Linear-time Detection of Non-linear Changes in Massively High Dimensional Time Series

Hoang-Vu Nguyen° Jilles Vreeken°

Abstract

Change detection in multivariate time series has applications in many domains, including health care and network monitoring. A common approach to detect changes is to compare the divergence between the distributions of a reference window and a test window. When the number of dimensions is very large, however, such a naïve approach has both quality and efficiency issues: to ensure robustness the window size needs to be large, which not only leads to missed alarms but also increases runtime.

To this end, we propose LIGHT, a linear-time algorithm for robustly detecting non-linear changes in massively high dimensional time series. Importantly, LIGHT provides high flexibility in choosing the window size, allowing the domain expert to fit the level of details required. To do such, we 1) perform scalable PCA to reduce dimensionality, 2) perform scalable factorisation of the joint distribution, and 3) scalably compute divergences between these lower dimensional distributions. Extensive empirical evaluation on both synthetic and real-world data show that LIGHT outperforms state of the art with up to 100% improvement in both quality and efficiency.

1 Introduction

Change detection in time series is an important task and has applications in many areas, e.g. health care and network monitoring [1,10,17]. In short, it is concerned with detecting time points at which important statistical properties of the time series change. To do so, a common approach is to compare the divergence between data distributions of a reference window and a test window. Traditionally, this works well for univariate time series [3, 10, 29]. Time series nowadays, however, are massively high dimensional, e.g. those from human physical activities have more than 5 000 dimensions [23], while those from online URLs have more than 50 000 dimensions [18]. With high dimensionality, existing work has quality and efficiency issues.

In particular, techniques working directly with distributions of *all* dimensions, such as [5,8,9], are prone to the curse of dimensionality. Naïvely, this issue seems to be resolved by using a large window size. A large window, however, in general increases runtime, causes high delay, and misses alarms [1]. More recent work [11, 22] alleviates the issue by principal component analysis (PCA). Using PCA they map data in reference and test windows to a lower dimensional space on which divergence score is computed per *individual* eigenvectors. While this in theory can avoid high dimensionality, it has three pitfalls. First, the traditional PCA method is cubic in the number of dimensions. While faster methods to compute PCA do exist [6, 16, 24, 27], their effect on change detection has not been investigated. Second, to ensure the stability of covariance matrices that PCA uses, these methods need a large window size [13]. Third, by assuming statistical independence in the PCA space, they may miss complex changes over correlations of dimensions. Thus, detecting changes in massively high dimensional time series still is an open problem.

In this paper, we aim at addressing this issue. We do so by proposing LIGHT, for **li**near-time change detection in hi**gh** dimensional **t**ime series. In short, LIGHT overcomes the drawbacks of existing techniques by a three-step approach. First, we perform scalable PCA mapping of the two windows using matrix sampling [6]. Through matrix sampling, we allow for medium to small window sizes. After PCA mapping, we obtain data embedded in a much lower-dimensional space. To further increase robustness of divergence assessment and flexibility in choosing the window size, we factorise the joint distribution in the transformed space to lower dimensional distributions. Finally, we propose a divergence measure that computes change score using these distributions. Importantly, our measure has the non-negativity property of well-known divergence measures. Further, it allows non-parametric computation on empirical data in closed form. It also permits incremental calculation and hence is suited to the setting of time series (but not data streams; making LIGHT applicable to data streams is future work). Our analysis and extensive experiments show that LIGHT is very scalable and achieves high quality.

The road map is as follow. In Sect. 2, we give an overview of LIGHT. In Sect. 3, we provide its details and analyse its scalability. In Sect. 4, we review related work. In Sect. 5, we report empirical study. We round up with a discussion in Sect. 6 and conclude in Sect. 7. For readability, we postpone all proofs to the Appendix.

2 Overview

Consider an n-dimensional real-valued time series $\{\mathbf{X}(1), \mathbf{X}(2), \ldots, \mathbf{X}(M)\}$. At each time instant $t \in [1, M]$ the sample $\mathbf{X}(t)$ consists of n values $\{X_1(t), \ldots, X_n(t)\}$ where $X_i(t)$ corresponds to dimension X_i ($i \in [1, n]$).

°Max Planck Institute for Informatics and Saarland University, Saarbrücken, Germany. {hnguyen,jilles}@mpi-inf.mpg.de

We assume that $X_i(t) \in [v_i, V_i]$ for all $t \in [1, M]$. Note that when the dimension scales are very different, we can normalise to bring them to comparable domains. Here we use different scales to provide more flexibility.

We give the pseudo-code of LIGHT as Algorithm 1. It works as follows. First, we form a reference window $\mathcal{W}_{ref}$ with size $|\mathcal{W}_{ref}| = m$ (Line 2). Next, we transform $\mathcal{W}_{ref}$ to $\mathcal{W}_{ref}^{trans}$ of lower dimensionality and extract structure $\mathcal{S}_{ref}$ from $\mathcal{W}_{ref}^{trans}$ (Line 3). We will explain the rationale of this step shortly. For each new sample $\mathbf{X}(t)$ of the time series, we form a test window $\mathcal{W}_{test} = \{\mathbf{X}(t-m+1), \ldots, \mathbf{X}(t)\}$ (Lines 4 and 7). Next, we apply to $\mathcal{W}_{test}$ the transformation that we perform on $\mathcal{W}_{ref}$ to obtain $\mathcal{W}_{test}^{trans}$; we also extract structure $\mathcal{S}_{test}$ based on $\mathcal{S}_{ref}$. Both steps are described in Lines 5 and 7. We define the change score as the divergence between $(\mathcal{W}_{ref}^{trans}, \mathcal{S}_{ref})$ and $(\mathcal{W}_{test}^{trans}, \mathcal{S}_{test})$ (Line 8). When a change indeed happens (Line 9), we report it (Line 10) and restart the process. Note that in this work, we do not focus on handling time-dependent information, e.g. auto-correlation. Such information can be captured by patching multiple windows as done in [9, 17].

In LIGHT, we transform $\mathcal{W}_{ref}$ by mapping it to another space $\mathbb{S}$ of $k \ll n$ dimensions. We achieve this with scalable PCA using matrix sampling [6]. The purpose of this step is multi-fold. First, in many domains the the default dimensions may not reflect the true physical dimensions [14]; hence a transformation is done to alleviate this issue. Second, when n is very large we need to choose a large window size m to ensure the robustness of divergence computation. A large m however causes high delay and misses alarms [1]. Hence, dimension reduction is necessary to ensure quality. Third, by reducing the number of dimensions and allowing a smaller window size we are able to boost efficiency.

The usual next step would be to perform divergence assessment on $\mathbb{S}$. Working with k-dimensional joint distributions however may still expose us to the curse of dimensionality [21]. For instance, when $n = 50\,000$ and $k = 100$ it is true that $k \ll n$; yet, to effectively process a joint distribution with $k = 100$ dimensions we may still require a moderate m. To tackle this we propose to factorise the joint distribution of $\mathcal{W}_{ref}^{trans}$ to a combination of selected lower dimensional distributions, preserving non-linear structures, e.g. non-linear correlations among k dimensions of $\mathbb{S}$. The set of lower-dimensional distributions makes up structure $\mathcal{S}_{ref}$ in our algorithm. The change score at time instant t is computed based on these low dimensional distributions. Hence, we achieve three goals at the same time: (1) more flexibility in setting m, (2) maintaining robustness of divergence assessment, and (3) detecting changes in complex non-linear structures. Note that the linear transformation with scalable PCA and the construction of $\mathcal{S}_{ref}$ could be perceived as compression of the data preserving first-order and higher-order dependencies, respectively.

