Generating High-Dimensional Datastreams for Change Detection

Diego Carrera, Giacomo Boracchi

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

Abstract

A popular testbed for change-detection algorithms consists in detecting changes that have been synthetically injected in real-world datastreams. Unfortunately, most of experimental practices in the literature lead to injecting changes whose magnitude is unknown and can not be controlled. As a consequence, results are difficult to interpret, reproduce, and compare with. Most importantly, controlling the change magnitude is a primary requirement to investigate the change-detection performance when data dimension scales, which is an issue to be typically addressed in big data scenarios.

Here we present a best practice to inject changes in multivariate / high-dimensional datastreams: "Controlling Change Magnitude" (CCM) is a rigorous method to generate datastreams affected by a change having a desired magnitude at a known location. In CCM, changes are introduced by directly applying a roto-translation to the data, and the change magnitude is measured by the symmetric Kullback-Leibler divergence between the pre- and post-change data distributions. The roto-translation parameters yielding the desired change magnitude are identified by two iterative algorithms whose convergence is here proven. Our experiments show that CCM can effectively control the change magnitude in real-world datastreams, while traditional experimental practices might not be appropriate for assessing the performance of change-detection algorithms in high-dimensional data.

1. Introduction

Commonly faced problems in datastream mining include determining whether something has changed in the monitored data, or whether two datastreams are different. Moreover, change-detection algorithms [1, 2, 3, 4, 5, 6, 7] are typically employed to pilot adaptation in the post-change condition of the data-generating process [8, 9, 10, 11, 12, 13]. Needless to say, change-detection problems are being addressed also in big data scenarios [14, 15], where datastreams have an increasingly large number of components. In these cases, change-detection algorithms are often combined with pre-processing techniques which extract features [16, 17] or reduce data dimensionality [18], and are then applied to the stream of extracted features.

To provide stable performance measures, experiments need to be performed over a large number of datastreams, which have to be affected by changes at known locations. Unfortunately, there are not many real-world datasets satisfying this requirement, and when the exact change-location is unknown, change-detection performance has to be interpreted by visualizing the analyzed datastream [6]. While visual inspection can indicate whether the algorithm was successful or not, it does not provide a statistically sound assessment, and it is definitively not a viable option when data dimension increases. Therefore, experiments are more conveniently performed on real-world datasets that have been manipulated to contain a change at a known location. This practice allows testing change-detection algorithms both in realistic conditions and on a sufficient number of datastreams to enable stable performance measures. Unfortunately, most of experimental practices are not able to control the magnitude of the injected changes, making results difficult to compare and reproduce on different datastreams. Despite the vast literature on change-detection algorithms [1, 2, 3, 4, 5, 6, 7] and a few frameworks designed for generat-

Email addresses: diego.carrera@polimi.it (Diego Carrera), giacomo.boracchi@polimi.it (Giacomo Boracchi)

ing datastreams containing changes [19, 20, 21], the problem of controlling the magnitude of injected changes has never been addressed before our study.

Here we present "Controlling Change Magnitude" (CCM), a rigorous method to prepare testbeds for change-detection algorithms. CCM has been designed to introduce changes $\phi_0 \to \phi_1$ having a desired magnitude, where ϕ_0 denotes the distribution of stationary (change/anomaly free) data, and ϕ_1 the post-change distribution (or the distribution of anomalies). The change magnitude is conveniently measured by $\mathrm{sKL}(\phi_0, \phi_1)$, namely the symmetric Kullback-Leibler divergence between ϕ_0 and ϕ_1. Changes are introduced by roto-transalating the data. This is a very general change model that encompasses both changes affecting the expected value and the correlation among the data components, as well as the most common experimental practices. These roto-translations are computed through iterative algorithms that are guaranteed to converge to the desired change magnitude. CCM is a flexible method that has been designed assuming that stationary data can be effectively approximated by a Gaussian mixture (GM). This assumption guarantees $\mathrm{sKL}(\phi_0, \phi_1)$ to be finite for any choice of roto-translation parameters, and it is used to prove the convergence of the proposed algorithms.

