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 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 change detection solution. In addition, our O-NSD framework (demonstrated with two popular unsupervised outlier classifiers) offers higher accuracy for outlier detection and requires a much lower running time than retrain-based and incremental update approaches.

Keywords: outlier detection, non-stationary, data streams

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 change. 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. A baseline approach to tackle

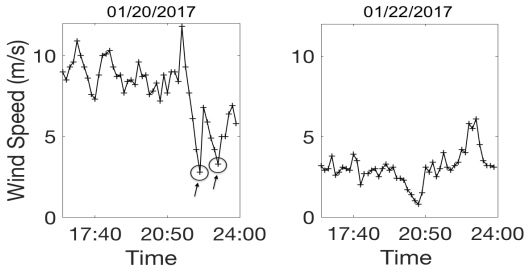


Figure 1: Wind speed data on two days

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 the timing of changes is usually unpredictable in practice. In this paper, we propose the *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

*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/

increase. Therefore, we can monitor the sequence of the aforementioned "difference" 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 [17] 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 metric IKL to measure the distance between two distributions. We prove that the distance between the current window and the reference window monotonically increases at the beginning of the a 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 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.

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 our problem definition. In Section 4 and 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. Principal Component Analysis (PCA) [1] and One-class SVM [31, 1] 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 [28] and in space craft components [12]. Incremental PCA has been used in visual novelty detection mechanism [19], outlier detection in energy data streams [10], and spatial-temporal data in [5]. 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., outlier detection in wireless sensor network [31] and time-series data [16]. 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 [6, 13, 30]. In this paper, we are interested in detecting a change in the unlabeled multidimensional data streams which were studied in [21, 8, 15]. Dasu et al. [8] 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 [15] 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 using Page-Hinckley test [18] which is shown to be superior to the methods in [8, 15]. 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 our problem definition.

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. 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 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.*

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; 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.

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 individual dimensions and in correlation between dimensions as in [21, 8]. 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 model-based approaches [10, 20, 22] 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 [21], 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. 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 (Function EstimatedHist in line 5) using histograms in each dimension whose edges are estimated as the maximum of the Sturges [27] and FD estimators [11]. Subsequently, the distance between two windows is the maximum distance across all dimensions, as 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 [14] is commonly used to measure distance between two probability distributions: $KL(P||Q) = \sum_j P(j) \log \frac{P(j)}{Q(j)}$ with $P(j)$ and $Q(j)$ are the probability of data value in bin j. $KL(P||Q)$ is positive if P and Q have different counts over bins. In other words, if D_0 and D_j are 2 sliding windows and are generated by different underlying distributions, we have $KL(D_0, D_j) > 0$. 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.

(4.1)
$$IKL(P||Q) = \sum_j \max(P(j) \log \frac{P(j)}{Q(j)}, Q(j) \log \frac{Q(j)}{P(j)})$$

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 and this new formula has the following characteristic.

THEOREM 1. *Assume the probability distribution of each slide is the same. Suppose D_0 is the last sliding window from the previous distribution, D_l and $D_{l'}$ are sliding windows that contain l and l' slides, respectively, from a new distribution, with $0 < l < l' < W/S$, we have: $IKL(D_0, D_l) < IKL(D_0, D_{l'})$*

The proof of this theorem is detailed in our appendix. 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.

4.2 Change Detection Algorithms

A significant distance to the reference window can signify a change in 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 as distribution changes, Dasu et al. [8] report a change after seeing p consecutive large distances. Qahtan et al. [21] 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 [21] and p in [8] 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 from the current window to the reference window is computed and appended to the buffer. 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 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.

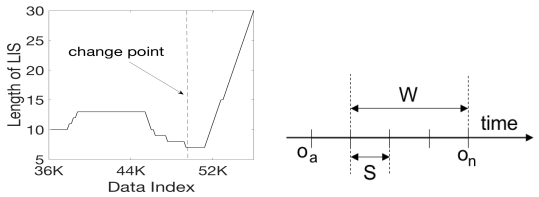
AVG. This algorithm is presented in Algorithm 2,

Algorithm 2 AVG

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

1: **function** IsChangeAVG($buffer, AVG\_th$)
2: $curAvg = Average(buffer)$
3: $allAvg = (allAvg * count + curAvg)/(count + 1)$
4: $count = count + 1$
5: **if** $curAvg > allAvg * AVG\_th$ **then**
6: $count = 0$
7: **return** TRUE
8: **return** FALSE

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. The overall average value *allAvg* is also updated accordingly. Our proposed solution is different from Page-Hinckley test [18] which monitors single distances, used in [21] and [25]. In AVG, we monitor the average value of M consecutive distances



(a) The LIS length with HPC Dataset (b) Detected change point o_n has a score of 0.9

Figure 2: LIS length and an example of the proposed evaluation metric

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 [21, 8], it is more intuitive for the user. Our second approach does not require any manual threshold setting.