Algorithm 1 LIGHT

1: Set $t_s = 0$
2: Set $\mathcal{W}_{ref} = \{\mathbf{X}(t_s+1), \ldots, \mathbf{X}(t_s+m)\}$
3: Set $[\mathcal{W}_{ref}^{trans}, \mathcal{S}_{ref}] =$ *transformAndExtract*$(\mathcal{W}_{ref})$
4: Set $\mathcal{W}_{test} = \{\mathbf{X}(t_s+m+1), \ldots, \mathbf{X}(t_s+2m)\}$
5: Set $[\mathcal{W}_{test}^{trans}, \mathcal{S}_{test}] =$ *apply*$(\mathcal{W}_{test}, \mathcal{W}_{ref}^{trans}, \mathcal{S}_{ref})$
6: **for** each new sample $\mathbf{X}(t)$ of the time series **do**
7: Update $\mathcal{W}_{test}$, $\mathcal{W}_{test}^{trans}$, and $\mathcal{S}_{test}$ by replacing $\mathbf{X}(t-m)$ by $\mathbf{X}(t)$
8: Set $score = div(\mathcal{W}_{ref}^{trans}, \mathcal{S}_{ref}, \mathcal{W}_{test}^{trans}, \mathcal{S}_{test})$
9: **if** *change*$(score)$ **then**
10: Report a change at time instant t and set $t_s = t$
11: Repeat from step 2
12: **end if**
13: **end for**

For divergence assessment, we propose a new divergence measure that judiciously combines component lower dimensional distributions to produce the change score for the original distributions. Our measure has the important non-negativity property of popular divergence measures. In addition, it allows computation on empirical data in closed form. It also facilitates incremental computation and hence is suitable for time series application. Through the development of our measure, we show that when the dimensions in $\mathbb{S}$ are indeed statistically independent, LIGHT reduces to existing PCA-based change detection methods [11, 22], which focus exclusively on linear relationships. Thus, LIGHT is more general, i.e. it can uncover different types of change, be them linear or non-linear.

Having given an overview of LIGHT, in the next section we provide its details and explain why it is a highly scalable solution for very high dimensional time series.

3 Details of LIGHT

In this section we provide the details of LIGHT, namely the data transformation, the distribution factorisation, the divergence measure, and the change score thresholding.

3.1 Data Transformation Following the previous section, we perform PCA on the reference window $\mathcal{W}_{ref}$ with m points and n dimensions. For simplicity we use A to denote the data of $\mathcal{W}_{ref}$; it has m rows and n columns. W.l.o.g., we assume that A is centered. The usual procedure would be to (1) solve the eigendecomposition problem on A^TA, and (2) use the top eigenvectors corresponding to the top eigenvalues to transform A. This costs $O(mn^2 + n^3)$. When n is very large this complexity is prohibitive for large scale processing. Further, when n is large we need to choose a large window size m; otherwise A^TA becomes very unstable making PCA results unreliable [13].

We overcome this by matrix sampling. In short, matrix

sampling is concerned with sampling rows/columns of matrices to reduce the computational cost of common procedures, e.g. matrix multiplication and SVD. These approximations come with error bounds that hold with high probability. Further, as the sampled submatrix has fewer columns the results are also more stable [6]. Matrix sampling has been recently applied in the context of data mining [15]. Here, we employ the technique in [6] as it allows us to avoid fixing a priori the number of dimensions k to be kept. This method essentially performs approximate singular value decomposition (SVD). Recall that SVD finds matrices U, Σ, and V such that $A = U\Sigma V^T$. Here, $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ contain the eigenvectors of AA^T and A^TA, respectively. $\Sigma = \textbf{diag}(\lambda_1, \ldots, \lambda_d)$ is a diagonal matrix of size $m \times n$ where $d = \min\{m, n\}$. The singular values $\lambda_1 \geq \ldots \geq \lambda_d$ of A are also the non negative square roots of the eigenvalues of both AA^T and A^TA. Finding the exact solution of SVD, like PCA, costs $O(mn^2 + n^3)$. With matrix sampling we obtain high quality approximate solution of SVD in much less time. The details are as follows.

We write the column j of A as $A^{(j)}$ for $j \in [1, n]$. First, we sample with replacement $c \ll n$ columns of A according to their relative variance $\frac{|A^{(j)}|^2}{||A||_F^2}$ where $|A^{(j)}|$ is the squared norm of vector $A^{(j)}$ and $||.||_F$ is the Frobenius norm. This forms a matrix $C \in \mathbb{R}^{m\times c}$. Second, we perform PCA on C^TC to extract its top $k \leq c$ eigenvectors $\{y_1, \ldots, y_k\}$ corresponding to its top eigenvalues $\alpha_1 \geq \ldots \geq \alpha_k$. The value of k is determined by how much variance of C to preserve; usually 90% to 99% suffices [14, 22]. Next, we form matrix $U_k \in \mathbb{R}^{m\times k}$ where each column $U_k^{(i)} = \frac{Cy_i}{\sqrt{\alpha_i}}$, and matrix $\Sigma_k = \textbf{diag}(\sqrt{\alpha_1}, \ldots, \sqrt{\alpha_k})$ of size $k \times k$. Let $V_k \in \mathbb{R}^{n\times k}$ be a matrix whose columns contain the top k eigenvectors of A^TA. According to [6], $A_k \approx U_k\Sigma_kV_k^T$ where A_k is the best rank k approximation of A w.r.t. both $||.||_F$ and $||.||_2$ (spectral norm). By definition, the PCA-transformed data $\mathcal{W}_{ref}^{trans}$ is given by $AV_k = A_kV_k = U_k\Sigma_k$. We also need to compute V_k to transform the test window $\mathcal{W}_{test}$ later. It holds that $V_k = A^T(U_k\Sigma_k) = A^T(AV_k)$. The original algorithm is stochastic. We can make it deterministic by picking c columns of A with largest relative variance; to break ties we use a canonical order.

The cost of approximating top k eigenvectors V_k of A^TA is $O(mc^2 + c^3)$. The cost of computing AV_k is $O(mk^2)$. The cost of computing V_k is $O(mnk)$. So the total complexity is $O(mc^2 + c^3 + mk^2 + mnk)$ with $k \leq c \ll n$, which can be simplified to $O(mc^2 + mnk)$ as we choose $c \leq m$. With reasonable c and small m the accuracy is increased [6], i.e. small window size is favored. The complexity of this sampling algorithm is on par with other sampling methods, such as [15, 24]. While these techniques require us to provide the final dimensionality k, our method permits to adapt k to the current reference window $A = \mathcal{W}_{ref}$. Overall, by matrix sampling we are able to boost efficiency and gain stability [6], as we perform PCA on C^TC where $c \ll n$ instead of A^TA. In other words, besides error bounds that hold with high probability the approximate solutions will tend to be more stable to high dimensionality.

