as in [21]. The buffer size M is equal to W/S. $AVG\_th = 1.3$. The default outlier rate is 5%.

6.2 Change Detection

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 0.05 to 0.2 for correlation coefficient. Table 1 shows the $wF1$ of PHDT, AVG and Dynamic LIS. As can be seen in this table, when ϵ increases, the $wF1$ of all the algorithms increase because of more difference between distributions. Dynamic LIS and AVG offer the highest $wF1$ in most cases. Especially, Dynamic LIS performs better than AVG and PHDT with small ϵ because it does not rely on the absolute increase in distances.

Slide Size Sensitivity We vary the slide size from 1 to 400. When the slide size increases, the difference between distributions 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 2 shows the $wF1$ of PHDT, AVG and Dynamic LIS. As can be seen in this table, when the slide size increases from 1 to 20, AVG and Dynamic LIS offer higher performances than PHDT in all cases. AVG offers the most stable $wF1$s because its detection is less affected by the reduction of the buffer size.

Change	ϵ	0.01	0.02	0.03	0.04	0.05
Mean	PHDT	0.38	0.58	0.67	0.87	0.9
	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**
	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
	AVG	0.28	0.84	**0.97**	**1.0**	**1.0**
	DLIS	**0.35**	**0.86**	0.89	**1.0**	**1.0**

Table 1: Varying Change Magnitude - $wF1$

Change	Slide Size	1	20	100	200	400
Mean	PHDT	0.67	0.68	0.66	0.64	0.69
	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
	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
	AVG	**0.98**	**0.97**	**0.97**	**0.98**	**0.96**
	DLIS	0.89	0.91	0.88	0.84	0.86

Table 2: Varying Slide Size - $wF1$

Change	Dataset	TAO	Ethylen	HPC	FC
Scale 1D	PHDT	0.88	0.88	0.88	**0.91**
	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
	AVG	**0.78**	0.96	0.73	0.80
	DLIS	**0.78**	**0.97**	**0.89**	**0.89**

Table 3: Change Detection with Real-world Datasets - $wF1$

Change Detection with Real-world Datasets Table 3 shows the $wF1$ of PHDT, AVG and Dynamic LIS with the 4 real-world datasets and 2 types of change, i.e., Gaussian 1D and Scale 1D. As can be seen in this table, Dynamic LIS and AVG offer the highest $wF1$ in most cases.

6.3 Outlier Detection Since Dynamic LIS detects distribution changes most accurately, we incorporate it into the outlier detection framework. We demonstrate the efficiency of our proposed framework by using PCA-

[4] https://www.pmel.noaa.gov/

[5] https://archive.ics.uci.edu/ml/datasets.html

based and One-class SVM Outlier Detection. Our approach including CD-SVM, CD-PCA is compared with one-time based approach including OT-SVM, OT-PCA which does not update the model at all and re-train based approaches including Retrain SVM, Retrain PCA which update the model periodically after every t data points. For PCA, we also compare with Incremental PCA [23] which updates the PCA model in every slide. Incremental approach for SVM is not available.

F1-score Table 4, 5 show the F1-score of One-class SVM and PCA based approaches with the real-world and synthetic datasets. As can be seen from these tables, the change detection approach offers the highest F1-scores. One-time approach with One-class SVM and PCA gives the lowest F1-score because the model is outdated when the distribution changes. Incremental PCA incrementally updates the model approximately, however, it does not adapt quickly to the change of distribution. Because retrain-based approach updates the model without the change information, the data used for model building can from two distributions and it causes wrong detection.

Dataset	CD-SVM	OT-SVM	Retrain SVM		
			5k	10k	20k
Mean	**0.92**	0.01	0.87	0.85	0.79
STD	**0.96**	0.69	0.95	0.94	0.92
Corr	**0.97**	0.83	0.96	0.95	0.93
FC-G	**0.93**	0.73	0.93	0.92	0.89
Ethyl-G	**0.94**	0.52	0.93	0.93	0.89
HPC-G	**0.94**	0.62	0.93	0.92	0.9
TAO-G	**0.85**	0.42	0.83	0.81	0.74
FC-S	**0.91**	0.48	0.90	0.88	0.85
Ethyl-S	**0.96**	0.49	0.93	0.92	0.87
HPC-S	**0.89**	0.78	0.90	0.89	0.87
TAO-S	**0.87**	0.45	0.84	0.82	0.76