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 subsequences. LIS has been widely studied and applied in a variety of domains, e.g., the system for aligning entire genomes [9]. Romik et al. [23] presented extensively the mathematics of LIS, e.g., the maximal LIS length in a random permutation. Figure 2(a) shows the LIS length of the distance sequence stored in the buffer using HPC dataset (complete description in Section 6). A change happens at the $50K^{th}$ data point. As illustrated in this figure, the LIS length is stable before the change happens and increases steadily afterwards. The pseudo code of Dynamic LIS is presented in Algorithm 3, which is an instance of function IsChange() in Algorithm 1. A change is detected if the LIS length 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

Algorithm 3 Dynamic LIS

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

```
1: function IsChangedLIS(buffer, LIS_th)
2:     possibleLIS = possibleLIS + 1
3:     if possibleLIS ≤ LIS_th then return FALSE
4:     possibleLIS, lis = GETLIS(buffer)
5:     if lis > LIS_th then
6:         possibleLIS = 0; return TRUE
7:     return FALSE
```

when the window slides. 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. 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) [23] *For all $n \geq 1$, we have: $E(L(\delta_n)) \geq \sqrt{n}$*

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 $LIS\_th = 2\sqrt{|buffer|}$ accordingly.

Efficient LIS Computation. For a long sequence, the computation of LIS can be complex and time-consuming. 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 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.

5 Outlier Detection Solution

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 a change is not detected. This reduces the number of model updates and re-evaluations for existing data points. When the window slides, if a change is detected (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 to reflect only the new distribution. The reference window for change detection is also updated.