3.2 Distribution Factorisation To further increase robustness of divergence assessment and flexibility in choosing the window size, we factorise joint distribution in the transformed space $\mathbb{S}$ to low dimensional distributions. We accomplish this with graphical modeling [30,31], a powerful approach for this purpose. We denote the k dimensions of $\mathbb{S}$ as $\{Y_1, \ldots, Y_k\}$. Assume that we have a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V} = \{Y_1, \ldots, Y_k\}$; two dimensions Y_i and Y_j not connected by an edge are regarded as conditionally independent given all other dimensions $\mathcal{V} \setminus \{Y_i, Y_j\}$. Given such a graph $\mathcal{G}$, one can factorise the joint distribution $p(Y_1, \ldots, Y_k)$ of $\mathcal{W}_{ref}^{trans}$ by first finding special structures (e.g. cliques or connected components) of $\mathcal{G}$, and then using the distributions of these structures to estimate $p(Y_1, \ldots, Y_k)$. Of course, the more complex the structures the better the estimation. Nevertheless, complex structures tend to contain many dimensions, causing quality and efficiency issues for divergence computation. We achieve a more balanced solution by using the edges of $\mathcal{G}$, which is also a good alternative to factorise $p(Y_1, \ldots, Y_k)$ [30, 31]. Under this model

$$p(Y_1, \ldots, Y_k) = \frac{\prod_{(Y_i,Y_j)\in\mathcal{E}} p(Y_i, Y_j)}{\prod_{Y\in\mathcal{V}} p(Y)^{deg(Y)-1}}$$

where $deg(Y)$ is the degree of Y in $\mathcal{G}$. One can see that the joint distribution is now represented by 1-D and 2-D distributions, further easing the 'pressure' on picking window size m. So far, we assume that $\mathcal{G}$ is available. Now we describe how to obtain it from $\mathcal{W}_{ref}^{trans}$.

Our solution consists of three steps: 1) computing pairwise correlation score of Y_i and Y_j with $i \neq j$ – the higher the score the more correlated, 2) initialising $\mathcal{G}$ containing all dimensions and having an edge between every two dimensions – the weight of each edge is their correlation score, and 3) simplifying $\mathcal{G}$ to one of its maximum spanning tree. Note that the simplified $\mathcal{G}$ has $(k-1)$ edges.

Recall that our goal is to detect non-linear changes. To this end, for the first step we choose the quadratic measure of dependency [26] as it captures non-linear correlations, is non-parametric, and permits computation on empirical data in closed form. In short, under this measure the correlation between Y_i and Y_j is given by

$$corr(Y_i, Y_j) = \int\int \left(P(y_i, y_j) - P(y_i)P(y_j)\right)^2 dy_i dy_j$$

where $P(.)$ denotes cumulative distribution function (cdf). Computing this measure for all dimension pairs takes

$O(m^2k^2)$. To boost efficiency, similar to [19] we apply AMS Sketch [2]. In short, to compute $corr(Y_i, Y_j)$ we precompute the sketches of both Y_i and Y_j by projecting their realisations onto multiple random vectors $u \in \{-1, +1\}^m$. We then use the sketch values to estimate $corr(Y_i, Y_j)$. The estimate is unbiased and its error is bounded [2]. The more random vectors used the better the estimate. We find that using $O(k)$ vectors suffices. The time complexity of computing pairwise correlation scores is thus reduced to $O(mk^2)$ [19]. To make our method deterministic we generate the vectors in advance and reuse whenever needed.

For the third step, as $\mathcal{G}$ initially is dense we employ the method proposed in [7]. It is deterministic and costs $O(k^2)$. The outcome of this step is the set of 1-D and 2-D distributions taken from the maximum spanning tree. These distributions also constitute the structure $\mathcal{S}_{ref}$ of $\mathcal{W}_{ref}^{trans}$.

The overall complexity of distribution factorisation is $O(mk^2)$, i.e. linear in m and independent of n.

3.3 Divergence Computation For each test window $\mathcal{W}_{test}$ we first map it to $\mathbb{S}$ using V_k (see Section 3.1). Then, we use $\mathcal{S}_{ref}$ to extract $\mathcal{S}_{test}$, i.e. the set of 1-D and 2-D distributions of $\mathcal{W}_{test}$ corresponding to those in $\mathcal{S}_{ref}$. Our goal now is to compute the divergence score

$$score = div\left(p(Y_1, \ldots, Y_k) \mid\mid q(Y_1, \ldots, Y_k)\right)$$

where $p(Y_1, \ldots, Y_k)$ and $q(Y_1, \ldots, Y_k)$ are the joint distributions of $\mathcal{W}_{ref}$ and $\mathcal{W}_{test}$, respectively. One important question to address here is: How to do this using two sets of distributions $\mathcal{S}_{ref}$ and $\mathcal{S}_{test}$? We answer this question based on the following observations.

LEMMA 3.1. *Let* $\mathrm{KL}\left(p(.) \mid\mid q(.)\right)$ *be the Kullback-Leibler divergence between* $p(.)$ *and* $q(.)$*. Using* $\mathcal{S}_{ref}$ *and* $\mathcal{S}_{test}$ *we have* $\mathrm{KL}\left(p(Y_1, \ldots, Y_k) \mid\mid q(Y_1, \ldots, Y_k)\right) =$

$$\sum_{(Y_i, Y_j) \in \mathcal{E}} \mathrm{KL}\left(p(Y_i, Y_j) \mid\mid q(Y_i, Y_j)\right) - \sum_{Y \in \mathcal{V}: deg(Y) > 1} (deg(Y) - 1)\mathrm{KL}\left(p(Y) \mid\mid q(Y)\right)$$

Proof. We postpone the proof to the online Appendix.

Lemma 3.1 tells us that *score* w.r.t. KL measure is equal to the sum of divergence scores of 2-D distributions (called 2-D divergence scores) offset by those of 1-D distributions (called 1-D divergence scores). Here, $\mathrm{KL}\left(p(Y_i, Y_j) \mid\mid q(Y_i, Y_j)\right)$ stands for the magnitude of changes in Y_i, Y_j, and their dependency. Thus, the sum of 2-D divergence scores stands for the magnitude of changes in the involved dimensions and their dependencies. The subtraction by 1-D divergence scores is to reduce the impact of dimensions contributing to more than one 2-D score term.

Though KL divergence features a nice computation of *score* based on 1-D and 2-D distributions in $\mathcal{S}_{ref}$ and $\mathcal{S}_{test}$, these distributions need to be estimated, e.g. parametrically or by kernel density estimation [25]. Here, we aim at a purely non-parametric approach to maintain the exploratory nature of LIGHT. Thus, we propose to *generalise* the result in Lemma 3.1 to divergence measures other than KL. In particular, for any measure div we propose to set $score$ to

$$\delta \times \left(\sum_{(Y_i, Y_j) \in \mathcal{E}} div\left(p(Y_i, Y_j) \mid\mid q(Y_i, Y_j)\right) \right) - \sum_{Y \in \mathcal{V}: deg(Y) > 1} (deg(Y) - 1)\, div\left(p(Y) \mid\mid q(Y)\right)$$

where δ is a regularisation factor, which is to guarantee that $score$ is non-negative. With our generalisation we give way to applying other non-parametric instantiations for div, e.g. the one in [20] with empirical computation in closed form, while still upholding the non-negativity property of KL divergence. An additional benefit of δ is that it can make the influence of Y with $deg(Y) > 1$ on $score$ more prominent, i.e. more impact is given to the dimensions correlated to multiple other dimensions.