Table 4: One-class SVM Outlier Detection - F1-score

Dataset	CD-PCA	OT-PCA	Retrain PCA			IPCA
			5k	10k	20k	
Mean	**0.94**	0.87	0.92	0.91	0.90	0.89
STD	**0.91**	0.86	0.89	0.88	0.87	0.82
Corr	**0.90**	0.64	0.88	0.87	0.85	0.7
FC-G	**0.91**	0.48	0.89	0.87	0.82	0.72
Ethyl-G	**0.94**	0.46	0.92	0.89	0.83	0.72
HPC-G	**0.95**	0.55	0.92	0.90	0.86	0.78
TAO-G	**0.92**	0.62	0.89	0.87	0.84	0.79
FC-S	**0.92**	0.43	0.88	0.86	0.85	0.84
Ethyl-S	**0.90**	0.19	0.87	0.84	0.75	0.62
HPC-S	**0.91**	0.70	0.89	0.88	0.86	0.82
TAO-S	**0.91**	0.78	0.88	0.86	0.85	0.88

Table 5: PCA Outlier Detection - F1-score

Running Time Table 6 shows the average running time for each sliding window in milliseconds of the approaches with the synthetic and real-word datasets. The result with One-class SVM is available in our technical report. As can be seen in these tables, for 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. CD-PCA incur the smallest running time. Incremental PCA updates the model in every slide and then the entire window is re-evaluated, thus requires the highest running times.

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

Table 6: PCA Outlier Detection - Running Time (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 use a buffer for storing the distance from the current window to the reference window, with two change detection algorithms, in which AVG exploits the increase in the average distance, and Dynamic LIS exploits the increase in the length of the longest increasing subsequence in the buffer. We proposed a new distance measure IKL, to ensure the increasing property of the distance values when the current window slides into the new distribution. We also proposed new evaluation metrics to quantify the timeliness of the detected changes. Our time and space analysis showed that the incorporated change detection part does not incur much overhead. Our experiment results showed that Dynamic LIS offers the highest $wF1$ for change detection, in comparison to AVG and a state-of-the-art algorithm PHDT [21]. Our proposed framework provides highly accurate outlier detection, requires significantly less running time than the incremental and retrain based approaches.

References

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

[2] R. Arora, A. Cotter, K. Livescu, and N. Srebro. Stochastic optimization for pca and pls. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 861–868, Oct 2012.

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

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

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

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

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

[8] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, pages 707–720, London, UK, UK, 2002. Springer-Verlag.

[9] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. 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*, 2006.

[10] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27(11):2369–2376, 1999.

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

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

[13] R. Fujimaki, T. Yairi, and K. Machida. 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*, pages 401–410. ACM, 2005.

[14] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[15] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[16] L. I. Kuncheva. Change detection in streaming multivariate data using likelihood detectors. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1175–1180, May 2013.

[17] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1741–1745. IEEE, 2003.

[18] P. M. Mafra, V. Moll, J. da Silva Fraga, and A. O. Santin. Octopus-iids: An anomaly based intelligent intrusion detection system. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 405–410. IEEE, 2010.

[19] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi. 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)*, volume 2, pages 815–818 Vol.2, July 2004.

[20] H. V. Neto and U. Nehmzow. Incremental pca: An alternative approach for novelty detection. *Towards Autonomous Robotic Systems*, 2005.

[21] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang. 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, pages 935–944, New York, NY, USA, 2015. ACM.

[22] D. Romik. *The surprising mathematics of longest increasing subsequences*, volume 4. Cambridge University Press, 2015.

[23] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141, 2008.

[24] R. Sebastiao and J. Gama. A study on change detection methods. In *Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA*, pages 12–15, 2009.

[25] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 667–676, New York, NY, USA, 2007. ACM.

[26] H. A. Sturges. The choice of a class interval. *Journal of the american statistical association*, 21(153):65–66, 1926.

[27] J.-i. Takeuchi and K. Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*, 18(4):482–492, 2006.

[28] M. Thottan and C. Ji. Anomaly detection in ip networks. *IEEE Transactions on signal processing*, 51(8):2191–2204, 2003.

