A PCA-Based Change Detection Framework for Multidimensional Data Streams

Abdulhakim Qahtan
King Abdullah University of Science and Technology
Thuwal, 23955, SA
abdulhakim.qahtan@kaust.edu.sa

Basma Alharbi
King Abdullah University of Science and Technology
Thuwal, 23955, SA
basma.harbi@kaust.edu.sa

Suojin Wang
Texas A&M University
College Station, TX, 77843, USA
sjwang@stat.tamu.edu

Xiangliang Zhang*
King Abdullah University of Science and Technology
Thuwal, 23955, SA
xiangliang.zhang@kaust.edu.sa

ABSTRACT

Detecting changes in multidimensional data streams is an important and challenging task. In unsupervised change detection, changes are usually detected by comparing the distribution in a current (test) window with a reference window. It is thus essential to design divergence metrics and density estimators for comparing the data distributions, which are mostly done for univariate data. Detecting changes in multidimensional data streams brings difficulties to the density estimation and comparisons. In this paper, we propose a framework for detecting changes in multidimensional data streams based on principal component analysis, which is used for projecting data into a lower dimensional space, thus facilitating density estimation and change-score calculations. The proposed framework also has advantages over existing approaches by reducing computational costs with an efficient density estimator, promoting the change-score calculation by introducing effective divergence metrics, and by minimizing the efforts required from users on the threshold parameter setting by using the Page-Hinkley test. The evaluation results on synthetic and real data show that our framework outperforms two baseline methods in terms of both detection accuracy and computational costs.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Data Streams, Density Estimation, Change Detection, Principal Component Analysis

*Corresponding author.

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.
ACM 978-1-4503-3664-2/15/08.
DOI: http://dx.doi.org/10.1145/2783258.2783359.

1. INTRODUCTION

Discovering changes in the properties of data is very important for building data mining models. Change detection covers a wide range of applications such as intrusion detection in computer networking [20], suspicious motion detection in vision systems [10], studying the effects of nuclear radiation on the environment [11], and online clustering and classification. For example, change detection can be used in online classification for reporting when the classifier should be re-trained (only if a change in the data stream is observed). The problem of change detection has been widely studied and referred to as data evolution [1], event detection [6] or change-point detection [9, 11].

Unsupervised window-based change detection is based on comparing the distribution in a current stream window with a reference distribution [2, 5, 9, 11, 12, 17], where density estimation techniques and divergence metrics are essential to model and compare the distributions. Modeling data distribution is a challenging task for multidimensional data, where most of the existing approaches either do not support data streams [12, 17] or work only for univariate data [2, 11, 19]. However, multidimensional data streams are ubiquitous in our application fields, because change detection in multidimensional data streams is receiving a lot of attention recently. Dasu et al. [5] used the *kdq-tree* data structure to model data distribution. However, this technique becomes inaccurate when data dimensionality increases to a moderate level, (e.g. 10). Kawahara and Sugiyama [9] used the density-ratio estimation that is based on the Kullback-Leibler Importance Estimation Procedure (KLIEP) to model data distribution. The method's complexity is quadratic with respect to (w.r.t.) the window size, which increases the computational costs significantly when large windows are used. Small windows make the model cheap, but they also make it sensitive and cause many false positives.

Here, we propose a new framework for detecting abrupt changes in unlabeled multidimensional data streams. The framework uses Principal Component Analysis (PCA) to project the original data stream into a lower dimensional space. Different types of changes in the original data variables, such as changes in mean and variance of data variables or changes in correlations between variables, can be observed on one or several Principal Components (PCs), which are orthogonal by their construction. In our proposed framework, data are projected onto each of the selected PCs, resulting in multiple univariate data streams, which exhibit variations if the

original data streams have changes. We monitor these streams of projections separately, and we quantitatively measure the variation in each stream by a change-score. Determining the final change-score is done by taking the aggregation from all selected components. The main advantage of monitoring changes on selected PCs separately is the reduction in computational costs, compared with monitoring all original variables individually or collectively.

In addition to using PCA, the accuracy and efficiency of our proposed framework in change detection are further assured by
1. Employing an accurate and efficient density estimation method, which inherits the efficiency of the KDE-Track in our previous study [16] and improves estimation accuracy by including a new method of allowing adaptive bandwidth settings.
2. Using divergence metrics for valid distribution comparisons: the Log-LikeliHood (LLH) [17], a symmetric Kullback-Leibler (KL) divergence, and the intersection area of two distributions.
3. Applying a dynamic threshold setting to change detection, which reduces the effort required from users for parameter's setting.

We validated the proposed framework on several synthetic and real world datasets including various types of changes. We also compared the performance of the framework with two baseline methods, namely *kdq-tree* [5] and another PCA -based change detection [12] that was designed differently not for data streams. The experimental results show that our framework with the divergence metric of the intersection area of the two density functions is more accurate in detecting changes in most of the datasets with fewer false alarms and higher efficiency when used with histograms. The results also show that our framework is robust against the parameters' settings and performs well for a wide range of the parameters.

The rest of the paper is organized as follows. Section 2 presents related work and Section 3 discusses the theoretical bases for our framework. Sections 4 and 5 present our change detection framework and evaluation results. Section 6 concludes our work with perspectives.

2. RELATED WORK

Table 1 summarizes the key characteristics of the existing methods for change detection: ability to handle MultiVariate (MV) data, applicability for streaming data, use of distribution comparison or predictive model, as well as how to set the threshold. As we study the change detection in unlabeled multivariate data streams, we focus our discussion on related work with MV=Yes.

Table 1: Summary of the key characteristics of change detection methods (MV = Multivariate).

Technique	MV	Data streams	Compare Distribution / Prediction	Threshold Settings
kdq-tree [5]	Yes	Yes	Compare Distribution	Bootstrap
PCA-SPLL [12]	Yes	No	Compare Distribution	Fixed
KLIEP [9]	Yes	Yes	Compare Distribution	Fixed
Kifer [11]	No	Yes	Compare Distribution	Bootstrap
ADWIN [2]	No	Yes	other	Dynamic
Density test [17]	Yes	No	Compare Distribution	Bootstrap
CF [19]	No	Yes	Prediction	Fixed

Dasu et al. [5] developed a window-based approach for detecting changes in unlabeled multidimensional data streams. Their approach follows a 3-step process: 1) updating the test distribution over the current window; 2) computing the change-score between the test and reference distributions; and 3) emitting an alarm signal if the change-score reaches the threshold specified using bootstrap sampling and the permutation test used in [11]. This approach is efficient because the change-score is updated from the previous value. However, modeling the distributions of the reference and test windows is done by space partitioning using *kdq-trees*, which becomes inaccurate when the dimension of the data increases.

A method called the Kullback-Leibler Importance Estimation Procedure (KLIEP) was proposed to directly measure the distribution difference without density estimation [18]. The method is unsupervised and works for multivariate data streams. An online version of KLIEP was studied in [9] and [15]. One common limitation of these methods is the user-based settings of key parameters, such as the threshold for change detection, regularization parameter λ, and learning rate η. Moreover, small windows have to be used to reduce the running time that cause false alarms of changes.

A statistical test, called the *density test* [17], determines whether the newly observed data S' are sampled from the same underlying distribution as the reference dataset S. The method works for unlabeled multivariate data; however, the high computational cost of the method reduces its applicability for data stream problems.

Change detection based on PCA in multidimensional data (not streams) was performed by considering only PCs with small eigenvalues in [12]. PCs with small eigenvalues are selected for reducing data dimensionally because they are analyzed to be more likely influenced by a change. However, there is an issue in the analysis of selecting such PCs: changes are assumed to occur in projected data on PCs with uniform distributions. In fact, changes occurring in data characteristics are demonstrated as variations in the original features'/variables' values. We will analyze and show that using PCs with large eigenvalues better capture the changes.

The framework proposed in this paper is featured as MV=Yes, Data Streams = Yes, Compare Distribution, Threshold Settings = Dynamic. It differs from existing approaches on 1) what data to monitor (original data or projections on PCs); 2) how to estimate data distribution functions; 3) how to compare distributions; and 4) how to set a threshold for reporting changes.

3. THEORETICAL ANALYSES

We discuss the building blocks for our framework of change detection in multidimensional data streams including PCA for change detection and divergence metrics for distribution comparison.