CCM is a necessary tool to test change-detection algorithms meant for high-dimensional data. In particular, experiments performed to validate the detectability loss [14], namely the fact that the performance of change-detection algorithms monitoring log-likelihood decays when the number of input components scales, would have not been possible without CCM. This and similar studies require special experimental settings (described in [5] and in Section 6.3) where CCM is used to generate, for each considered dimension, multiple datastreams affected by changes that *i*) have the same magnitude and *ii*) occur at the same location. In contrast, other experimental practices result in changes having increasing magnitude when data dimension scales, thus are not suitable to assess change-detection performance at different dimensions.

In more traditional experimental settings, CCM can be used to ease the interpretation and repeatability of experiments, since introducing changes having a known magnitude allows to fairly compare multiple change-detection algorithms on different datasets. In those scenarios where the datastream is conveniently pre-processed by feature-extraction or dimensionality-reduction techniques, CCM can be used to manipulate the output stream, and can provide some guidelines for feature extraction. CCM has been implemented in a MATLAB package[1] that allows to prepare datastreams for change-detection purposes using datasets of popular machine learning repositories like the UCI one [22].

This paper extends [23], which describes CCM and compares it against other experimental practices to inject changes in datastreams. Our extension mainly concerns NP-CCM, a non-parametric version of CCM that allows controlling the change magnitude without having to fit a parametric distribution ϕ_0 to the stationary data. NP-CCM requires a reliable technique to measure the symmetric Kullback-Leibler divergence between two sample populations, and to this purpose we use the method presented in [24]. We also describe CCM with further details that were omitted in [23] due to space limitations, and present a more comprehensive experimental campaign.

The paper is structured as follows. Section 2 overviews the experimental practices commonly used to generate datastreams for testing change-detection algorithms, while Section 3 formally states the addressed problem. CCM is presented in Section 4 together with Theorems 1 and 2 that guarantee its convergence. Proofs are reported in the Appendices. The extension to NP-CCM is illustrated in Section 5, while experiments are shown in Section 6, and the MATLAB framework implementing CCM (including NP-CCM) is described in Section 7. Finally, discussions and conclusions are in Section 8.

2. Background and Related Works

There are two major options for generating datastreams for testing change/anomaly detection algorithms. The first option consists in choosing two different distributions, ϕ_0 and ϕ_1, and use these to generate the pre- and post-change data, respectively (see e.g., [6]). This is probably the easiest way to arbitrarily increase the number of datastreams to be tested, and achieve the desired accuracy in the performance measures. However, synthetically generated data rarely represent realistic

[1]publicly available for download at: `http://home.deib.polimi.it/carrerad`

monitoring scenarios, as they follow quite simplistic distributions.

The second option consists in manipulating real-world datastreams to inject changes at known locations. This is certainly a more appealing strategy, since stationary data remain unaltered, yielding very realistic datastreams where to test change-detection algorithms. Most of the papers in the literature resort to introducing arbitrary shifts, scaling or swapping few components of the original dataset [2, 18]. Changes can be also introduced by rotations and other linear transformations in multivariate data [13, 18], by mixing different datasets [25] or by swapping the classes of labeled data [3, 26]. Unfortunately, none of these approaches control the magnitude of the introduced changes, namely to which extent ϕ_1 differs from ϕ_0.

To the best of authors knowledge, the problem of determining, given ϕ_0, a suitable distribution ϕ_1 such that the change $\phi_0 \to \phi_1$ achieves a desired magnitude, has never been considered in the change-detection literature before our studies. Except [14], all the experimental studies seem to ignore the magnitude of injected changes and none of the existing testbed includes datastreams affected by changes having a desired magnitude. As we will we show in our experiments (Section 6) most of these experimental practices yield changes having a magnitude that increases with data dimension, and prevent a correct assessment of the change-detection performance in high-dimensional data.

The problem of generating datastreams for change-detection purposes has also been addressed in the classification literature, where changes in the data distribution are typically referred to as concept drifts [9]. Here, changes are typically introduced by modifying labels associated to test data [8, 11, 12, 27] and few software frameworks for generating datasets have been developed [19, 20, 21], mainly for online classification purposes. In the concept drift literature, even when changes are injected by manipulating the covariates or raw data as in [13, 28], the impact of the change is only assessed in terms of classification performance, and the distance between ϕ_0 and ϕ_1, i.e. the change magnitude, is ignored. It is also worth mentioning that the research on concept-drift detection and adaptation often considers drifts manifesting in different forms over time, namely: abrupt, gradual, intermittent, transient or recurrent (see the taxonomy in [9]). Studying the change-detection performance in these cases is certainly interesting, even though CCM does not explicitly address the temporal evolution of post-change data. However, these variants can be introduced in our framework by properly modifying the roto-translation parameters computed through CCM.