Before introducing the setting of div we show that PCA-based change detection methods [11, 22] – using the previous two steps of our framework – also generalise from Lemma 3.1, yet in a more restrictive manner. In particular, these methods estimate $score$ by $\sum_{i=1}^{k} div\left(p(Y_i) \mid\mid q(Y_i)\right)$ or $\max_{i \in [1,k]} div\left(p(Y_i) \mid\mid q(Y_i)\right)$; note that the two forms are similar. From Lemma 3.1 we see that if $p(Y_i, Y_j) = p(Y_i)p(Y_j)$ and $q(Y_i, Y_j) = q(Y_i)p(Y_j)$ for $(Y_i, Y_j) \in \mathcal{E}$, then $score$ under KL is equal to $\sum_{i=1}^{k} \mathrm{KL}\left(p(Y_i) \mid\mid q(Y_i)\right)$. Thus, PCA-based methods [11, 22] also generalise from KL divergence; however, they impose two additional assumptions that Y_i and Y_j where $(Y_i, Y_j) \in \mathcal{E}$ are statistically independent under both the data of $\mathcal{W}_{ref}^{trans}$ and $\mathcal{W}_{test}^{trans}$. We in turn do not impose these restrictions and can capture correlation in the PCA space $\mathbb{S}$, i.e. we provide a more general solution.

Choosing div**.** We use the quadratic measure of distribution divergence [20]. It is purely non-parametric and its empirical computation is in closed form. Under this measure

$$div\left(p(Y) \mid\mid q(Y)\right) = \int \left(P(y) - Q(y)\right)^2 dy$$

and $div\left(p(Y_i, Y_j) \mid\mid q(Y_i, Y_j)\right)$ is defined similarly. We set $\delta = 2\sqrt{\sum_{i=1}^{n} \max\{v_i^2, V_i^2\}}$ following our below lemma.

LEMMA 3.2. *Setting* div *to the measure in [20] and* δ *as above, it holds that* $score \geq 0$ *with equality iff* $p(Y_1, \ldots, Y_k)$ *and* $q(Y_1, \ldots, Y_k)$ *under the factorisation model (cf., Section 3.2) are equal.*

Proof. We postpone the proof to the online Appendix.

Complexity analysis. The cost of computing $score$ for initial $\mathcal{W}_{ref}$ and $\mathcal{W}_{test}$ or after every change is $O(m^2k)$. The cost of computing divergence score for each new sample of the time series is $O(mk)$. In case a large window size m is required, e.g. due to the application scenario, we can further boost the efficiency of our method by sampling with replacement the data of $\mathcal{W}_{ref}^{trans}$ and $\mathcal{W}_{test}^{trans}$.

3.4 Change Score Thresholding We perform an adaptive thresholding scheme to decide when a change indeed happens. The scheme we consider is the Page-Hinkley test, which has been used in [22]. Under this test we keep track of the past change scores corresponding to time instants without change. For a new time instant, we assess how much its change score deviates from these historical scores and raise the flag when the deviation is large (w.r.t. an easy-to-adapt threshold). More details are in [22].

3.5 Summing Up On a time series with r changes, LIGHT costs $O\left((mc^2 + mnk + m^2k)r + (M-r)mk\right)$. In our experiment $m = O(n)$, which simplifies the complexity to $O\left((c^2 + nk)mr + (M-r)mk\right)$ time.

4 Related Work

In this work, we focus on change detection time series. For comprehensiveness, we however review related work on both data streams and time series. Being an important task of data mining, change detection on time series and data streams has been studied extensively and there have been many related methods proposed in the literature [1, 3–5, 8–12, 17, 22, 28, 29]. In this work, we target change detection in high dimensional time series, thus in the following we discuss [1, 4, 5, 8, 9, 11, 17, 22, 28] in more details.

PCA-based change detection is studied in [11, 22]. Kuncheva and Faithfull [11] propose to use eigenvectors with small eigenvalues for transformation. Qahtan et al. [22] in turn show theoretically and empirically that eigenvectors corresponding to large eigenvalues instead are more relevant. Both methods apply *traditional* PCA on original reference windows and have cubic runtime in the number of dimensions n. Further, they may miss complex changes due to the assumption that dimensions in the transformed space are independent, i.e. dimensions in the original space have linear correlations only.

Change detection based on estimating the ratio between distributions of reference and test windows is introduced in [9, 17]. The main idea is to directly approximate this ratio by kernel models *without* approximating the two distributions. This procedure implicitly assumes that data is uniformly distributed in the n-dimensional space. When n is large, we need a large window size m to fill in this space to uphold this assumption. Computing the distribution ratio analytically costs $O(m^3)$. For efficiency purposes the window size m must hence be kept small.

Song et al. [28] propose a divergence test based on kernel density estimation for change detection. By performing density estimation on the original n-dimensional space, this test is susceptible to the curse of dimensionality [25]. While one could apply this test after distribution factorisation, proving the non-negativity of the resulting score is non-trivial. Other change detection techniques on multivariate time series [1, 4, 5, 8] also work on the n-dimensional space, and hence, are also prone to the curse of dimensionality.

Our method in contrast alleviates the high dimensionality issue by scalable PCA mapping using matrix sampling. Furthermore, it does not make any assumption on data distributions nor that dimensions are linearly correlated. Lastly, it permits to set the windows size to match the level of details required by the domain expert.

5 Experiments

In this section, we empirically evaluate LIGHT. In particular, we study its effectiveness in detecting known change points on both synthetic and real-world data sets. We implemented LIGHT in Java, and make our code available for research purposes.[1] All experiments were performed single-threaded on an Intel(R) Core(TM) i7-4600U CPU with 16GB RAM. We report wall-clock running times.

We compare to PIND [22] and SPLL [11], two state of the art methods for PCA-based change detection. Both apply traditional PCA, assuming that dimensions in the PCA space are statistically independent. In addition, we consider RSIF [17], which measures divergence scores by directly approximating density ratio of distributions. For each competitor, we optimise parameter settings according to their respective papers. LIGHT has 4 parameters, namely the number of sampled dimensions c which is used in scalable PCA mapping; the percentage of variance preserved which is used in scalable PCA mapping; the number of sketches s_1 and the number of average sketch values s_2 which are used in distribution factorisation. Note that the last two parameters apply for any method using AMS Sketch [2]. The default setting for the parameters is: $c = 200$, percentage $= 90\%$, $s_1 = 50$, and $s_2 = 3$.

We experiment on both synthetic and real data sets. For the latter, we draw 7 data sets from the UCI Machine Learning Repository: Amazon, EMG Actions 1, EMG Actions 2, Human Activities, Human Postural Transitions, Sport Activities, and Youtube Video Games. All are high dimensional time series. As change points, we use existing class labels – a common practice in change detection [9, 28]. Table 1 summarises the characteristics of these data sets.