3.1 Principal Component Analysis (PCA)

Principal component analysis is a well known technique for dimensionality reduction. The central idea of PCA is to identify a set of orthogonal (uncorrelated) principal components from the original feature space such that the new space spanned by PCs better represents the data (i.e., the 1st PC captures the largest variability, the 2nd PC captures the 2nd largest variability, and so on). Let $S = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a dataset containing n data samples with $\mathbf{x}_i \in R^d$ and $i = 1, \cdots, n$. The first principal component $\mathbf{P}_1$ maximizes the variance of the projected data on it:

$$\mathbf{P}_1 = \underset{\|\mathbf{P}\|=1}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{P}^T \mathbf{x}_i - \mathbf{P}^T \overline{\mathbf{x}} \right)^2 = \underset{\|\mathbf{P}\|=1}{\operatorname{argmax}} \mathbf{P}^T \Sigma \mathbf{P},$$

where $\Sigma = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}}) (\mathbf{x}_i - \overline{\mathbf{x}})^T$ is the sample covariance matrix and $\overline{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$ is the sample mean. The residual $\mathbf{x}'_i = \mathbf{x}_i - \mathbf{P_1} \mathbf{P_1}^T \mathbf{x}_i$, represents the distortion introduced by reducing the dimensionality to 1. The second principal component $\mathbf{P}_2$ is selected in the direction that maximizes the projected variance from the set of vectors that are orthogonal to $\mathbf{P}_1$. Let $S' = \{\mathbf{x}'_1, \cdots, \mathbf{x}'_n\}$ be the set of the residuals; $\mathbf{P}_2$ is defined as:

$$\mathbf{P}_2 = \underset{\substack{\|\mathbf{P}\|=1, \\ \mathbf{P}^T \mathbf{P}_1 = 0}}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{P}^T \mathbf{x}'_i - \mathbf{P}^T \overline{\mathbf{x}'} \right)^2 = \underset{\substack{\|\mathbf{P}\|=1, \\ \mathbf{P}^T \mathbf{P}_1 = 0}}{\operatorname{argmax}} \mathbf{P}^T \Sigma \mathbf{P}.$$

In other words, the k-th principal component is equal to the eigenvector that corresponds to the k-th largest eigenvalue of the covariance matrix Σ.

3.2 PCA for Change Detection

Change detection in data streams refers to the problem of finding time points, where for each point, there exists a significant change in the current data distribution. A typical window-based solution is to extract a fixed S_1 (reference window) from streaming samples and to update an S_2 (test window) with newly arriving samples [5, 11]. Changes are then detected by measuring the difference between the distributions in S_1 and S_2.

Modeling the data distribution and selecting a comparison criterion are essential for change detection in data streams. However, density estimation of multidimensional data is difficult. It becomes less accurate and more computationally expensive with increasing dimensionality. In our framework, we apply PCA to project the multidimensional data samples from the stream on the principal components to obtain multiple streams of 1D data. Density estimation, distribution comparison, and change-score calculations can then be conducted in parallel on those 1D data streams. Compared with projecting the data on the original coordinates (i.e., using the original variables), projecting on PCs has the following advantages: 1) it allows the detection of changes in data correlations, which cannot be detected in the original individual variables; 2) it can guarantee that any changes in the original variables are reflected in PC projections; and 3) it reduces the computation cost by discarding trivial PCs. We use a 2D Gaussian example to theoretically analyze how different types of changes are reflected in PC projections.

For simplicity, we assume the data without change has a mean $\mu = [0, 0]^T$ and the covariance matrix $\Sigma_1 = \begin{bmatrix} \sigma^2 & r \\ r & \sigma^2 \end{bmatrix}$. A widely used divergence metric, KL-divergence is used for distribution comparison, which is defined as

$$D_{KL}\left(g||f\right) = \int_x g(x) \log\left(\frac{g(x)}{f(x)}\right) dx. \tag{1}$$

Here, f represents the Probability Density Function (PDF) before the change and g represents the PDF after the change.

LEMMA 1. *For univariate normal distributions f, g where $f \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $g \sim \mathcal{N}(\mu_2, \sigma_2^2)$*

$$D_{KL}\left(g||f\right) = \frac{\left(\mu_1 - \mu_2\right)^2}{2\sigma_1^2} + \frac{1}{2}\left(\frac{\sigma_2^2}{\sigma_1^2} - 1 - \log\frac{\sigma_2^2}{\sigma_1^2}\right). \tag{2}$$

PROOF. *Eq. (2) is obtained by substituting the normal distribution expression in Eq. (1) and simplifying the formula.* □

1) Case 1: change of correlation

When changes happen to the correlation of the two variables, the covariance matrix after the change can be $\Sigma_2 = \begin{bmatrix} \sigma^2 & r+\eta \\ r+\eta & \sigma^2 \end{bmatrix}$, where $\eta > 0$.

If PCA is not applied, data are projected on the original coordinates $\mathbf{u}_1 = [1, 0]^T$ and $\mathbf{u}_2 = [0, 1]^T$. Before the change of correlation, the projection on $\mathbf{u}_1$ has the distribution $f_1^u \sim \mathcal{N}(0, \mathbf{u}_1^T\Sigma_1\mathbf{u}_1) = \mathcal{N}(0, \sigma^2)$. After the change, the projection on $\mathbf{u}_1$ has the distribution $g_1^u \sim \mathcal{N}(0, \mathbf{u}_1^T\Sigma_2\mathbf{u}_1) = \mathcal{N}\left(0, \sigma^2 + \mathbf{u}_1^T\begin{bmatrix} 0 & \eta \\ \eta & 0 \end{bmatrix}\mathbf{u}_1\right) = \mathcal{N}(0, \sigma^2)$, which is the same as f_1^u. The KL-divergence measuring the difference between f_1^u and g_1^u can be calculated by Eq. (2) $D_{KL}(g_1^u||f_1^u) = 0$.

Similarly, the projection on $\mathbf{u}_2$ before the change has the distribution $f_2^u \sim \mathcal{N}(0, \mathbf{u}_2^T\Sigma_1\mathbf{u}_2) = \mathcal{N}(0, \sigma^2)$. After the change, the projection is $g_2^u \sim \mathcal{N}(0, \mathbf{u}_2^T\Sigma_2\mathbf{u}_2) = \mathcal{N}\left(0, \sigma^2 + \mathbf{u}_2^T\begin{bmatrix} 0 & \eta \\ \eta & 0 \end{bmatrix}\mathbf{u}_2\right)$ $= \mathcal{N}(0, \sigma^2)$, which is the same as f_2^u. That is, $D_{KL}(g_2^u||f_2^u) = 0$ too, and the change will not be detected.

If we apply PCA, we obtain the first PC $\mathbf{P}_1 = [1, 1]^T/\sqrt{2}$ with eigenvalue $\lambda_1 = \sigma^2 + r$ and the second PC $\mathbf{P}_2 = [-1, 1]^T/\sqrt{2}$ with eigenvalue $\lambda_2 = \sigma^2 - r$. On $\mathbf{P}_1$, the projection before the change is $f_1^P \sim \mathcal{N}(0, \mathbf{P}_1^T\Sigma_1\mathbf{P}_1) = \mathcal{N}(0, \lambda_1)$ and after the change is $g_1^P \sim \mathcal{N}(0, \mathbf{P}_1^T\Sigma_2\mathbf{P}_1) = \mathcal{N}\left(0, \lambda_1 + \mathbf{P}_1^T\begin{bmatrix} 0 & \eta \\ \eta & 0 \end{bmatrix}\mathbf{P}_1\right) = \mathcal{N}(0, \lambda_1 + \eta)$. The KL-divergence value when comparing f_1^P and g_1^P is $D_{KL}(g_1^P||f_1^P) = \frac{1}{2}\left(\frac{\eta}{\lambda_1} - \log\frac{\lambda_1+\eta}{\lambda_1}\right)$.

Similarly on the second PC, the projection before change is $f_2^P \sim \mathcal{N}(0, \mathbf{P}_2^T\Sigma_1\mathbf{P}_2) = \mathcal{N}(0, \lambda_2)$, and the projection after change is $g_2^P \sim \mathcal{N}(0, \mathbf{P}_2^T\Sigma_2\mathbf{P}_2) = \mathcal{N}\left(0, \lambda_2 + \mathbf{P}_2^T\begin{bmatrix} 0 & \eta \\ \eta & 0 \end{bmatrix}\mathbf{P}_2\right) = \mathcal{N}(0, \lambda_2 - \eta)$. From Eq. (2), the change-score $D_{KL}(g_2^P||f_2^P) = \frac{1}{2}\left(-\frac{\eta}{\lambda_2} + \log\frac{\lambda_2}{\lambda_2-\eta}\right)$. We can see that $D_{KL}(g_i^P||f_i^P) > 0$ with $i = 1, 2$, which means that changes will be detected on PCs.