[29] L. Tran, L. Fan, and C. Shahabi. Distance-based outlier detection in data streams. *Proc. VLDB Endow.*, 9(12):1089–1100, Aug. 2016.

[30] V. V. Veeravalli and T. Banerjee. Quickest change detection. In *Academic Press Library in Signal Processing*, volume 3, pages 209–255. Elsevier, 2014.

[31] Y. Zhang, N. Meratnia, and P. J. M. Havinga. Ensuring high sensor data quality through use of online outlier detection techniques. *Int. J. Sen. Netw.*, 7(3):141–151,

May 2010.

8 Appendix

Proof of Theorem 1. Assume the histogram of the current distribution contains n bins $B_0 = \{b_1^{(0)}, b_2^{(0)}, ..., b_n^{(0)}\}$ with the probability that a data point belongs to bin b_i is $x_i^{(0)}$, $x_1^{(0)} + x_2^{(0)} + ... + x_n^{(0)} = 1$. Assume the probability distribution of the new distribution over the bins B_0 is $\{\gamma_1, \gamma_2, ..., \gamma_n\}$, $\gamma_1+\gamma_2+...+\gamma_n \leq 1$. Here, we assume that the slide size is large enough so that the data points in one slide are distributed over the bins similarly to the data points in the entire window. When the window receives new k slides from a new distribution and it removes k expired slides, let the probability that one data point belongs to bin b_i be $x_i^{(k)}$. We will compute $x_i^{(k)}$ from $x_i^{(0)}$, γ_i, the window size W, and the slide size S. With k expired slides, the probability of a data point to belong to bin b_i decreases $kS/Wx_i^{(0)}$, and with new k slides from the new distribution, the probability of a data point to belong to bin i gains $k\gamma_i S/W$. Therefore,

$$x_i^{(k)} = x_i^{(0)} - k\frac{S}{W}x_i^{(0)} + k\frac{S}{W}\gamma_i \tag{8.1}$$

Let $\beta_i = \frac{S}{W}x_i^{(0)} - \frac{S}{W}\gamma_i$, we have $x_i^{(k)} = x_i^{(0)} - k\beta_i$. The IKL distance between D_0 and D_k is:

$$IKL(D_0, D_k) = \sum_{i=1}^{n} \max(x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(k)}}, x_i^{(k)} \log \frac{x_i^{(k)}}{x_i^{(0)}})$$
$$= \sum_{i=1}^{n} \max(x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - k\beta_i}, x_i^{(k)} \log \frac{x_i^{(0)} - k\beta_i}{x_i^{(0)}})$$

Similarly, we have

$$IKL(D_0, D_l) = \sum_{i=1}^{n} \max(x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - l\beta_i}, x_i^{(l)} \log \frac{x_i^{(0)} - l\beta_i}{x_i^{(0)}})$$

It is easy to see that:

$$\max(x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - k\beta_i}, x_i^{(k)} \log \frac{x_i^{(0)} - k\beta_i}{x_i^{(0)}}) = x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - k\beta_i}$$

when $\beta_i \geq 0$, and

$$\max(x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - k\beta_i}, x_i^{(k)} \log \frac{x_i^{(0)} - k\beta_i}{x_i^{(0)}}) = x_i^{(k)} \log \frac{x_i^{(0)} - k\beta_i}{x_i^{(0)}})$$

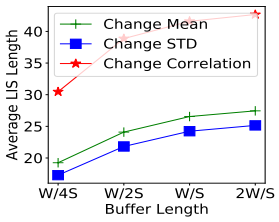
when $\beta_i < 0$. Therefore, the difference between these two IKL distances is:

$$diff = IKL(D_0, D_l) - IKL(D_0, D_k)$$
$$= \sum_{\beta_i \geq 0} (x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - l\beta_i} - x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - k\beta_i})$$
$$+ \sum_{\beta_i < 0} (x_i^{(l)} \log \frac{x_i^{(0)} - l\beta_i}{x_i^{(0)}} - x_i^{(k)} \log \frac{x_i^{(0)} - k\beta_i}{x_i^{(0)}})$$
$$= \sum_{\beta_i \geq 0} x_i^{(0)} \log(1 + \frac{(l-k)\beta_i}{x_i^{(0)} - l\beta_i})$$
$$+ \sum_{\beta_i < 0} (x_i^{(k)} \log(1 - \frac{(l-k)\beta_i}{x_i^{(0)} - k\beta_i})$$
$$+ (x_i^{(l)} - x_i^{(k)}) \log \frac{x_i^{(0)} - l\beta_i}{x_i^{(0)}})$$