[1] http://eda.mmci.uni-saarland.de/light/

Data	M	n
Amazon	30 000	20 000
EMG Actions 1	1 800	3 000
EMG Actions 2	3 600	2 500
Human Activities	10 299	561
Human Postural Transitions	10 929	561
Sport Activities	9 120	5 625
Youtube Video Games	120 000	50 000

Table 1: Characteristics of the real data sets. M is the length of the time series and n is its number of dimensions.

5.1 Synthetic Data Generation Each n-dimensional time series we generate contains 100 segments, each having 2000 time steps. We create the change point between every two consecutive segments by varying either distributions or correlation patterns of some dimensions. This means that each time series has 99 change points. We evaluate change detection techniques based on how well they retrieve these known change points. In line with previous work [11, 22], a change point detected by a method is considered to be a true positive if it is flagged correctly before $2m$ points from the new distribution arrive. As performance metric, we use the F1 measure, which is defined as $\frac{2 \times precision \times recall}{precision + recall}$. Note that the higher the F1 score the better. Below we describe three different ways to create change points.

Gaussian Distributions. In this setting, vector $(X_1, \ldots, X_{2l})$ has multivariate Gaussian distribution ($l = \lfloor \frac{n}{3} \rfloor$). The mean vector and covariance matrix are initialised randomly. We consider three types of change: 1) change in mean vector, 2) change in individual variance terms, and 3) change in covariance terms. When creating segments, we switch among those three types of change in a round robin fashion. Each remaining dimension X_j where $j \in [2l+1, n]$ in turn has its distribution fixed to *Gaussian*$(0, 1)$.

Linear Correlations. We embed linear correlations in dimensions $X_1, \ldots, X_{2l}$ where $l = \lfloor \frac{n}{3} \rfloor$. To model correlation in each segment, we first generate $\mathbf{X}_{l \times 1} = \mathbf{A}_{l \times l} \times \mathbf{Z}_{l \times 1}$ where $Z_i \sim$ *Gaussian*$(0, 1)$ and $\mathbf{A}_{l \times l}$ is fixed with a_{ij} initially drawn from *Uniform*$[0, 1]$. Here, $\mathbf{X}_{l \times 1}$ and $\mathbf{Z}_{l \times 1}$ are two sets, each containing l dimensions. We let $\{X_1, \ldots, X_l\} = \mathbf{X}_{l \times 1}$. Next, we generate $\mathbf{W}_{l \times 1} = \mathbf{B}_{l \times l} \times \mathbf{X}_{l \times 1}$ where $\mathbf{B}_{l \times l}$ is fixed with b_{ij} initially drawn from *Uniform*$[0, 0.5]$. Then, using a function f we generate $X_{i+l} = f(W_i) + e_i$ where $i \in [1, l]$, and $e_i \sim$ *Gaussian*$(0, \sigma)$; we fix $\sigma = 0.01$. We use two linear instantiations of f:

$$f_1(x) = 2x + 1, \qquad f_2(x) = \frac{x}{3} - 4.$$

When creating time series segments, we alternatively pick f_1 and f_2. In this way, in every two consecutive segments the correlations between $\{X_1, \ldots, X_l\}$ and $\{X_{l+1}, \ldots, X_{2l}\}$ are different. Each dimension X_j where $j \in [2l+1, n]$ is drawn from *Gaussian*$(0, 1)$.

Non-linear Correlations. To embed non-linear correlations among dimensions $X_1, \ldots, X_{2l}$ where $l = \lfloor \frac{n}{3} \rfloor$, we follow the procedure as in linear correlations except for that we here use four non-linear and complex instantiations of f:

$$f_3(x) = x^2 - 2x\ , \qquad f_4(x) = x^3 + 3x + 1\ ,$$
$$f_5(x) = \log(|x| + 1)\ , \qquad f_6(x) = \sin(2x)\ .$$

When creating time series segments, we switch among f_3, f_4, f_5, and f_6 in a round robin fashion. Each dimension X_j where $j \in [2l+1, n]$ is drawn from *Gaussian*$(0, 1)$.

5.2 Quantitative Results on Synthetic Data We assess quality of the methods tested under different values of dimensionality n and window sizes m. For quality against m we fix $n = 2000$. For quality against n we fix $m = 500$. The results are in Figures 1(a)—1(f).

We find that LIGHT consistently achieves the best performance across different values of n and m, and different types of change – from linear to highly non-linear. Its quality improvement over PIND and SPLL is up to 100%.

PIND and SPLL in turn do not perform well, most likely because they use unstable covariance matrices for PCA transformation (note that $n > m$ in all cases tested). At $n = 8000$, we have to stop the execution of PIND and SPLL due to excessive runtime (more than 12 hours).

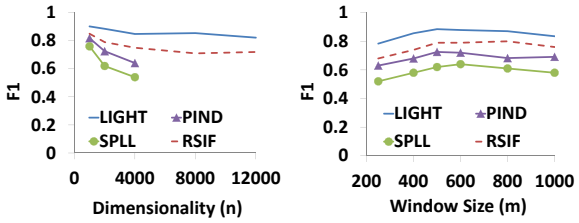
RSIF does not perform so well (especially on non-linear correlations), most likely as it assumes that data is uniformly distributed in high dimensional spaces.

LIGHT in contrast reliably detects changes in high dimensional time series with small window sizes. By not making assumptions on data distributions nor that dimensions are linearly correlated, it copes well with different types of change and yields high quality results.

5.3 Efficiency Results using Synthetic Data Here we test efficiency against window size m and dimensionality n. The setup is as above. We show representative results on data sets with *non-linear correlations* in Figures 2(a) and 2(b). Results on other types of data sets are similar and hence skipped for brevity.

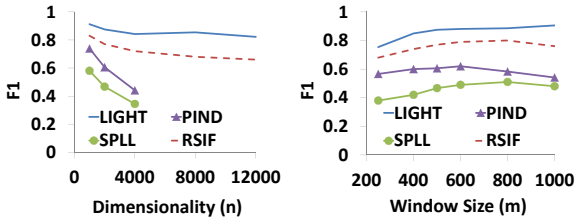
We see that in both cases, LIGHT is more efficient than all competitors. The performance improvement over PIND and SPLL is more than 100% for high values of n. Further, the experiments show that LIGHT has linear scalability to n and m, which corroborates our analysis in Section 3.5.