In conclusion, in the case of correlation changes, zero change-scores are obtained when checking the difference in the original variables. However, non-zero change-scores are calculated in PCA projections, which indicates that changes are detectable.

2) Case 2: change of variance

When changes happen to the variance of the two variables, the covariance matrix after the change can be $\Sigma_2 = \begin{bmatrix} \sigma^2+\eta_1 & r \\ r & \sigma^2+\eta_2 \end{bmatrix}$, where $\eta_i \geq 0, i = 1, 2$.

In the original space, before and after the change, we compare the projections on $\mathbf{u}_1$, $f_1^u \sim \mathcal{N}(0, \sigma^2)$ and $g_1^u \sim \mathcal{N}(0, \mathbf{u}_1^T\Sigma_2\mathbf{u}_1) = \mathcal{N}(0, \sigma^2 + \eta_1)$. Similarly on $\mathbf{u}_2$, the projections before and after the change are $f_2^u \sim \mathcal{N}(0, \sigma^2)$ and $g_2^u \sim \mathcal{N}(0, \mathbf{u}_2^T\Sigma_2\mathbf{u}_2) = \mathcal{N}(0, \sigma^2 + \eta_2)$, respectively. Then we have $D_{KL}(g_i^u||f_i^u) = \frac{1}{2}\left(\frac{\eta_i}{\sigma^2} - \log\frac{\sigma^2+\eta_i}{\sigma^2}\right), i = 1, 2$.

If we apply PCA, before and after the change, we have on $\mathbf{P}_1$, $f_1^P \sim \mathcal{N}(0, \lambda_1)$ and $g_1^P \sim \mathcal{N}(0, \mathbf{P}_1^T\Sigma_2\mathbf{P}_1) = \mathcal{N}(0, \lambda_1 + \frac{1}{2}(\eta_1 + \eta_2))$. Similarly on $\mathbf{P}_2$, the projections before and after changes are $f_2^P \sim \mathcal{N}(0, \lambda_2)$ and $g_2^P \sim \mathcal{N}(0, \mathbf{P}_2^T\Sigma_2\mathbf{P}_2) = \mathcal{N}(0, \lambda_2 + \frac{1}{2}(\eta_1 + \eta_2))$, respectively. Then the change-score is $D_{KL}(g_i^P||f_i^P) = \frac{1}{2}\left(\frac{\eta_1+\eta_2}{2\lambda_i} - \log\frac{2\lambda_i+\eta_1+\eta_2}{2\lambda_i}\right), i = 1, 2$. We see that any changes with positive η_1 or η_2 can result in non-zero $D_{KL}(g_i^P||f_i^P), i = 1, 2$. In other words, changes in the variances of the original variables are observable on projections of PCs.

3) Case 3: change of mean

Assume that the mean of the data shifts to a new one, $\mu_{new} = [\eta_1, \eta_2]^T$. In the original space, the projections on $\mathbf{u}_1$ before and after the change are $f_1^u \sim \mathcal{N}(0, \sigma^2)$ and $g_1^u \sim \mathcal{N}(\eta_1, \sigma^2)$, respectively. Similarly on $\mathbf{u}_2$, the projections are $f_2^u \sim \mathcal{N}(0, \sigma^2)$ and $g_2^u \sim \mathcal{N}(\eta_2, \sigma^2)$, respectively. Then $D_{KL}(g_i^u||f_i^u) = \frac{\eta_i^2}{2\sigma^2}, i = 1, 2$.

If PCA is applied, the projections on $\mathbf{P}_1$ before and after the change are $f_1^P \sim \mathcal{N}(0, \lambda_1)$ and $g_1^P \sim \mathcal{N}(\mu_{new}^T\mathbf{P}_1, \lambda_1)$, respectively. Then $D_{KL}(g_1^P||f_1^P) = \frac{(\eta_1+\eta_2)^2}{4\lambda_1}$. Similarly on $\mathbf{P}_2$, we compare $f_2^P \sim \mathcal{N}(0, \lambda_2)$ and $g_2^P \sim \mathcal{N}(\mu_{new}^T\mathbf{P}_2, \lambda_2)$. Then $D_{KL}(g_2^P||f_2^P) = \frac{(\eta_1-\eta_2)^2}{4\lambda_2}$. We can see that a mean shift occurring at any original variables is reflected in the projections on PCs,

leading to the non-zero change-score $D_{KL}(g_i^P||f_i^P), i = 1, 2$. There is one special case where the mean shifts in the direction in parallel with one PC $\mathbf{P}_i$. For example, $\eta_1 = \eta_2$ in μ_{new} indicates the mean shifts in the direction of $\mathbf{P}_1$. In such a case, change-score on the other PC is zero, except that on this PC $\mathbf{P}_i$, (e.g., $D_{KL}(g_1^P||f_1^P)$ is non-zero, and $D_{KL}(g_2^P||f_2^P) = 0$ in the example). It is worth noting that the shift in the direction of $\mathbf{P}_i$ is fully captured by the projection on $\mathbf{P}_i$, both the magnitude and the direction.

Our analysis shows that changes in the original variables are reflected in PC projections. However, it is expensive to examine projections of each PC to detect changes, especially when data dimensionality is high. We select the first k PCs with large eigenvalues λ_i that satisfy $\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j} \geq 0.999$. Such a threshold on the percentage of explained variance is a conservative setting. It ensures selecting a sufficient number of PCs even when the distribution of λ is flat. An experimental evaluation of this threshold setting will be presented in Section 5. The discarded PCs have very small (negligible) eigenvalues. The variance of projected data on these PCs are also very small (0.1% at most). Densities with small variances are difficult to estimate and to compare due to their sensitivity to sample size and density model parameters. False alarms of changes may be reported due to the error in the density estimation. Thus, discarding such PCs has benefits in reducing the computational cost and minimizing false positive rates. When PCA is employed in [12] for change detection, PCs with small eigenvalues are selected but those with large eigenvalues are discarded, according to the analysis that changes are more noticeable from the projected data on the space spanned by PCs with small eigenvalues. It should be noted that the analysis in [12] assumed changes to occur in the projected data on PCs. However, in practice, changes of data characteristics are demonstrated as variations in the original features'/variables' values.

3.3 Divergence Metrics

Divergence metrics are crucial for computing change-scores. The KL-divergence defined in Eq. (1) is a non-negative (≥ 0) and non-symmetric measure. It is 0 when the two distributions are completely identical, and becomes larger as the two distributions deviate from each other. The non-symmetry property of the D_{KL} complicates the procedure of setting the threshold for detecting changes in data streams. To overcome the problem of the KL-divergence, we focus our study on three important divergence metrics. The first divergence metric is a modified symmetric KL-divergence [14]

$$D_{MKL}(\hat{g}||\hat{f}) = \max\left(D_{KL}\left(\hat{g}||\hat{f}\right), D_{KL}\left(\hat{f}||\hat{g}\right)\right). \quad (3)$$

This divergence metric mimics powerful order selection tests developed in current statistics literature; see, for example, [8].

The second divergence metric we use is a measure of the intersection area under the curves of two density functions [3],

$$D_A\left(\hat{g}||\hat{f}\right) = 1 - \int_x \min\left(\hat{f}(x), \hat{g}(x)\right) dx. \quad (4)$$

This D_A takes values in $[0, 1]$, where the value one means that the two distributions are completely different and zero means the two distributions are identical. This measure can also be computed using numerical integration techniques on the intersection area.