3. Problem Formulation

Let us consider a dataset $\mathcal{S}$ containing stationary data, which we assume are d-dimensional random vectors $\mathbf{s} \in \mathbb{R}^d$ that are independent and identically distributed (i.i.d.). We assume that $\mathcal{S}$ is generated by ϕ_0, a probability density function (pdf) that most often is unknown. We manipulate $\mathcal{S}$ to generate a datastream $X = \{\mathbf{x}(t),\, t = 1, \dots, \mathrm{T}\}$ affected by an abrupt change at location $t = \tau$ as follows:

$$\mathbf{x}(t) \sim \begin{cases} \phi_0 & t < \tau \\ \phi_1 & t \geq \tau \end{cases}, \text{ being } \phi_1(\mathbf{x}) := \phi_0(Q\mathbf{x}+\mathbf{v})\,, \tag{1}$$

where $\mathbf{x}(t)$ are independent $\forall t$, $\mathbf{v} \in \mathbb{R}^d$ and $Q \in \mathbb{R}^{d\times d}$ is an orthogonal matrix, i.e. $Q^TQ = QQ^T = I$. We denote changes in (1) as $\phi_0 \to \phi_1$.

Our goal is to identify suitable Q and $\mathbf{v}$ such that the change $\phi_0 \to \phi_1$ features a specific magnitude, that we measure by the symmetric Kullback-Leibler divergence (sKL) between ϕ_0 and ϕ_1 (also known as Jeffreys' divergence):

$$\begin{aligned} &\mathrm{sKL}(\phi_0, \phi_1) := \mathrm{KL}(\phi_0, \phi_1) + \mathrm{KL}(\phi_1, \phi_0) \\ &= \int_{\mathbb{R}^d} \log \frac{\phi_0(\mathbf{x})}{\phi_1(\mathbf{x})} \phi_0(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} \log \frac{\phi_1(\mathbf{x})}{\phi_0(\mathbf{x})} \phi_1(\mathbf{x}) d\mathbf{x}\,. \end{aligned} \tag{2}$$

In practice, given ϕ_0 and the desired magnitude κ, our goal is to compute $Q \in \mathbb{R}^{d\times d}$ and $\mathbf{v} \in \mathbb{R}^d$ such that

$$\mathrm{sKL}\left(\phi_0, \phi_1\right) = \mathrm{sKL}\left(\phi_0, \phi_0(Q\cdot+\mathbf{v})\right) = \kappa\,. \tag{3}$$

3.1. Rationale behind our design choice

There are three major choices underlying our problem formulation that need to be discussed: *i)* the i.i.d. assumption for stationary data, *ii)* introducing changes by roto-translation in (1), and *iii)* using the symmetric Kullback-Leibler divergence to measure the change magnitude in (3).

Independence of samples $\mathbf{x}(t)$ generated in stationary conditions is typically assumed in the

Table 1: Notation used in the paper

$\mathcal{S}$	Stationary dataset
X	Datastream affected by a change
T	Number of samples in the datastream X
τ	Location of the change in X
ϕ_0	pdf generating $\mathcal{S}$
ϕ_1	pdf of the post-change data
$\widetilde{\phi}_0$	Gaussian Mixture fitted on $\mathcal{S}$
$\mathrm{KL}(\phi_0, \phi_1)$	Kullback-Leibler divergence between ϕ_0 and ϕ_1
$\mathrm{sKL}(\phi_0, \phi_1)$	Symmetric Kullback-Leibler divergence between ϕ_0 and ϕ_1
$\mathbf{v} = \rho\mathbf{u}$	Vector defining a translation in $\mathbb{R}^d$, whose direction and extent are specified by $\mathbf{u} \in \mathbb{R}^d$ and $\rho \in \mathbb{R}$, respectively
Q	Orthogonal matrix defining a rotation in $\mathbb{R}^d$
P	Orthogonal matrix specifying the planes of rotation of Q
$\boldsymbol{\theta}$	Angles of rotation of Q
$\psi_0,\ \psi_1$	Upper bound of $\log \widetilde{\phi}_0$ and $\log \widetilde{\phi}_1$
$\mathcal{X}_0,\ \mathcal{X}_1$	Sets of samples used to estimate $\mathrm{sKL}(\phi_0, \phi_1)$ in CCM
$\mathcal{S}_0,\ \mathcal{S}_1$	Sets of samples used to estimate $\mathrm{sKL}(\phi_0, \phi_1)$ in NP-CCM
$\mathcal{T},\ \mathcal{V}$	Training and validation sets used in the experiments