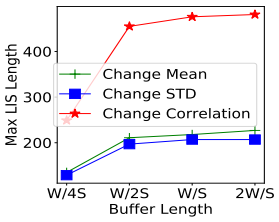
Since $k < l < W/S$, we have $l - k > 0$ and $x_i^{(0)} - k\beta_i > 0$, $x_i^{(0)} - l\beta_i > 0$, $x_i^{(l)} - x_i^{(k)} > 0$ when $\beta_i < 0$ $\Rightarrow diff > 0$. This completes the proof of the theorem.

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 more distance values in the buffer, the number of values which are in an increasing order can be larger. Figure 2 shows the average and maximal LIS length in the buffer when varying the buffer length from $W/4S$ to $2W/S$ with mean, standard deviation, and correlation changing data. 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 $M = W/S$, we can get high average LIS length and maximal LIS length, comparable to $M = 2W/S$. 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 W/S such windows for each change. It confirms our choice of buffer length which is W/S.

Parameter Studying. In many applications, sooner change detection or accurate change detection can be preferable. We vary AVG_th and LIS_th to examine the trade-off between the $wPrecision$ and $wRecall$ of the detected changes. AVG_th is varied from 1.1 to 2. When AVG_th increases, the criteria for detecting a change is more strict. Table 7 shows the $wPrecision - wRecall - wF1$ of AVG when varying AVG_th with the synthetic datasets. As we can see in this table, when AVG_th increases, the $wPrecision$ increases as the condition is more strict, and the $wRecall$ decreases

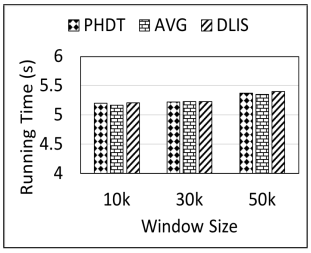


(c) Average LIS Length

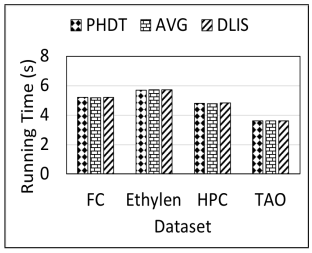


(d) Maximum LIS Length

Figure 2: Varying Buffer Length



(a) Varying Window Size



(b) Varying Dataset

Figure 3: Change Detection Running Time

as fewer changes are detected. With $AVG\_th = 1.5$, AVG offers the highest $wF1$ in most cases.

AVG_th	Mean	STD	Correlation
1.1	0.61-0.93-0.74	0.72-0.77-0.75	0.66-0.90-0.76
1.3	**0.80-0.70-0.75**	**0.90-0.75-0.82**	0.90-0.70-0.79
1.5	**0.90-0.63-0.75**	**0.97-0.70-0.82**	0.94-0.68-0.79
1.7	0.91-0.60-0.72	0.96-0.53-0.68	**0.97-0.62-0.86**
2	0.91-0.55-0.69	0.96-0.25-0.40	0.97-0.60-0.74

Table 7: Varying AVG_th with Synthetic Datasets

In Dynamic LIS, the changes can be detected sooner with a smaller LIS_th and the precision of the detection is higher when the LIS_th is increased. We set $LIS\_th = \Theta\sqrt{W/S}$ and vary the parameter Θ from 1 to 2. The results with the synthetic datasets are shown in Table 8. As can be seen in this table, $wRecall$ decreases when LIS_th increases, meanwhile, $wPrecision$ increases. When $LIS\_th = 2\sqrt{W/S}$, Dynamic LIS offers the highest average $wF1$ and that confirms our choice of LIS_th.