The third divergence metric that we study is based on measuring the likelihood of the data samples in the test window to be extracted from the same distribution as the reference window. It is a modified version of the LLH metric used in [17]. Let $\hat{f}$ be the density function estimated using data samples in reference window S_1. The log-likelihood of the test window S_2 w.r.t. $\hat{f}$ is $LLH(S_2|\hat{f}) = \log\left\{\prod_{x \in S_2} \hat{f}(x)\right\}$. The divergence between S_1 and S_2 is

$$D_{LLH}(S_1, S_2|\hat{f}) = \frac{LLH(S_2|\hat{f})}{|S_2|} - \frac{LLH(S_1|\hat{f})}{|S_1|}. \quad (5)$$

When the data distribution is stationary and the samples in S_1 and S_2 are independent and identically distributed, D_{LLH} will have a limiting normal distribution with mean 0. The normality of D_{LLH} was used in [17] to define a critical region such that the divergence values falling in that region indicate a change in the distribution from S_1 to S_2.

When there is a change, D_{LLH} can be extremely positive or negative depending on the type of change. In order to use this metric in our framework, we take

$$D_{LLH}(S_1, S_2|\hat{f}) = \left|\frac{LLH(S_2|\hat{f})}{|S_2|} - \frac{LLH(S_1|\hat{f})}{|S_1|}\right| \quad (6)$$

as the divergence metric, which takes values in the interval $[0, \infty)$.

This metric is asymmetric and the change from S_2 to S_1 might be easier to detect. Song et al. [17] suggested using the average of $D_{LLH}(S_1, S_2|\hat{f})$ and $D_{LLH}(S_1, S_2|\hat{g})$ as the final change-score, where $\hat{g}$ is the PDF estimated using the samples in S_2. However, computing the $D_{LLH}(S_1, S_2|\hat{g})$ will require computing $LLH(S_1|\hat{g})$ after receiving every data sample from the stream as the density function $\hat{g}$ will change, which is very expensive to deploy for data streams. We leave this issue for future study.

4. CHANGE DETECTION FRAMEWORK

In this section, we present our framework for change detection in data streams. The different components of the framework will be discussed in detail including the divergence metrics, the density estimation, and the dynamic threshold setting. The framework is given in Algorithm 1, where D_M denotes any divergence metric.

4.1 Setting Windows

Line 3 in Algorithm 1 sets the reference window S_1 to be the first w samples arriving after the change point t_c. Intuitively, when a data distribution shifts to a new one, the reference window should be updated to represent the new distribution. This update also enables the detection of further changes. Line 8 sets the test window S_2 as a collection of w samples after the reference window. This S_2 will slide along the data stream to include the newest w samples.

The window size w is usually set according to the application problems. A small window size will allow for detecting short-term changes and reduces the delay but may lead to a large number of false positives. A large window size will make the algorithm more robust against short-term changes but may miss alarms [1]. The setting of this parameter is usually left to the user to give them the ability to monitor the long/short term changes, depending on their interests and the application sensitivity against changes [1].

4.2 Projecting the Data

After receiving the first w data samples in reference window S_1, PCA is applied to extract the principal components from S_1. The first k PCs with the largest eigenvalues are selected if they account for 99.9% of data variance (i.e., $\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j} \geq 0.999$). The data in the reference and test windows are then projected on these k components. On each component, projections of the reference and test windows are compared and a change-score value is recorded.

Algorithm 1 Change Detection Framework

Parameters: window size w, ξ, δ
Online flow in: streaming data $S = \{\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots\}$
Online output: time t when detecting a change
Procedure:

1: Initialize $t_c = 0$, $step = min(0.05w, 100)$
2: Initialize $\overline{Sc}, m, M$ to NULL.
3: Set reference window $S_1 = \{\mathbf{x}_{t_c+1}, \cdots, \mathbf{x}_{t_c+w}\}$
4: Extract principal components by applying PCA on S_1 to obtain $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_k$
5: Project S_1 on $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_k$ to obtain $\breve{S}_1$
6: $\forall\, i\ (1 \leq i \leq k)$ estimate $\hat{f}_i$ using data of the i-th component of $\breve{S}_1$
7: Clear S_1 and $\breve{S}_1$
8: Set test window $S_2 = \{\mathbf{x}_{t_c+w+1}, \cdots, \mathbf{x}_{t_c+2w}\}$
9: Project S_2 on $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_k$ to obtain $\breve{S}_2$
10: Clear S_2
11: Estimate $\hat{g}_i$ using data of the i-th component of $\breve{S}_2$
12: **while** a new sample $\mathbf{x}_t$ arrives in the stream **do**
13: Project $\mathbf{x}_t$ on $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_k$ to obtain $\breve{\mathbf{x}}_t$
14: Remove $\breve{\mathbf{x}}_{t-w}$ from $\breve{S}_2$
15: $\forall\, i\ (1 \leq i \leq k)$ update $\hat{g}_i$ using $\breve{\mathbf{x}}_t^{(i)}$ and $\breve{\mathbf{x}}_{t-w}^{(i)}$
16: **if** $mod(t, step) = 0$ **then**
17: $curScore = \max_i \left(D_M \left(\hat{g}_i || \hat{f}_i \right) \right)$
18: **if Change**($curScore, \overline{Sc}, m, M, \xi, \delta$) **then**
19: Report a change at time t and set $t_c = t$
20: Clear $\breve{S}_2$ and GOTO step 2
21: **Subprocedure: Change**($curScore, \overline{Sc}, m, M, \xi, \delta$)
22: Update $\overline{Sc}$ to include the $curScore$ in the average.
23: $new\_m = m + \overline{Sc} - curScore + \delta$
24: **if** $|new\_m| > M$ **then**
25: $new\_M = new\_m$.
26: $\tau_t = \xi * \overline{Sc}$
27: **if** $curScore > \tau_t$ **then**
28: return True
29: **else**
30: $M = new\_M, m = new\_m$
31: return False

The maximum value among the k change-score values is considered as the final change-score. Any new data sample is projected on the k components and the density functions of the projection of the test window are updated and compared with the reference densities.

When the number of considered principal components k is very large, maintaining an accurate PDF for data projections becomes very expensive. A tradeoff between accuracy and efficiency should be considered. Histograms, the oldest and most efficient density estimators, can be used to estimate PDFs $\hat{f}_i$ and $\hat{g}_i$. The bins of the histogram are specified using an origin point x_0 and a bin width h. The bins are defined as the intervals $[x_0 + mh, x_0 + (m+1)h]$ for positive and negative integers m. The density at a given point is estimated using $\hat{p}(x) = N_i/(N * h)$, where N is the total number of samples and N_i is the number of samples in the same bin as x. Note that the bin width of the histogram can vary from one bin to another. Histograms provide discontinuous, less accurate density estimation. The estimated density might change considerably based on the selected origin point x_0 and the bin width h. For such reasons, histograms can be used only when efficiency is an important issue. In this paper, we use histograms for estimating the density in experiments with high-dimensional data.

4.3 Estimating Density Functions

Because the change-score is used directly to trigger change alarms, PDFs for distribution comparison must be accurately and efficiently estimated. KDE-Track, a dynamic density estimator we studied in [16], adapts KDE to handle the evolving underlying distribution in data streams. It gains linear time complexity by adopting linear interpolation and adaptive resampling. However, it inherits the issue of bandwidth setting in KDEs. A small bandwidth value causes more fluctuation in the density function's curve, which might give incorrect information about the density, whereas selecting a large bandwidth value over-smoothes the function's curve and hides information. In data streams, density estimators should update the bandwidth h online rather than using a fixed setting.

In [16], we selected the bandwidth for KDE-Track using the normal rule, which is the most widely used method for selecting the bandwidth. It selects the bandwidth with the assumption that the density is normal. It is efficient, but the estimated densities are over-smoothed when the density deviates much from normality, (e.g., in the cases of multimodal data).

In this paper, we employ KDE-Track with an adaptive bandwidth h for density estimation. Merits of both efficiency and accuracy advance the calculation of divergence scores and thus change detection. We minimize the effect of the normality assumption of f by estimating f'', the second derivative of f. The motivation for doing this is that the estimated $\hat{f}''$ may often be a better approximation of f'' than assuming f to be a normal density. Using the KDE-Track model allows for estimating $\hat{f}''$ in constant time and memory cost. In this way, the estimated $\hat{f}''$ will serve two roles. First, it is used to approximate $R(f'') = \int (f''(x))^2\, dx$ to estimate the bandwidth $\hat{h}$. Second, it is used as a more accurate indicator of the high curvature of the density function, which facilitates the adaptive resampling in KDE-Track for obtaining more accurate estimation.