5.4 Parameter Sensitivity To assess the sensitivity of LIGHT to its parameters, we re-use the synthetic data sets above, fixing $n = 2000$ and $m = 500$. The default setting



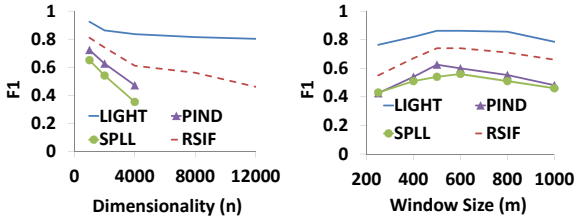
(a) Gaussian distributions: F1 vs. dimensionality

(b) Gaussian distributions: Runtime vs. dimensionality



(c) Linear correlations: F1 vs. dimensionality

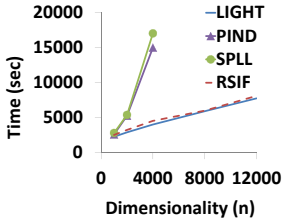
(d) Linear correlations: Runtime vs. dimensionality



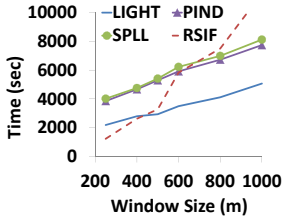
(e) Non-linear correlations: F1 vs. dimensionality

(f) Non-linear correlations: Runtime vs. dimensionality

Figure 1: [Higher is better] Comparison with competitors: F1 scores on synthetic data sets. Overall, LIGHT yields the best quality across different types of change, values of dimensionality n, and window sizes m.

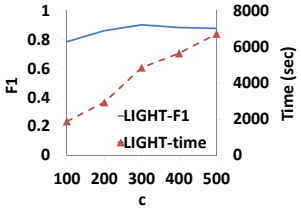


(a) Runtime vs. dimensionality

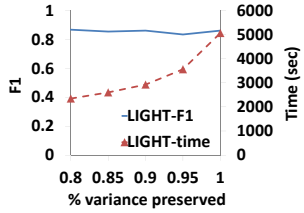


(b) Runtime vs. dimensionality

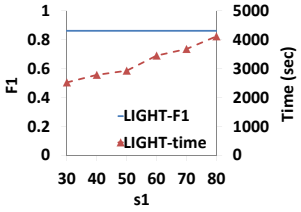
Figure 2: [Lower is better] Comparison with competitors: Runtime on synthetic data sets with non-linear correlations. Overall, LIGHT has the best scalability across different values of dimensionality n and window sizes m.



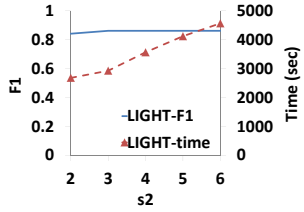
(a) F1 and Runtime vs. c



(b) F1 and Runtime vs. % variance



(c) F1 and Runtime vs. % s_1



(d) F1 and Runtime vs. % s_2

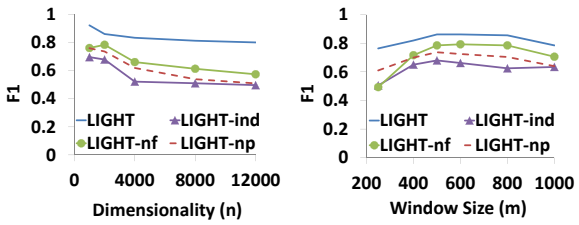
Figure 3: Sensitivity of LIGHT to parameter setting on synthetic data sets with non-linear correlations. Overall, in terms of quality LIGHT is very stable to parameter setting.

for the parameters is: $c = 200$, percentage $= 90\%$, $s_1 = 50$, and $s_2 = 3$. That is, when testing against one parameter we use the default setting for the other parameters.

The representative results on data sets with *non-linear correlations* are in Figures 3(a)—3(d). We see that LIGHT is very stable to different values of its parameters, which facilitates easy parameterisation. The results also suggest that our default setting makes a reasonable choice in terms of both quality and efficiency.

5.5 Ablation Study To study the impact of our design choices, we consider three variants of LIGHT, each of which we create by switching off one of its properties. The first one is LIGHT$_{ind}$, for LIGHT with independence assumption. Essentially, LIGHT$_{ind}$ applies our scalable PCA mapping (see Section 3.1) but assumes that dimensions in PCA spaces are statistically independent. It could be seen as an extension of PIND. The second variant is LIGHT$_{nf}$, for LIGHT without factorisation. That is, LIGHT$_{nf}$ applies our scalable PCA mapping and then computing change score using joint distributions in full PCA spaces. For LIGHT$_{nf}$, we use the quadratic measure of divergence [20] with computation on empirical data in closed form. The third variant is LIGHT$_{np}$, for LIGHT without PCA mapping, i.e. factorisation is performed on original n-dimensional spaces.

We show representative results on data sets with *non-linear correlations* in Figures 4(a) and 4(b). We can see that LIGHT outperforms all of its variants. That LIGHT is better than LIGHT$_{ind}$ highlights the importance of taking into account correlations in PCA spaces. That LIGHT outperforms LIGHT$_{nf}$ shows the effectiveness of our factorisation step. Finally, LIGHT$_{np}$ tries to approximate very high dimensional distributions. This is much harder than approximating lower dimensional ones, as LIGHT does. This explains why LIGHT achieves a better performance than LIGHT$_{np}$. In terms of



(a) F1 vs. dimensionality (b) F1 vs. dimensionality

Figure 4: [Higher is better] Comparison with variants: F1 scores on synthetic data sets with non-linear correlations. Overall, LIGHT outperforms all of its variants.

runtime, LIGHT_{ind} and LIGHT_{nf} are faster than LIGHT since they do not spend time to factorise the joint distribution of each PCA space. The difference however is negligible.

5.6 Results on Real Data We now study the performance of LIGHT on real data. We give an overview of the data sets in Table 1. Each of them is a labeled time series. As change points, we use these class labels, a common practice in change detection literature [9, 28]. For Human Activities (HAR) and Human Postural Transitions (HAPT), as their numbers of dimensions are only in the hundreds, we set $c = 50$ and $m = 100$ for LIGHT. For the other time series, we use its default parameter setting. We evaluate quality using the F1 measure.

The results are in Table 2 and 3. Overall, we can see that LIGHT consistently yields the best quality and is the most efficient across all time series. Notice that EMG1, EMG2, and Sport are relatively small in length while having many dimensions. For an effective change detection on such time series, it is necessary to use small window sizes. This is an issue for PIND and SPLL. In particular, they have to perform PCA transformation on very unstable covariance matrices, which could be an explanation why they do not perform well on EMG1, EMG2, and Sport. For Amazon and Youtube data, the runtime of PIND and SPLL exceeds 12 hours. LIGHT in contrast achieves high accuracy on all high dimensional time series tested. Further, it finishes within 1.5 hours even on the 20 000 dimensional Amazon and 50 000 dimensional Youtube data, and has very high accuracy on both data sets.

6 Discussion

The experiments show that LIGHT is both very efficient and yields high quality for change detection in very high dimensional time series containing different types of change, be them linear or non-linear. Furthermore, it allows small window sizes to be used. This makes it applicable to different types of time series, e.g. those where the dimensionality is even larger than the series length, such as EMG1. Its good performance over the competition suggests the following. First, our scalable PCA transformation is much more effective than traditional PCA mapping when it comes to high dimensionality. The benefits scalable PCA mapping brings here lie in both quality and efficiency. Second, our distribution factorisation yields better quality than using joint distributions or assuming statistical independence in PCA spaces. That LIGHT outperforms competitors and its variants could also be attributed to our new divergence measure, which can capture changes in both linear and non-linear structures.