change-detection literature [1]. When this assumption is violated, as sometimes it happens in real-world scenario, the datastream needs to undergo a preprocessing phase [5, 6, 29, 30], to extract features that describe the datastream and that meet the assumption in (1). Changes are then detected by monitoring the stream of extracted features. Examples of features typically employed are residuals w.r.t. to an approximation model [16, 29, 31, 32] and indicators computed on representation learned from data [17, 33, 34, 35]. Feature extraction can be also used to reduce the data dimensionality [18].

Choosing roto-translations in our change-model (1) is particularly advantageous first of all because the datastream X can be *assembled* without having to synthetically *generate* any sample from ϕ_1. In fact, stationary data (i.e. $\mathbf{x}(t),\ t < \tau$) can be randomly selected from $\mathcal{S}$, while changed data can be obtained by directly roto-translating other randomly selected samples $\mathbf{s} \in \mathcal{S}$, i.e. $\mathbf{x}(t) = Q^T(\mathbf{s} - \mathbf{v})$

Algorithm 1 CCM: an overview

```
Input: $\mathcal{S}$ the stationary dataset, $\kappa$ the desired change magnitude, $\varepsilon$ the tolerance for the
    change magnitude, N the number of datastreams to generate, T the length of the generated
    datastreams, $\tau$ the change location.
Output: $X_1, \ldots, X_{\mathrm{N}}$, generated datastreams.
 1: Fit a GM $\widetilde{\phi}_0$ to $\mathcal{S}$
 2: for $i = 1, \ldots, \mathrm{N}$ do
      // Estimation of roto-translation parameters
 3:     Run Algorithm 2 to initialize $Q^{(0)}$ and $\mathbf{v}^{(0)}$,
        yielding $\mathrm{sKL}(\phi_0, \phi_0(Q^{(0)} \cdot + \mathbf{v}^{(0)})) > \kappa$
 4:     Run Algorithm 3 to compute $Q$ and $\mathbf{v}$ yielding $|\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q \cdot + \mathbf{v})) - \kappa| < \varepsilon$
      // Generation of the datastream
 5:     for $t = 1, \ldots, \mathrm{T}$ do
 6:         Randomly draw $\mathbf{s}$ from $\mathcal{S}$.
 7:         if $t < \tau$ then
 8:             Set $\mathbf{x}(t) = \mathbf{s}$.
 9:         else
10:             Set $\mathbf{x}(t) = Q^T(\mathbf{s} - \mathbf{v})$.
11:         end if
12:     end for
13:     Set $X_i = \{\mathbf{x}(t),\ t = 1, \ldots, \mathrm{T}\}$.
14: end for
```

for $t \geq \tau$. Second, assuming $\phi_1(\cdot) = \phi_0(Q \cdot + \mathbf{v})$ is quite a general change model, which encompasses changes in the mean as well in the correlation among the data components. As we will show in Section 6, changes obtained by "component swap" or "adding an offset", which are typically encountered in the change-detection literature, correspond to particular roto-tranlations. Third, the matrix Q and the vector $\mathbf{v}$ themselves can be easily parametrized with respect to planes / angles of rotation, and to vector length and direction, respectively.

The symmetric Kullback-Leibler divergence is quite a natural choice in the change-detection scenario, as discussed in [14]. In particular, the Stein's Lemma [36] states that $\mathrm{KL}(\phi_0, \phi_1)$ yields an upper-bound to the power of parametric hypothesis tests that determine whether a given sample population is generated from ϕ_0 (null hypothesis) or ϕ_1 (alternative hypothesis). As a consequence: large values of $\mathrm{sKL}(\phi_0, \phi_1)$ indicate that tests designed to detect either $\phi_0 \to \phi_1$ or $\phi_1 \to \phi_0$ can be very powerful, thus that these changes are very apparent. We observe that (2) is defined (though possibly infinite) if and only if the supports of ϕ_0 and ϕ_1 coincide.