The new KDE-Track[1], a more advanced density estimator, is used for estimating the density functions of the projected data of S_1 (line 6) and S_2 (line 11) on the considered PCs, as well as for updating the test densities when a new sample arrives (lines 13-15). Note that the update at line 15 ensures that $\hat{g}_i$ is the current distribution of the i-th component in the data stream.

4.4 Computing the Change-Score Values

Change-scores are computed by using a divergence metric on two density functions $\hat{f}_i$ and $\hat{g}_i$ (line 17), which are updated upon the arrival of each sample. However, it is not necessary to compute change-score at each time step, as the change of a distribution cannot be observed after a single data sample. Therefore, to reduce unnecessary repeated comparisons that may increase the execution time noticeably, we compute change-scores every $\min(0.05 * w, 100)$ samples (line 16). This setting of checkpoints complies with the monitoring requirements of users. Monitoring short-term changes with a small w needs frequent check points while monitoring long-term changes by setting a large w allows bigger checkpoint intervals. The granularity can be adjusted by changing the 5% according to the users' needs. When using histograms for density estimation, it is feasible to compute the change-score after receiving every data sample. Note that the change detection methods based on our framework are named according to which divergence metric is used. The method use the maximum KL-divergence (Eq. (3)) is called CD-MKL, the one that uses the Area metric (Eq. (4)) is called CD-Area, and the one that uses the LLH metric (Eq. (6)) is called CD-LLH.

After computing the change-scores on all the PCs, the different change-score values are aggregated by taking the maximum over all values. This is necessary to maintain a single statistical quan-

[1]Demos, code and additional evaluation results are available at: http://mine.kaust.edu.sa/Pages/Software.aspx

tity. The maximum is preferable for aggregating the change-score values as it allows for treating any changes happening at any component equally important. Also, when a change happens in a single PC, the change-score will not be affected by the small change-score values obtained from the other PCs.

4.5 Dynamic Threshold Settings

Typical statistical tests for change detection start by considering the null hypothesis, which assumes that the data distribution is stationary. A change-score value is then calculated to determine the probability of rejecting the null hypothesis. The most popular technique to reject the null hypothesis is to specify a threshold and declare a change whenever the change-score becomes greater than the threshold. Most existing methods require a user-specified threshold [4], which has two main issues. First, the fixed threshold cannot be used to detect changes in different magnitudes. Second, the threshold is difficult to set, as it is sensitive to the divergence metric, window size, underlying distribution, and change types. In this work, we propose a dynamic threshold setting, which is adjustable to the evolving distribution (procedure **change**, lines 21-31).

Since evolving data streams can be viewed as concatenated segments such that the distribution of each segment is different from the distribution of its neighboring segments [7], we learn first the regularity of the data by recording the change-scores between the reference and test windows. A change can be detected by monitoring how the current change-score deviates from the historical values (i.e., by comparing the current change-score with the mean of historic values since the last reported change).

The divergence metrics that we are studying generate small values (close to zero) when the two windows have the same distribution. The change-score values increase when more data samples from the new distribution arrive, as well as the mean value of the change-scores. We employ the Page-Hinley (PH) test to track changes in the change-score. A change is reported when the current change-score value significantly deviates for a reasonable period of time from the history of the score values.

The PH test is designed as a sequential test to detect changes in the average of a Gaussian signal. The test defines a cumulative variable (m_t) to store the cumulated difference between the observed values and the mean of the previously observed values, defined as:

$$m_t = \sum_{t'=1}^{t} (\overline{s}_{t'} - s_{t'} + \delta), \tag{7}$$

where $s_{t'}$ is the observed value at time t', $\overline{s}_{t'} = 1/t' \sum_{i=1}^{t'} s_i$, δ is the magnitude of the allowed change, which is often set close to zero, (e.g., half the minimum of the obtained change-score values in [21]). A change is reported when the difference between $M_t = \max\{m_1, \cdots, m_t\}$ and m_t is greater than a threshold value θ. The value of θ controls the sensitivity of the model. A large θ makes the model more robust but changes might go undetected, and a small θ makes small changes detectable but increases the false alarm rate.

We use a dynamic threshold setting which adjusts θ_t according to the observed change-score values, $\theta_t = \xi * \overline{s}_t$, where ξ is a constant value called the θ-factor and represents the number of witnesses that should be observed before declaring a change [21]. This approach has a high accuracy for setting the threshold value, which results in higher detection accuracy with less false positives as we will demonstrate in the experimental evaluation section.

5. EXPERIMENTAL EVALUATION

This section describes the evaluation of the proposed change detection framework. The performance of the framework is compared with the performance of two baseline methods presented in [5] and [12]. Our framework uses three divergence metrics presented in Eqs. (3), (4) and (6). The performance of the different methods is measured according to the number of True Positives (TP), Late detections (L), False Positives (FP) and False Negatives (FN). By true positives, we mean the changes that were reported correctly before receiving $2w$ from the new distribution where w is the window size. Late detections are the changes reported after processing $2w$ data samples from the new distribution. False positives are changes reported by the method when there are no changes, and false negatives are the missed changes. The experimental results show that our framework outperforms the two baseline methods in terms of the number of correctly detected changes with less false positives. Also, the performance of the framework when using the Area metric (Eq. (4)) outperforms the performance of the framework when using the MKL (Eq. (3)) and LLH (Eq. (6)) metrics in most of the evaluation datasets. We will start our discussion by providing more detail about the baseline methods before we present the experimental results.

The first baseline change detection method was proposed by Dasu et al. in [5], which adopts a spatial partitioning scheme to construct an empirical distribution of the data. The partitioning scheme, *kdq-tree*, partitions the data space into separate cells not necessarily covering the whole space of the data. Each node in the *kdq-tree* has two variables to store the number of data samples falling within that cell from the reference and test windows. The divergence metric is used to compute the change-score between the reference and test windows. When the change-score is greater than a threshold determined using a bootstrap technique, a change in the data distribution is reported. Frequent changes require repeating the bootstrapping routine for the new reference window in order to obtain a threshold for the new distribution. This method uses both the Kullback-Leibler (KL) and the Average Absolute Difference (AAD) as divergence metrics. We were able to reproduce the results reported in [5] using the AAD metric[2], which is defined as $D(S_1, S_2) = \frac{1}{n} \sum_{i=1}^{n} |n_{ri} - n_{ti}|$, where n is the window size, n_{ri} (n_{ti}) is the number of data samples from the reference (test) window that falls in the i-th node. The results obtained using the KL-divergence are inaccurate with many false negatives. The method with AAD seems to work well for datasets with dimensions less 10 but it is vulnerable to the curse of dimensionality as we will see in the experimental results.

The second baseline method was proposed by Kuncheva and Faithfull in [12]. The method employs PCA to reduce data dimensionality by eliminating the PCs with large variances, and uses a newly proposed asymmetric divergence metric called the Semi-Parametric Log-Likelihood (SPLL) to measure distribution difference. A threshold for reporting changes is left for the user to tune.

5.1 Experiments on Synthetic Data

The first experiment evaluates the performance of our change detection framework and compares it with the two baseline methods using two-dimensional datasets with three types of changes. Each dataset contains 5×10^6 data samples with changes that occur every 5×10^4 data samples with a total of 99 change points. The datasets are generated following the same generation mechanism of [5] and contain the same types of changes. The datasets are given symbols to indicate the type of change ($M(\epsilon)$ means varying the mean value, $D(\epsilon)$ means varying the standard deviation, and $C(\epsilon)$ means varying the correlation). At each change point, a set of random numbers in the interval $[-\epsilon, -\epsilon/2] \cup [\epsilon/2, \epsilon]$ are generated and added to the

[2]We thank the authors for providing the code to replicate the results.

Table 2: Evaluation results of Dasu's method (*kdq-tree*) with AAD and KL-divergence metrics, Kuchneva's method (PCA-SPLL), CD-Area, CD-LLH, and CD-MKL with KDE-Track and histograms as density estimators. The results are in the form TP/L/FP/FN with TP = True Positives, L = Late detections, FP = the False Positives and FN = the False Negatives. The best results are in bold.