Outlier Detection Algorithms. This framework is applicable to various model-based outlier detection algorithms. In this study, to demonstrate the framework, we use PCA [1, 28] and One-class SVM technique [1, 20, 22] considering their prevalence in practice. In PCA, the outlier score of a data point is measured by the normalized distance to the centroid along the principal components. That distance is weighted based on the eigenvalues. In One-class SVM, the outlier score of a data point is the distance from it to the learned boundary of normal data. 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 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: yield OUTLIERDETECTION(model, D_r)
3: for every sliding window D_t do
4:     if CHANGEDETECTION(D_t, D_r) then
5:         model = TRAIN(D_{t+W-S})   ▷ Build model with data of new distribution
6:         yield OUTLIERDETECTION(model, D_{t+W-S})
7:     else
8:         yield OUTLIERDETECTION(model, {o_{t-S+1}, ..., o_t})
```

Timeliness Evaluation for 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 new metric to quantify how timely the changes are detected. The metric should satisfy the following properties: 1) the score of a detected change is higher than or equal to zero, and 2) the score of a detected change decreases as its distance to the actual change point increases. In [8, 21], 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 cutoff, i.e., 2 windows length, 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 smoothly over time. 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 the score of a detected change point, we align it with the actual change point. 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 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: $score(D_n) = e^{-\lambda \lfloor \frac{n-a}{W} \rfloor}$ where λ is a decay factor, which can be a small positive number, $0 < \lambda < 1$. 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 when the distance from the detected windows to the actual change point increases. For example, with $\lambda = 0.1$, $D_n = o_a + 4/3W$ as in Figure 2(b), D_n has score 0.9. 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: $wPrecision = \frac{\sum_{i=1}^{K} score_i}{K}$, $wRecall = \frac{\sum_{i=1}^{K} score_i}{A}$. $wPrecision$ measures the correctness, $wRecall$ measures the sufficiency of detected changes, and $wF1$ is their harmonic mean.

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 $O(dW + WlogW)$, $O(dW)$ 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 $O(d^2W)$, incremental update of estimated histogram which costs constant time, and LIS computation that costs $O(MlogM) = O(\frac{W}{S}log(\frac{W}{S}))$ as we set $M = W/S$ in our evaluation. In summary, suppose that there is one change after N 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 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 $O(dW)$, and 2) the reference window that costs $O(dW)$, and 3) the distances in the buffer that cost $O(M)$, and 4) the estimated histograms that cost $O(dW)$. In total, the framework requires $O(dW)$ 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 change detection method PHDT [21]. The outlier detection framework is compared with the incremental approach [2, 7] 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 and our source codes are available online[2]. 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 [8, 21]. 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 with ϵ is a parameter controlling the magnitude of change and added to the distribution's parameters. The length of distributions l is 50000 in the change detection experiments and 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[3], 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[4]. Because the ground-truth of distribution changes is not available, we simulate artificial changes as in [26]. 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 a dataset name to indicate added Gauss-1D changes and "-S" for Scale-1D changes.

Default Parameters. The window size W and slide

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

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

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

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 - $wF1$

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}, respectively as in [21]. The buffer size M is set to W/S and AVG_th is set to 1.3. The default outlier rate is 5%. The decay factor λ in the evaluation metric is set to 0.1.

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 $wF1$s of PHDT, AVG and Dynamic LIS. As can be seen in this table, when ϵ increases, the $wF1$s of all the algorithms increase because of more difference between distributions. Dynamic LIS and AVG offer the highest $wF1$s 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.

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 - $wF1$

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 $wF1$s of PHDT, 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 higher performance 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 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., 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.

6.3 Outlier Detection

Since Dynamic LIS shows robust performance across datasets and does not require any threshold setting, we adopt this method in the outlier detection framework. We demonstrate the efficiency of our proposed framework by using PCA-based and One-class SVM classifers. Our approach including DLIS SVM, DLIS PCA is compared with static approach including Static SVM, Static PCA which does not update the model at all and re-train based approach including Retrain SVM, Retrain PCA which updates the model periodically after every t data points. For PCA, we also compare with Incremental PCA [24] 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.

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. Static One-class SVM and PCA offer 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. In retrain-based approach, the data used for model building can be from two distributions and it results wrong detection.

Running Time. Table 6 and 7 show the average running time for each sliding window in milliseconds of the approaches with the synthetic and real-word datasets.

Dataset	DLIS SVM	Static 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	0.90
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	DLIS PCA	Static PCA	Retrain PCA			Incr. PCA
			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	0.70
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

As can be seen in these tables, in 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. DLIS PCA and DLIS SVM incur the smallest running time. Incremental PCA updates the model in every slide and the entire window is re-evaluated, thus requires the highest running time.

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 IKL measures, 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 increase in the length of the longest increasing subsequence in the buffer which is parameter-free. Our proposed IKL measure ensures monotonicity of the distance values as 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 does not incur much overhead in the outlier detection framework. The experiment results showed that Dynamic LIS offers robust performance for change detection without the need of a user specified threshold. 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.

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

Table 6: One-class SVM Outlier Detection - Running Time(ms)

Dataset	DLIS PCA	Retrain PCA			Incr. PCA
		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 7: PCA Outlier Detection - Running Time (ms)

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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] R. Perdisci, G. Gu, and W. Lee. 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*, pages 488–498. IEEE, 2006.

[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] S. Rajasegarar, C. Leckie, J. C. Bezdek, and M. Palaniswami. Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks. *IEEE Transactions on Information Forensics and Security*, 5(3):518–533, Sept 2010.

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

[24] 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.

[25] 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.

[26] 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.

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

[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

8.1 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 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 $x_i^{(0)}$, 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 $lS/W x_i^{(0)}$, 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,

$$x_i^{(l)} = x_i^{(0)} - l\frac{S}{W}x_i^{(0)} + l\frac{S}{W}\gamma_i \tag{8.2}$$

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

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

Similarly, we have

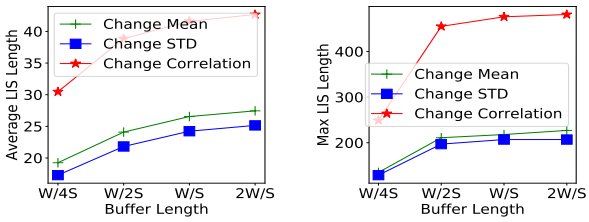
$$IKL(D_0, D_{l'}) = \sum_{i=1}^{n} \max(x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - l'\beta_i}, x_i^{(l')} \log \frac{x_i^{(0)} - l'\beta_i}{x_i^{(0)}})$$

It is easy to see that:

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

when $\beta_i \geq 0$, and

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



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

Figure 3: Varying Buffer Length

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

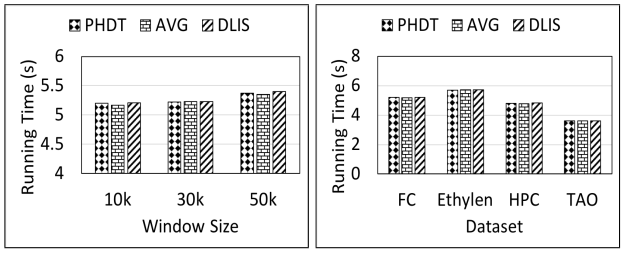
$$diff = IKL(D_0, D_{l'}) - IKL(D_0, D_l)$$
$$= \sum_{\beta_i \geq 0} (x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - l'\beta_i} - x_i^{(0)} \log \frac{x_i^{(0)}}{x_i^{(0)} - l\beta_i})$$
$$+ \sum_{\beta_i < 0} (x_i^{(l')} \log \frac{x_i^{(0)} - l'\beta_i}{x_i^{(0)}} - x_i^{(l)} \log \frac{x_i^{(0)} - l\beta_i}{x_i^{(0)}})$$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

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

8.2 Additional Experiment Results

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 3 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.

Scalability Comparison. Figure 4 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



(a) Varying Window Size (b) Varying Dimensions

Figure 4: Change Detection Running Time

r	DLIS PCA	Static PCA	RetrainPCA			Incr. PCA
			5k	10k	20k	
0.01	**0.86**	0.41	0.81	0.74	0.83	0.77
0.05	**0.91**	0.43	0.85	0.78	0.87	0.84
0.1	**0.92**	0.55	0.88	0.82	0.89	0.87

Table 8: PCA-based Outlier Detection with FC-G Dataset - F1-score - Varying Outlier Rate

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.

Stability with Outlier Rate. We compare the approaches when varying the outlier rate r from 0.01 to 0.1. Table 8 shows 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.