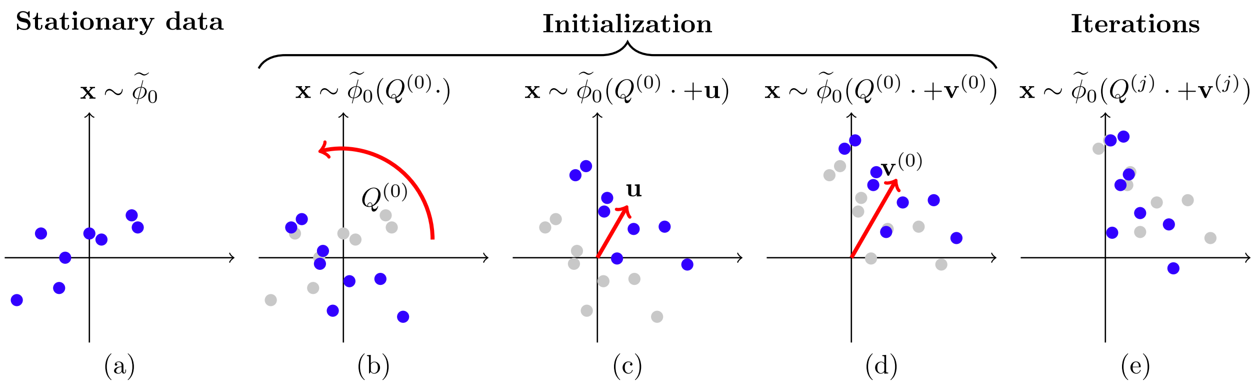

Stationary data
Initialization
Iterations
$\mathbf{x} \sim \widetilde{\phi}_0$
$\mathbf{x} \sim \widetilde{\phi}_0(Q^{(0)}\cdot)$
$Q^{(0)}$
$\mathbf{x} \sim \widetilde{\phi}_0(Q^{(0)}\cdot +\mathbf{u})$
$\mathbf{u}$
$\mathbf{x} \sim \widetilde{\phi}_0(Q^{(0)}\cdot +\mathbf{v}^{(0)})$
$\mathbf{v}^{(0)}$
$\mathbf{x} \sim \widetilde{\phi}_0(Q^{(j)}\cdot +\mathbf{v}^{(j)})$
(a)
(b)
(c)
(d)
(e)

Figure 1: An illustration of the overall CCM procedure, solid blue dots and shaded gray dots indicate the output and the input of each step, respectively. (a) At first, a Gaussian Mixture $\widetilde{\phi}_0$ is fitted on the stationary dataset $\mathcal{S}$. Then, Algorithm 2 is used to initialize CCM by (b) randomly choosing a matrix $Q^{(0)}$ that defines an arbitrary rotation of $\mathbb{R}^d$, (c) computing random translation direction $\mathbf{u}$ and (d) doubling the length of the translation vector $\mathbf{v}^{(0)}$ until $\text{sKL}(\phi_0, \phi_0(Q^{(0)}\cdot +\mathbf{v}^{(0)})) > \kappa$. The value of Q and $\mathbf{v}$ are then adjusted by the bisection method in Algorithm 3 until the corresponding roto-translation yields a change having the desired magnitude.

This is the reason why we assume that ϕ_0 is strictly positive on $\mathbb{R}^d$.

4. The CCM Method

In this section we first provide an overview of CCM and then describe in detail the algorithms and techniques used to generate datastreams affected by changes $\phi_0 \to \phi_1$ having a desired magnitude κ. An important issue in CCM is that to compute $\text{sKL}(\phi_0, \phi_1)$ in (2) we need ϕ_0, which is typically unknown when manipulating real-world datasets $\mathcal{S}$. We solve this issue by replacing ϕ_0 with an empirical estimate, and in particular we adopt a Gaussian mixture (GM) $\widetilde{\phi}_0$ to approximate ϕ_0. Table 1 summarizes the notation adopted for the most relevant elements of CCM.