Dataset	*kdq-tree* [5]		PCA-SPLL [12]	CD-Area		CD-LLH		CD-MKL	
	AAD	KL		Histograms	KDE-Track	Histograms	KDE-Track	Histograms	KDE-Track
M(0.01)	30/15/1/54	3/7/0/89	10/16/5/73	25/17/0/57	**38/23/0/38**	28/23/7/48	10/11/0/78	34/12/0/53	34/31/1/34
M(0.02)	77/14/6/8	3/7/0/89	18/9/3/72	69/14/1/16	**87/12/0/0**	39/27/5/33	23/16/1/60	78/10/1/11	89/7/0/3
M(0.05)	98/1/4/0	12/21/0/66	66/11/9/42	96/0/3/3	**99/0/0/0**	76/6/1/17	70/6/2/23	97/0/2/2	95/0/4/4
D(0.01)	42/18/2/39	4/2/0/93	29/5/4/65	32/13/0/54	45/29/0/25	84/0/10/15	**84/6/3/9**	41/17/0/41	54/25/0/20
D(0.02)	98/1/9/0	12/7/0/80	85/0/2/14	94/0/0/5	97/0/1/2	87/5/8/7	93/1/4/5	**98/1/0/0**	94/1/2/4
D(0.05)	99/0/2/0	24/4/2/71	89/2/2/8	**99/0/0/0**	**99/0/0/0**	97/0/3/2	98/0/1/1	98/0/1/1	98/0/2/1
C(0.1)	67/13/2/19	8/4/1/87	55/4/2/40	69/9/0/21	68/19/0/12	89/1/6/9	**93/0/4/6**	60/17/1/22	76/17/2/6
C(0.15)	78/9/5/12	8/7/1/84	63/6/4/30	68/5/2/26	84/7/1/8	90/3/5/6	**94/1/2/4**	81/2/3/16	85/9/0/5
C(0.2)	96/3/10/0	21/5/3/73	75/3/3/21	93/1/1/5	**97/1/1/1**	97/0/1/2	97/0/2/2	93/1/3/5	93/2/0/4

distribution's parameters that will be changed. The parameter ϵ controls the magnitude of the change, where smaller values for ϵ make changes harder to detect and vice versa.

In the first three datasets $M(0.01)$, $M(0.02)$ and $M(0.05)$, the standard deviation values are fixed ($\sigma_1 = \sigma_2 = 0.2$) and the correlation coefficient is fixed ($\rho = 0.5$). The mean values μ_1, μ_2 are changing by η_1, η_2 randomly selected from the interval $[-\epsilon, -\epsilon/2] \cup [\epsilon/2, \epsilon]$. The second three datasets $D(0.01)$, $D(0.02)$ and $D(0.05)$ are generated by fixing the mean values $\mu_1 = \mu_2 = 0.5$ and the correlation coefficient value $\rho = 0.5$. The standard deviation values σ_1, σ_2 are changing by adding random values from $[-\epsilon, -\epsilon/2] \cup [\epsilon/2, \epsilon]$. In the last three datasets $C(0.01)$, $C(0.02)$ and $C(0.05)$, the mean values and the standard deviation values are fixed $\mu_1 = \mu_2 = 0.5$ and $\sigma_1 = \sigma_2 = 0.2$. The coefficient ρ makes random walks in the interval $(-1, 1)$ with random steps selected from $[-\epsilon, -\epsilon/2] \cup [\epsilon/2, \epsilon]$.

The parameters' setting for the *kdq-tree* method follow the settings in [5] with $\delta = 0.01$, $k = 500$ and the window size is set to 10^4. For CD-LLH, CD-Area and CD-MKL, the parameters of the PH test for setting the threshold are set as $\delta = 0.005$ and $\theta_t = \xi * \overline{s}_t = 500 * \overline{s}_t$. The window size is set to 10^4.

Table 2 presents the results obtained by the evaluated methods on the nine datasets. When deployed with the KL-divergence metric, the *kdq-tree* method, has poor performance. The *kdq-tree* method with AAD metric, CD-Area, and CD-MKL have comparably good results with CD-Area showing more accurate results. CD-LLH has lower performance on datasets with varying means while it shows good performance when the changes affect the standard deviation or the correlation. PCA-SPLL [12] shows the worst performance, especially for detecting mean shifts.

The second set of experiments evaluates the detection accuracy in high-dimensional datasets. As changes become less observable when the data dimensionality increases, we used only datasets with a reasonable magnitude of change ($M(0.05)$, $D(0.05)$, and $C(0.2)$). For each type of change, we generated three datasets with $10, 20$, and 30 dimensions. The changes in the data distribution affect only two dimensions (variables) and for more complicated change detection cases, we selected the variables that are affected by the change randomly at each change point. When the changes affect the same variables, the methods' accuracy is comparable to the accuracy on the 2D data.

We report the results for the *kdq-tree* method with AAD-divergence only because KL-divergence shows very low accuracy for 2D data. The results for CD-LLH, CD-MKL, and CD-Area were obtained using histograms since KDE-Track becomes expensive for high-dimensional data especially when all the PCs are considered. Using histograms improves the running time noticeably without affecting the performance of the framework in reporting the changes correctly and maintaining a low number of false positives.

The results in Table 3 show that CD-Area and CD-MKL are not affected by data dimensionality. CD-LLH performs better when the changes affect the data variances and correlation. Like all space partitioning methods, the *kdq-tree* method, suffers from the curse of dimensionality as data dimensionality increases. The PCA-SPLL has low accuracy in detecting the changes again because the PCs with large eigenvalues that we selected in our framework are dismissed and only the PCs with small eigenvalues are used for data projection and for change detection. The changes in the directions of the major (ignored) PCs will not be detected.

Table 3: Evaluation results of *kdq-tree* with AAD metric, PCA-SPLL, CD-LLH, CD-MKL, and CD-Area for (a) changes in Gaussian high-dimensional data (1st 9 datasets), (b) changes in density shape from a list of non-Gaussian distributions (DistCh), and (c) changes in non-linear dependent data streams (DEMC, DDC and SWRL). The best results are in bold.

Dataset	*kdq-tree* (AAD) [5]	PCA-SPLL [12]	CD-LLH	CD-MKL	CD-Area
M(10D)	96/1/9/2	63/10/10/26	51/17/4/31	**99/0/0/0**	**99/0/0/0**
M(20D)	75/7/3/17	32/11/1/56	45/14/1/40	96/1/2/2	**99/0/0/0**
M(30D)	59/8/4/32	33/13/2/53	30/25/0/44	97/0/1/2	**97/1/1/1**
D(10D)	93/3/3/3	70/1/1/28	93/1/4/5	**99/0/0/0**	**99/0/0/0**
D(20D)	87/3/7/9	59/0/0/40	93/0/5/6	**99/0/0/0**	**99/0/0/0**
D(30D)	78/4/4/17	56/1/4/42	88/0/9/11	**96/1/1/2**	96/0/0/3
C(10D)	64/7/10/28	67/1/0/31	98/0/1/1	95/0/5/4	**98/1/0/0**
C(20D)	29/12/12/58	49/0/0/50	93/1/2/5	**98/0/2/1**	95/2/2/2
C(30D)	11/4/5/84	48/0/0/51	94/0/4/5	97/0/2/2	**99/0/0/0**
DistCh	99/0/7/0	72/0/44/27	97/1/1/1	**99/0/0/0**	**99/0/0/0**
DEMC	84/6/2/9	24/15/28/60	70/6/3/23	84/6/3/9	**88/0/4/11**
DDC	96/2/3/1	19/18/29/62	61/7/7/31	94/1/3/4	**97/0/0/2**
SWRL	99/0/6/0	23/10/29/66	73/5/0/21	95/0/2/4	**99/0/2/0**

The third set of experiments evaluates the detection accuracy in two special cases: (1) changes in the density shape of data streams, and (2) changes in data streams with non-linear dependencies. Four datasets were generated for evaluation, where each dataset contains 100 batches with batch size of 5×10^4, resulting in a total of 99 changes. $DistCh$ is generated by changing the data distribution in consecutive batches, where a distribution is randomly drawn from a list of preset distributions including: standard Normal, highly skewed Normal, bimodal Normal, trimodal Normal, Gamma and Laplace distributions. In each batch, the mean, variance and correlation are kept constant. The remaining three datasets evaluate the detection accuracy on non-linear dependent data streams. The first dataset includes a Disc with EMpty Circle in the middle ($DEMC$). Changes in consecutive batches are introduced by randomly altering the radius of the empty circle without affecting the mean, variance, or correlation values. The second dataset includes Disc with Dense Circle in the middle (DDC). Non-linear changes in the distribution are introduced by altering the radius of the dense circle in the center of the data points. The third dataset is a *Swiss roll*