Yet, there is room for alternative methods as well as further improvements. For instance, besides the matrix sampling method we employ, it also is interesting to explore other related techniques, such as [24]. The details, however, are beyond the scope of this work. In addition, we here pursue the non-parametric setting. As long as the knowledge on data distributions is known, one can resort to parametric methods to compute other divergence measures, e.g. Kullback-Leibler divergence or Jensen-Shannon divergence. As future work, we plan to extend LIGHT to time series with mixed data types, e.g. those with numeric and categorical di-

Data	LIGHT	PIND	SPLL	RSIF
Amazon	**0.91**	-	-	0.64
EMG1	**0.77**	0.48	0.45	0.72
EMG2	**0.84**	0.41	0.44	0.67
HAR	**0.83**	0.62	0.55	0.70
HAPT	**0.85**	0.68	0.62	0.71
Sport	**0.94**	0.51	0.46	0.84
Youtube	**0.93**	-	-	0.76
Average	**0.87**	0.54	0.50	0.72

Table 2: [Higher is better] F1 scores on real data sets. Best values are in **bold**. '-' means excessive runtime (more than 12 hours). Overall, LIGHT consistently yields the best quality across all data sets.

Data	LIGHT	PIND	SPLL	RSIF
Amazon	**1273.6**	∞	∞	1944.5
EMG1	**1.2**	92.6	98.1	3.1
EMG2	**1.1**	345.7	341.5	2.3
HAR	**2.9**	11.2	12.8	3.3
HAPT	**2.4**	12.5	12.4	3.3
Sport	**5.6**	1295.7	1280.4	11.9
Youtube	**4863.5**	∞	∞	7338.4
Average	**878.6**	∞	∞	1329.5

Table 3: [Lower is better] Runtime (in seconds) on real data sets. Best values are in **bold**. '∞' means excessive runtime (more than 12 hours). Overall, LIGHT consistently is the most efficient across all data sets.

mensions. This will help us to enrich the capability of LIGHT in real-world applications.

7 Conclusion

In this paper, we studied the problem of change detection on very high dimensional time series. This setting poses both efficiency and quality issues. To address these, we proposed LIGHT. In short, it works in three steps: 1) scalable PCA mapping to reduce dimensionality, 2) scalable factorisation of joint distributions in PCA spaces to increase robustness, and 3) scalable computation of divergence scores on factorised distributions. Experiments on both synthetic and real-world data show LIGHT outperforms state of the art with up to 100% improvement in both quality and efficiency.

Acknowledgements

The authors are supported by the Cluster of Excellence "Multimodal Computing and Interaction" within the Excellence Initiative of the German Federal Government.

References

[1] C. C. Aggarwal. A framework for change diagnosis of data streams. In *SIGMOD Conference*, pages 575–586, 2003.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[3] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *SDM*, pages 443–448, 2007.

[4] T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi. Change (detection) you can believe in: Finding distributional shifts in data streams. In *IDA*, pages 21–34, 2009.

[5] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE Trans. Signal Process.*, 53(8):2961–2974, 2005.

[6] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1):158–183, 2006.

[7] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[8] Z. Harchaoui, F. R. Bach, and E. Moulines. Kernel change-point analysis. In *NIPS*, pages 609–616, 2008.

[9] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *SDM*, pages 389–400, 2009.

[10] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *VLDB*, pages 180–191, 2004.

[11] L. I. Kuncheva and W. J. Faithfull. PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE Trans. Neural Netw. Learning Syst.*, 25(1):69–80, 2014.

[12] N. Laptev, S. Amizadeh, and I. Flint. Generic and scalable framework for automated time-series anomaly detection. In *KDD*, pages 1939–1947, 2015.

[13] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004.

[14] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, New York, 2007.

[15] E. Liberty. Simple and deterministic matrix sketching. In *KDD*, pages 581–588, 2013.

[16] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2014.

[17] 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.

[18] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *ICML*, pages 681–688, 2009.

[19] H. V. Nguyen, E. Müller, and K. Böhm. 4S: Scalable subspace search scheme overcoming traditional apriori processing. In *BigData Conference*, pages 359–367, 2013.

[20] H. V. Nguyen, E. Müller, J. Vreeken, and K. Böhm. Unsupervised interaction-preserving discretization of multivariate data. *Data Min. Knowl. Discov.*, 28(5-6):1366–1397, 2014.

[21] H. V. Nguyen and J. Vreeken. Non-parametric jensen-shannon divergence. In *ECML/PKDD*, pages 173–189, 2015.

[22] A. A. Qahtan, B. Harbi, S. Wang, and X. Zhang. A PCA-based change detection framework for multidimensional data streams. In *KDD*, pages 935–944, 2015.

[23] J. L. Reyes-Ortiz, L. Oneto, A. Ghio, A. Samá, D. Anguita, and X. Parra. Human activity recognition on smartphones with awareness of basic activities and postural transitions. In *ICANN*, pages 177–184, 2014.

[24] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.

[25] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons Inc, New York, 1992.

[26] S. Seth, M. Rao, I. Park, and J. C. Príncipe. A unified framework for quadratic measures of independence. *IEEE Transactions on Signal Processing*, 59(8):3624–3635, 2011.

[27] A. Sharma and K. K. Paliwal. Fast principal component analysis using fixed-point algorithm. *Pattern Recognition Letters*, 28(10):1151–1155, 2007.

[28] X. Song, M. Wu, C. M. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *KDD*, pages 667–676, 2007.

[29] J. Takeuchi and K. Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE Trans. Knowl. Data Eng.*, 18(4):482–492, 2006.

[30] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[31] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, 1990.

A Proofs

Proof. [Lemma 3.1] By definition we have that

$$\begin{aligned}&\mathrm{KL}\left(p(Y_1,\ldots,Y_k)\;||\;q(Y_1,\ldots,Y_k)\right)\\&=\int p(y_1,\ldots,y_k)\log\frac{p(y_1,\ldots,y_k)}{q(y_1,\ldots,y_k)}dy_1\cdots dy_k.\end{aligned}$$

From Section 3.2 and based on the convention of our framework, we have $p(Y_1,\ldots,Y_k) = \frac{\prod_{(Y_i,Y_j)\in\mathcal{E}} p(Y_i,Y_j)}{\prod_{Y\in\mathcal{V}} p(Y)^{deg(Y)-1}}$ and $q(Y_1,\ldots,Y_k) = \frac{\prod_{(Y_i,Y_j)\in\mathcal{E}} q(Y_i,Y_j)}{\prod_{Y\in\mathcal{V}} q(Y)^{deg(Y)-1}}$. Using these information we deduce that

$$\begin{aligned}&\mathrm{KL}\left(p(Y_1,\ldots,Y_k)\;||\;q(Y_1,\ldots,Y_k)\right)\\&=\sum_{(Y_i,Y_j)\in\mathcal{E}}\int p(y_1,\ldots,y_k)\log\frac{p(y_i,y_j)}{q(y_i,y_j)}dy_1\cdots dy_k\\&-\sum_{Y\in\mathcal{V}:deg(Y)>1}\int p(y_1,\ldots,y_k)\log\frac{p(y)}{q(y)}dy.\end{aligned}$$

Thus, we arrive at the result.