Figure 1 illustrates the data manipulation performed by CCM, which is overviewed in Algorithm 1. At first, we start by fitting a GM $\widetilde{\phi}_0$ to stationary data $\mathcal{S}$ (line 1 and Section 4.1), which is used for computing our target function $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q\cdot +\mathbf{v}))$. The same GM $\widetilde{\phi}_0$ is used to generate an arbitrary number N of datastreams X as in (1), each corresponding to a different roto-translation. The parametrization adopted to define the roto-translation is described in Section 4.3. For each datastream to be generated we initialize the values of $Q^{(0)}$ and $\mathbf{v}^{(0)}$ by Algorithm 2 (described in Section 4.4). Such initialization is meant to yield a change having a magnitude larger than the target value κ and is illustrated in Figure 1(b-d). Then, the bisection method described in Algorithm 3 and in Section 4.5 is used to iteratively adjust $Q^{(0)}$ and $\mathbf{v}^{(0)}$ to achieve the desired change-magnitude κ with an approximation error smaller than a given $\varepsilon > 0$. Finally, the computed Q and $\mathbf{v}$ are used to generate the datastream X (lines 5-13) by randomly sampling data from $\mathcal{S}$, and transforming those to be placed after the change location τ.

4.1. Fitting Pre-Change Distribution:

The pdf $\widetilde{\phi}_0$ of a GM is a convex combination of k Gaussian functions, which we express as

$$\widetilde{\phi}_0(\mathbf{x}) = \sum_{i=1}^{k} \frac{\lambda_{0,i} \cdot e^{-\frac{1}{2}(\mathbf{x}-\mu_{0,i})^T \Sigma_{0,i}^{-1}(\mathbf{x}-\mu_{0,i})}}{(2\pi)^{d/2}\det(\Sigma_{0,i})^{1/2}}, \qquad (4)$$

where $\sum_{i=1}^{k} \lambda_{0,i} = 1$ and $\lambda_{0,i} \geq 0$, $i \in 1, \ldots, k$ is the weight of the Gaussian having mean $\mu_{0,i}$ and covariance $\Sigma_{0,i}$.

We fit the GM to $\mathcal{S}$ by traditional statistical techniques. In particular, we test different values of k and select the best number of Gaussians by 5-fold cross-validation, by analyzing the distribution of the average likelihood over the test sets. Once having defined k, we estimate the parameters $\{\lambda_{0,i}\}$, $\{\mu_{0,i}\}$ and $\{\Sigma_{0,i}\}$ using the Expectation-Maximization (EM) algorithm [37].

We have adopted GMs since these are rather flexible models that can approximate skewed and multimodal distributions. Most importantly, $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q\cdot +\mathbf{v}))$ is always defined and finite when $\widetilde{\phi}_0$ is a GM, and there exist approximated and fast expressions for computing the log-likelihood w.r.t. a GM, which we describe next.

4.2. *Computing* sKL

When running CCM, we have to repeatedly compute $\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1)$, where both $\widetilde{\phi}_0$ and $\widetilde{\phi}_1(\cdot) = \widetilde{\phi}_0(Q\,\cdot + \mathbf{v})$ are Gaussian mixtures. Since there are no analytical expressions to compute the symmetric Kullback-Leibler divergence between two GMs, we approximate the integrals in (2) via Montecarlo simulations.

More precisely, we synthetically generate two sets $\mathcal{X}_0$ and $\mathcal{X}_1$ containing a sufficient number of i.i.d. realizations drawn from $\widetilde{\phi}_0$ and $\widetilde{\phi}_1$, respectively, and compute the following approximation:

$$\begin{aligned}\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1) \approx \frac{1}{|\mathcal{X}_0|} \sum_{\mathbf{x} \in \mathcal{X}_0} \log \frac{\widetilde{\phi}_0(\mathbf{x})}{\widetilde{\phi}_1(\mathbf{x})} + \\ + \frac{1}{|\mathcal{X}_1|} \sum_{\mathbf{x} \in \mathcal{X}_1} \log \frac{\widetilde{\phi}_1(\mathbf{x})}{\widetilde{\phi}_0(\mathbf{x})},\end{aligned} \tag{5}$$

where $|\mathcal{X}_0|$ and $|\mathcal{X}_1|$ denote the cardinality of $\mathcal{X}_0$ and $\mathcal{X}_1$, respectively.