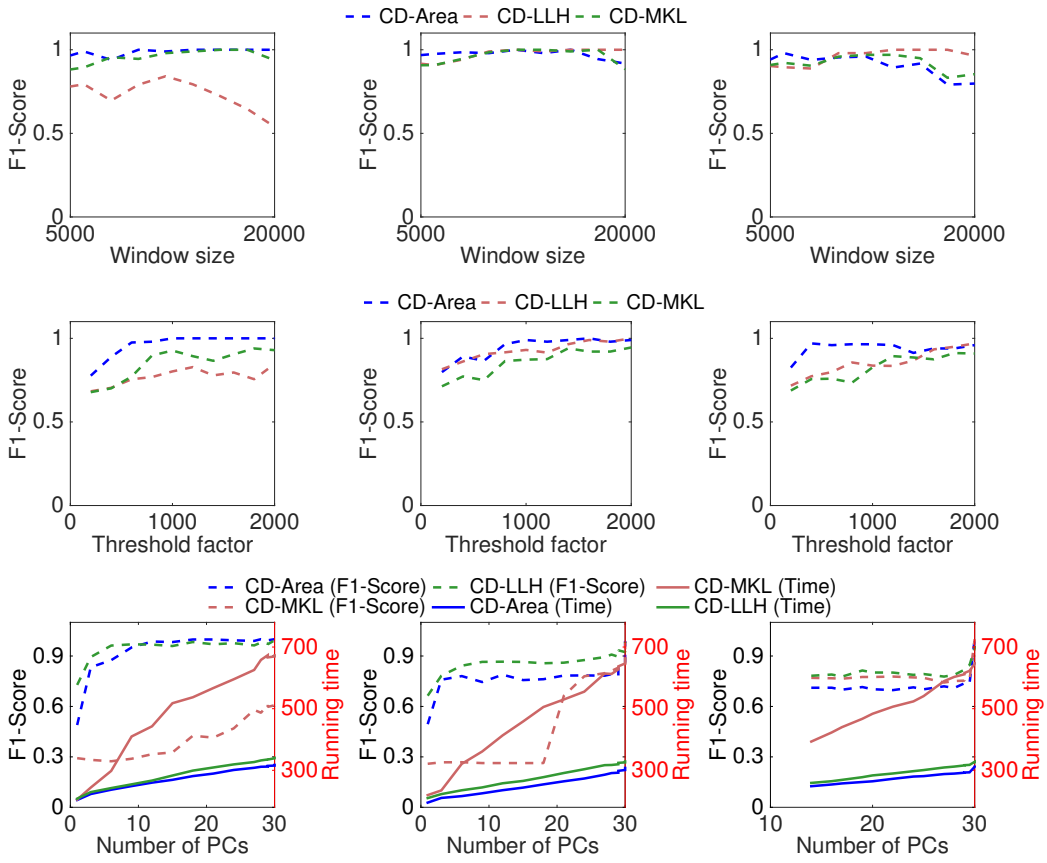

Figure 1: The effects of the parameters' settings on the detection accuracy of our method. The evaluated parameters are the window size (at the first line), the factor that is multiplied by the average score to set the threshold (at the second line), and the percentage of the captured data variance (at the third line). The datasets used for evaluation are M(0.05) dataset (left column), D(0.05) dataset (middle column), and C(0.2) dataset (right column), in 2D (the first and second lines) and 30D (the third line).

(SWRL) dataset with changes designed by altering the distance between any two consecutive contours of the *Swiss roll*.

The results presented in Table 3, show that both CD-MKL and CD-Area outperform other methods in $DisCh$, and that CD-Area obtained better results in all non-linear dependent data streams. While the experimental results are promising, the proposed method is restricted by PCA's limitation to handling only linearly dependent data streams. Change detection in non-linear dependent data streams will be part of our future study.

5.2 Sensitivity to the Parameters' Settings

In this experiment, we study the effects of the most important parameters on the performance of the change detection framework. These parameters are 1) the window size; 2) the factor ξ that is multiplied by the average score to set the threshold value θ_t; and 3) the number of used PCs, which are selected by considering the percentage of the data variance they captured (explained).

Figure 1 shows the detection accuracy of the framework measured by

$$\text{F1-Score} = \frac{2 * precision * recall}{precision + recall}.$$

In Figure 1, the first row of subfigures shows the sensitivity of window size; CD-Area and CD-MKL are less sensitive to the window size setting than CD-LLH. The detection of change in correlation is more sensitive to the setting of window size (third column in the first row), while detection of mean shift and variance change are largely unaffected by different window sizes. Sensitivity of the threshold factor is shown in the second row. Generally, a small threshold factor is not preferred because it causes more false positives. When it increases to moderate values, the good performance of our approaches stays stable for a wide range of values.

The subfigures in the last row of Figure 1 evaluate the sensitivity of the framework w.r.t. the percentage of explained data variance $\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j} * 100\%$ by k PCs. The y-axis on the left represents the F1-Score and the y-axis on the right represents the running time in seconds. The x-axis represents the number of considered PCs. Note that unlike the previous two parameter sensitivity analysis, this evaluation highly depends on evaluation data. The datasets used, M(0.05), D(0.05), and C(0.2) in 30D, have a typical L-shape distribution of λ. The subfigures show that it is possible to detect most of the changes accurately when selecting only the first ten PCs that capture more than 60% variance, and the detection accuracy stays stable when more PCs are selected, except CD-LLH on the datasets with mean shift and variance change. As expected, the running time increases linearly when additional PCs are selected.

5.3 Experiments on Real Data

We also evaluated our method with the *kdq-tree* and PCA-SPLL methods on ten real datasets obtained from machine learning repositories. The information of each dataset is presented in Table 4. Evaluation required that datasets be large enough and have change points where the distribution before it differs from that after it. To increase the data size while maintaining the distributional characteristics of the data, we follow the technique in [17]. The general

Table 4: The sources of the ten real datasets used in the evaluation.

Dataset	Description
D1: Jogging (3D)	This dataset and the next dataset contain reading for physical activities in a gym recorded by smart phones' applications [13]. The dataset contains $342,177$ records, each of which has three features representing the acceleration on the x, y and z axes.
D2: Walking (3D)	Refer to *Jogging* (3D). It includes $424,400$ records.
D3: El Nino (5D)	The dataset is obtained from the UCI KDD archive and contains, after removing the records with missing values, $93,935$ data records, which are readings of oceanographic and surface meteorological from a series of buoys placed throughout the equatorial pacific.
D4: Spruce (10D)	This dataset and the next one from the UCI KDD archive describe the forest cover type for 30×30 meter cells. There are $211,840$ records. We use the 10 quantitative features of each record.
D5: Lodgepole Pine (10D)	Refer to *Spruce* (10D). It includes $283,301$ records.
D6: Ascending Stairs (30D)	This dataset and the following four datasets are from UCI machine learning repository. The dataset (PAMAP2) contains more than 3 million records for physical activities from 12 classes. We use readings from three sensors placed on arm, chest, and knee. After removing incorrect readings and invalid features, the dataset contains 30 features. The number of records in Ascending Stairs dataset is 117,216.
D7: Cycling (30D)	Refer to *Ascending Stairs* (30D). The dataset contains 164,600 records.
D8: Descending stairs (30D)	Refer to *Ascending Stairs* (30D). The dataset contains 104,944 records.
D9: Ironing (30D)	Refer to *Ascending Stairs* (30D). The dataset contains 238,690 records.
D10: Vacuum cleaning (30D)	Refer to *Ascending Stairs* (30D). The dataset contains 175,353 records.

Table 5: Evaluation results of the *kdq-tree* method with the AAD metric, Kuchneva's method (PCA-SPLL), CD-LLH, CD-MKL, and CD-Area for real world data. The values in bold face are the best performance results. The best results are in bold.