Θ	Mean	STD	Correlation
1	0.45-1.00-0.62	0.51-0.99-0.67	0.51-1.00-0.67
1.3	0.58-1.00-0.73	0.69-0.94-0.79	0.71-0.89-0.79
1.5	0.63-1.00-0.77	0.82-0.89-0.86	0.78-0.89-0.83
1.7	0.79-1.00-0.88	0.93-0.85-0.89	0.87-0.87-0.87
2	**0.89-1.00-0.94**	**1.00-0.84-0.91**	**0.94-0.83-0.88**

Table 8: Varying LIS_th with Synthetic Datasets

Scalability Comparison. Figure 3 shows the running time of the algorithms with different window sizes with FC dataset and different dimensions by using different datasets. 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 dimension. In both cases, all three algorithms have comparable running time, showing our DLIS does not incur much overhead.

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. Due to 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:

$$score(x) = \sum_{i=1}^{d} \frac{|(x-\mu)e_i|^2}{\lambda_i} \tag{8.2}$$

where d is the number of dimensions, e_i and λ_i are the i^{th} eigenvector and the i^{th} largest eigenvalue, respectively. **One-class SVM Outlier Detection.** One-class SVM [1] 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. The algorithm can be summarized as mapping the data into a feature space H using an appropriate kernel function Φ, and then trying to separate the mapped vectors from the origin with maximum margin. $f(x) = 1$ if x is a normal data and $f(x) = -1$ if x is an outlier. In our context, suppose $x_1, x_2, ..., x_n$ are training examples, let $\Phi : X \to H$ be a kernel map which transforms the training examples to another space, then to separate the data set from the origin, one needs to solve the following quadratic programming problem:

$$min \frac{1}{2}||w^2|| + \frac{1}{vn}\sum_{i=1}^{n}\chi_i - p \tag{8.3}$$

subject to

$$w.\Phi(x_i) \geq p - \chi_i \tag{8.4}$$

$i = 1, 2, ..., n$, p is a bias term, χ_i is a slack variable for point i, $\chi_i \geq 0$, that allows it to lie on the other side of the decision boundary. v is the regularization parameter. If w, p solves this problem, the decision function $f(x) = sign(w.\Phi(x) - p)$ will be positive for most of x in the training data.

r	CD-PCA	OT-PCA	RetrainPCA					IPCA
			100	10k	20k	50k	80k	
0.01	0.86	0.41	**0.89**	0.81	0.74	0.83	0.49	0.77
0.05	0.91	0.43	**0.93**	0.85	0.78	0.87	0.55	0.84
0.1	0.92	0.55	**0.94**	0.88	0.82	0.89	0.65	0.87

Table 9: PCA-based Outlier Detection with FC-G Dataset - F1-score - Varying Outlier Rate

Stability with Outlier Rate. We compare the approaches when varying the outlier rate r from 0.01 to 0.1. Table 10, 9 show the F1-score of the all approaches with the FC dataset with Gaussian-1D change injection. As can be seen in these tables, the F1-scores of all methods increase when the outlier rate increases because there is more chance a local outlier, i.e., the outlier in a sliding window can be a global outlier, i.e., an outlier in the entire distribution. For One-class SVM, the change detection approach has a comparable result with Retrain, $t = 100$ and $t = 50k$ in most cases. For PCA, the change detection approach offers higher F1-score than Retrain, $t = 50k$ and close F1-score to Retrain, $t = 100$.

r	CD-SVM	OT-SVM	Retrain SVM				
			100	10k	20k	50k	80k
0.01	**0.9**	0.5	**0.9**	0.88	0.85	0.88	0.66
0.05	**0.91**	0.67	**0.91**	**0.91**	0.88	**0.91**	0.73
0.1	**0.93**	0.69	**0.93**	0.92	0.88	**0.93**	0.72

Table 10: One-class SVM Outlier Detection with FC-G Dataset - F1-score - Varying Outlier Rate

Dataset	CD-OCSVM	Retrain OCSVM		
		5k	10k	20k
Mean	**49**	130	77	50
STD	**47**	123	77	51
Corr	**47**	120	74	53
FC-G	**63**	167	97	75
Ethyl-G	**68**	173	106	70
HPC-G	64	140	81	52
TAO-G	**41**	109	64	44
FC-S	**56**	121	65	60
Ethyl-S	**58**	162	92	60
HPC-S	**46**	135	80	51
TAO-S	**40**	110	64	42

Table 11: OCSVM Outlier Detection - Running Time(s)