Proof. [Lemma 3.2] First, we prove that

$$\begin{aligned}&div\left(P(Y_i)\;||\;Q(Y_i)\right)\\&\leq(\max(Y_j)-\min(Y_j))\times div\left(P(Y_i,Y_j)\;||\;Q(Y_i,Y_j)\right).\end{aligned}$$

In particular we have that $P(y_i) = \int P(y_i,y_j)dy_j$ and similarly for $Q(y_i)$. Thus,

$$\begin{aligned}&\left(P(y_i)-Q(y_i)\right)^2\\&=\left(\int\left(P(y_i,y_j)-Q(y_i,y_j)\right)dy_j\right)^2\\&\leq(\max(Y_j)-\min(Y_j))\times\int\left(P(y_i,y_j)-Q(y_i,y_j)\right)^2dy_j.\end{aligned}$$

The inequality is in fact the Cauchy-Schwarz inequality. Hence,

$$\begin{aligned}&\frac{1}{(\max(Y_j)-\min(Y_j))}\times div\left(P(Y_i)\;||\;Q(Y_i)\right)\\&\leq\int\left(P(y_i,y_j)-Q(y_i,y_j)\right)^2dy_idy_j.\end{aligned}$$

The right hand side of the above inequality is indeed $div\left(P(Y_i,Y_j)\;||\;Q(Y_i,Y_j)\right)$.

As Y_is are obtained by PCA it holds that

$$Y_i\in\left[-\sqrt{\sum_{i=1}^{n}\max\{v_i^2,V_i^2\}},\sqrt{\sum_{i=1}^{n}\max\{v_i^2,V_i^2\}}\right].$$

We now need to prove that for all $Y\in\mathbf{Y}=\{Y:deg(Y)>1\}$, we can choose for each Y a set of $(deg(Y)-1)$ terms of the form $div\left(P(Y,Y_i)\;||\;Q(Y,Y_i)\right)$ such that $(Y,Y_i)\in\mathcal{E}$ and different Ys will not share any common term. First, as the number of edges of the maximum spanning tree is $(k-1)$, we have that

$$\sum_{Y\in\mathbf{Y}}(deg(Y)-1)=k-2.$$

Now we pick the terms for all $Y\in\mathbf{Y}$ as follows. We consider edges (Y_i,Y_j) whose one end-point is a leaf. When $k>2$ it must be the case that either $Y_i\in\mathbf{Y}$ or $Y_j\in\mathbf{Y}$. Assuming that the former holds we remove edge (Y_i,Y_j) and assign term $div\left(P(Y_i,Y_j)\;||\;Q(Y_i,Y_j)\right)$ to Y_i. We carry on until all edges are removed. Note that under this procedure a node $Y\in\mathbf{Y}$ is not removed from the tree until $deg(Y)$ becomes 1. This also means that when Y is removed there have been $(deg(Y)-1)$ terms assigned to it. This completes the second part of the proof.

With the results in the first part and the second part, we arrive at $score\geq 0$. Equality happens when $P(Y_i,Y_j)=Q(Y_i,Y_j)$ for $(Y_i,Y_j)\in\mathcal{E}$ and $P(Y)=Q(Y)$ for $Y\in\mathbf{Y}$. This means that $p(Y_1,\ldots,Y_k)$ and $q(Y_1,\ldots,Y_k)$ under the factorization model are equal.

When $p(Y_1,\ldots,Y_k)$ and $q(Y_1,\ldots,Y_k)$ are equal, we have that $P(Y_i,Y_j)=Q(Y_i,Y_j)$ for $(Y_i,Y_j)\in\mathcal{E}$ and $P(Y)=Q(Y)$ for $Y\in\mathbf{Y}$. Thus, $score=0$. We complete our proof.

B Incremental Computation of Divergence Score

We illustrate the idea on incrementally computing $div\left(p(E,F)\;||\;q(E,F)\right)$ where $E,F\in\{Y_1,\ldots,Y_k\}$. Incrementally computing $div\left(p(Y)\;||\;q(Y)\right)$ where $Y\in\{Y_1,\ldots,Y_k\}$ follows straightforwardly.

Assume that the empirical data forming $p(E,F)$ contains data points $\{(e_{p,1},f_{p,1}),\ldots,(e_{p,m},f_{p,m})\}$. Analogously, we denote $\{(e_{q,1},f_{q,1}),\ldots,(e_{q,m},f_{q,m})\}$ as the data points forming $q(E,F)$. According to [20], $div\left(p(E,F)\;||\;q(E,F)\right)=$

$$\begin{aligned}&\frac{1}{m^2}\sum_{j_1=1}^{m}\sum_{j_2=1}^{m}\left(V_e-\max(e_{p,j_1},e_{p,j_2})\right)\left(V_f-\max(f_{p,j_1},f_{p,j_2})\right)\\&-\frac{2}{m^2}\sum_{j_1=1}^{m}\sum_{j_2=1}^{m}\left(V_e-\max(e_{p,j_1},e_{q,j_2})\right)\left(V_f-\max(f_{p,j_1},f_{q,j_2})\right)\\&+\frac{1}{m^2}\sum_{j_1=1}^{m}\sum_{j_2=1}^{m}\left(V_e-\max(e_{q,j_1},e_{q,j_2})\right)\left(V_f-\max(f_{q,j_1},f_{q,j_2})\right).\end{aligned}$$

Thus, $div\left(p(E,F)\;||\;q(E,F)\right)$ can be factorized per data point. By storing the contribution of each point to $div\left(p(E,F)\;||\;q(E,F)\right)$, we can incrementally update this score in $O(m)$ time. We have $O(k)$ score terms in total. Thus, the cost to compute divergence score for each new sample of the time series is $O(mk)$.

C Scaling to Large Window Sizes

The idea is that $score$ is computed based on terms of the form $div\ (p(.)\ ||\ q(.))$ where $p(.)$ and $q(.)$ are estimated from the data of $\mathcal{W}_{ref}^{trans}$ and $\mathcal{W}_{test}^{trans}$, respectively. An implicit assumption here is that the data of $\mathcal{W}_{ref}^{trans}$ (similarly for $\mathcal{W}_{test}^{trans}$) are i.i.d. samples of $p(.)$. By definition, i.i.d. samples are obtained by randomly sampling from an infinite population or by randomly sampling with replacement from a finite population. In both cases, the distribution of i.i.d. samples are assumed to be identical to the distribution of the population. This is especially true when the sample size is very large [25]. Thus, when m is very large the empirical distribution $\hat{p}(.)$ formed by $\mathcal{W}_{ref}^{trans}$ approaches the true distribution $p(.)$. Assume now that we randomly draw with replacement $\epsilon \times m$ samples of $\mathcal{W}_{ref}^{trans}$ where $\epsilon \in (0, 1)$. As mentioned above, these subsamples contain i.i.d. samples of $\hat{p}(.) \approx p(.)$. As with any set of i.i.d. samples with a reasonable size, we can assume that the empirical distribution formed by the subsamples is identical to $p(.)$. Thus, we can use them to approximate $score$. In that case, the complexity of computing $score$ for initial $\mathcal{W}_{ref}$ and $\mathcal{W}_{test}$ or after every change is reduced to $O(\epsilon^2 m^2 k)$. When subsampling is needed we only use it once for each $\mathcal{W}_{ref}$.