Change	Method	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	$D7$	$D8$	$D9$	$D10$
G1D	*kdq-tree*	0.674	0.674	0.957	0.676	0.674	0.585	0.614	0.563	0.599	0.473
	PCA-SPLL	0.423	0.375	0.458	0.492	0.446	0.504	0.352	0.535	0.559	0.628
	CD-LLH	0.816	0.768	**1.0**	0.913	0.872	0.838	0.814	0.809	**0.862**	0.809
	CD-MKL	0.821	0.851	**1.0**	0.846	0.856	0.785	0.832	**0.828**	0.753	**0.831**
	CD-Area	**0.859**	**0.866**	**1.0**	**0.925**	**0.904**	**0.848**	**0.849**	0.811	0.772	0.822
S1D	*kdq-tree*	0.674	0.674	0.966	0.676	0.674	0.592	0.658	0.746	0.649	0.691
	PCA-SPLL	0.298	0.324	0.337	0.346	0.735	0.772	0.387	0.406	0.615	0.691
	CD-LLH	0.813	0.784	**1.0**	0.778	0.794	**0.995**	**1.0**	**0.989**	**0.995**	**0.985**
	CD-MKL	0.765	0.717	**1.0**	0.772	0.741	0.761	0.805	0.878	0.794	0.782
	CD-Area	**0.833**	**0.825**	**1.0**	**0.884**	**0.839**	0.821	0.833	0.872	0.828	0.814

idea is to draw a sample $\mathbf{x}$ randomly from the dataset and then take five samples $(\mathbf{x_1}, \cdots, \mathbf{x_5})$ from the five nearest neighbors of $\mathbf{x}$ with replacement. A new sample is then generated by taking the average of $\mathbf{x}, \mathbf{x_1}, \cdots, \mathbf{x_5}$. This process is repeated until the dataset reaches the required size.

In order to create change points in a dataset, we used similar techniques to [17]. We change the data distribution every 2×10^4 data samples by sampling a batch from the original data and the distribution of the following batch is changed by applying one of the following techniques: **1) *Gauss-1D (G1D)***: the batch is changed by adding a standard 1D Gaussian random variable to a randomly selected dimension. **2) *Scale-1D (S1D)***: the batch is changed by multiplying the values of a randomly selected dimension by two. We used these two types of changes because they affect only a single dimension of the dataset and they are harder to detect. The results in Table 5 show the performance evaluated by $F1$-Score of precision and recall. The results confirm that our framework outperforms the *kdq-tree* and PCA-SPLL methods in detecting different types of changes. The *kdq-tree* has a large number of false positives, even though we used a more strict threshold than the one recommended by the authors.

5.4 Computational Cost Analysis

The time complexity of our framework and the *kdq-tree* method is linear w.r.t. the size of the data stream. However, the constant in the complexity formula controls the efficiency of the evaluated methods. The cost of our method depends on three main subroutines: 1) The PCA routine for extracting PCs, which is called only when a change is reported. It has a complexity of $O(d^2 \times w \times R_c)$, where d is the data dimensionality, w is the window size, and R_c is the number of reported changes. The frequency of calling this subroutine depends on the nature of the data. 2) The incremental density update by KDE-Track or in histograms, which requires a constant time at the arrival of a new data sample from the stream. 3) Computing the divergence metric, which is done incrementally and costs a constant time upon the arrival of a new data sample. Therefore, the computational cost of our framework is affected more by data dimensionality, as the PCA routine has quadratic complexity w.r.t. dimensionality d, and maintaining the histograms and computing the change-score may have to be done on more PCs when d is higher. The window size has a relatively less effect on the running time of our framework as it affects only the PCA routine.

The running time of the *kdq-tree* method increases fast w.r.t. window size and data dimensionality, mainly due to the expensive bootstrap sampling routine for computing the threshold to report changes. The bootstrap techniques requires $O(kdw \log(\frac{1}{\nu}))$ to set the threshold, where k is the number of bootstrap samples and ν is the minimum cell width. The complexity for the *kdq-tree* construction using w samples is $O(dw \log(\frac{1}{\nu}))$, and the *kdq-tree* update and change-score maintenance requires $O(d \log(\frac{1}{\nu}))$ upon the ar-

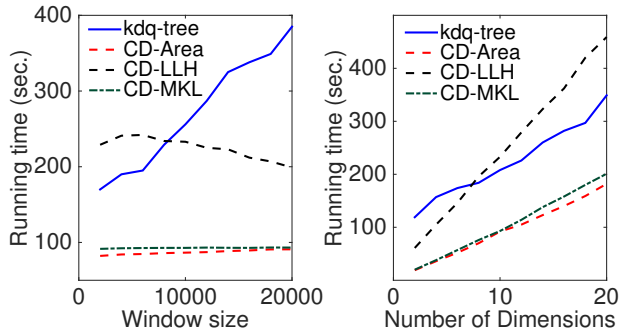

Figure 2: Running time of the evaluated methods for different window sizes (left) and different number of dimensions (right). The stream size is 5×10^6 data samples. Changes occur every 5×10^4 data samples and the number of changes is 99.

rival of a new data sample. Hence, the methods overall complexity is $O(kdw \log(\frac{1}{\nu} R_c))$, where R_c is the number of reported changes.

The results in Figure 2 confirm our analysis. We see a slight increase in the running time of CD-Area and CD-MKL when increasing the window size (Figure 2 Left), while the runtime increases almost linearly w.r.t. data dimensionality (Figure 2 Right). CD-LLH requires extra $O(dw)$ compared to CD-Area and CD-MKL for computing the initial change-score values in Eq. (6) after each change. Thus, it costs more than CD-Area and CD-MKL. The running time of *kdq-tree* increases linearly with the window size and data dimensionality.

Our framework is also superior to *kdq-tree* in processing a real data stream. The *kdq-tree* requires that the data be normalized (into $[0,\ 1]$) before running to ensure that growth of the tree growing is not stopped prematurely by the stopping condition of the minimum cell width. However, it is unrealistic to normalize data streams before hand, as data arrives continuously. Our framework does not require normalization.

6. CONCLUSION

In this paper, we presented a new framework for detecting abrupt changes in multidimensional data streams. The framework is based on projecting data on selected principal components. On each projection, densities in reference and test windows are estimated and compared. Then a change-score value is calculated by one of the divergence metrics. By treating all selected PCs with equal importance, the maximum change-score among different PCs is considered as the final change-score.

The framework uses accurate and efficient density estimators, different divergence metrics to compare the data distributions, and a dynamic threshold setting to report changes. The performance results on both synthetic and real data show that our framework outperforms two baseline methods for detecting changes in unlabeled multidimensional data streams. The results also show that our framework scales well for high-dimensional data.

Our next target is to explore the deployment of change detection in other data mining applications such as online classification and online clustering.

Acknowledgments

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST).

7. REFERENCES

[1] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *SIGMOD*, 2003.

[2] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *SDM*, 2007.

[3] S. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *Intl. J. of Math. Models and Methods in App. Sci.*, 1:300–307, 2007.

[4] X. L. Dai and S. Khorram. Remotely sensed change detection based on artificial neural networks. *Photogrammetric Engineering & Remote Sensing*, 65, 1999.

[5] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Symp. on the Interface of Stat., Computing Science, and Applications*, 2006.

[6] V. Guralnik and J. Srivastava. Event detection from time series data. In *KDD*, 1999.

[7] S. Ho. A martingale framework for concept change detection in time-varying data streams. In *ICML*, 2005.

[8] L. Jin, S. Wang, and H. Wang. A new nonparametric stationarity test of time series in time domain. *Journal of the Royal Statistical Society: Series B*, to appear, 2015.

[9] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *SDM*, 2009.

[10] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *ICCV*, 2007.

[11] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *VLDB*, 2004.

[12] L. I. Kuncheva and W. J. Faithfull. PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE Transactions on Neural Networks and Learning Systems*, 25:69–80, 2014.

[13] J. Kwapisz, G. Weiss, and S. Moore. Activity recognition using cell phone accelerometers. In *Proc. of the 4th Int. Workshop on Knowledge Discovery from Sensor Data*, 2010.

[14] D. Liu, D. Sun, and Z. Qiu. Feature selection for fusion of speaker verification via maximum kullback-leibler distance. In *ICSP*, 2010.

[15] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[16] A. Qahtan, X. Zhang, and S. Wang. Efficient estimation of dynamic density functions with an application to outlier detection. In *CIKM*, 2012.

[17] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *KDD*, 2007.

[18] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Math.*, 60:699–746, 2008.

[19] J. Takeuchi and K. Yamanishi. A unifying framework for detecting outliers and change points from time series. *TKDE*, 18:482–492, 2006.

[20] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8:275–300, 2004.

[21] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag. Data stream clustering with affinity propagation. *TKDE*, 26:1644–1656, 2014.