Since the computation of $\widetilde{\phi}_0(\mathbf{x})$ (resp. of $\widetilde{\phi}_1(\mathbf{x})$) and of its logarithm in (5) might rise severe numerical issues when d is large, we further approximate the likelihood of a GM by considering only the Gaussian of the mixture yielding the largest likelihood, as in [2, 14]. In particular, this yield to the following upper bound of $\log \phi_0(\mathbf{x})$

$$\begin{aligned}\psi_0(\mathbf{x}) = -\frac{1}{2}\Big[\log\left((2\pi)^d \det(\Sigma_{0,i^*})\right) + \\ + (\mathbf{x} - \mu_{0,i^*})^T \Sigma_{0,i^*}^{-1} (\mathbf{x} - \mu_{0,i^*})\Big] + \log(k\lambda_{0,i^*}),\end{aligned} \tag{6}$$

where i^* is defined as

$$i^* = \underset{i=1,\dots,k}{\operatorname{argmax}} \frac{\lambda_{0,i} \cdot e^{-\frac{1}{2}(\mathbf{x}-\mu_{0,i})^T \Sigma_{0,i}^{-1}(\mathbf{x}-\mu_{0,i})}}{(2\pi)^{d/2} \det(\Sigma_{0,i})^{1/2}}. \tag{7}$$

The upperbound of $\log \widetilde{\phi}_1(\mathbf{x})$, namely $\psi_1(\mathbf{x})$, is similarly defined. Then, the Montecarlo estimate of $\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1)$ is obtained by computing

$$\begin{aligned}\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1) \approx \frac{1}{|\mathcal{X}_0|} \sum_{\mathbf{x} \in \mathcal{X}_0} (\psi_0(\mathbf{x}) - \psi_1(\mathbf{x})) + \\ + \frac{1}{|\mathcal{X}_1|} \sum_{\mathbf{x} \in \mathcal{X}_1} (\psi_1(\mathbf{x}) - \psi_0(\mathbf{x})).\end{aligned} \tag{8}$$

4.3. *Parametrization*

To ease calculations, we express Q with respect to its planes and angles of rotation. We store $m := \lfloor d/2 \rfloor$ angles $\theta_1, \dots, \theta_m$ in a vector $\boldsymbol{\theta}$, and define the matrix $T(\boldsymbol{\theta}) \in \mathbb{R}^{d \times d}$ as

$$\begin{aligned} T(\boldsymbol{\theta}) &= \begin{bmatrix} R(\theta_1) & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & R(\theta_m) & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}, \\ R(\theta_i) &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}. \end{aligned} \tag{9}$$

where $R(\theta_i)$ defines a counterclockwise rotation of angle θ_i around the origin in $\mathbb{R}^2$. When d is even, the last column and the last row of $T(\boldsymbol{\theta})$ are dropped. The matrix $T(\boldsymbol{\theta})$ defines a rotation in $\mathbb{R}^d$ whose planes of rotation are generated by the coordinate axes: rotation matrices Q referring to different coordinate axes can be obtained by multiplying $T(\boldsymbol{\theta})$ against an orthogonal matrix $P \in \mathbb{R}^{d \times d}$. Thus, we parametrize the rotation matrix Q as

$$Q(\boldsymbol{\theta}, P) = PT(\boldsymbol{\theta})P^T. \tag{10}$$

The translation vector $\mathbf{v}$ is parameterized by its norm and direction:

$$\mathbf{v}(\rho, \mathbf{u}) = \rho\mathbf{u}, \quad \text{where } \|\mathbf{u}\|_2 = 1,\ \rho = \|\mathbf{v}\|_2. \tag{11}$$

4.4. *Initialization*

CCM initialization is described in Algorithm 2 and it is meant to set $Q^{(0)}$ and $\mathbf{v}^{(0)}$ yielding $\mathrm{sKL}(\phi_0, \phi_0(Q^{(0)} \cdot + \mathbf{v}^{(0)})) > \kappa$. This is a necessary condition for the bisection procedure described in Section 4.5 to work. Given a stationary dataset (Figure 1a), we randomly define the angles of rotation $\boldsymbol{\theta}^{(0)}$ in $[-\pi/2, \pi/2]^m$ (line 2) and randomly select an orthogonal matrix P (lines 3-4). The rotations angles are drawn from a multivariate uniform distribution, and the matrix P is conveniently defined by computing the QR decomposition [38] of a matrix A having random Gaussian entries. The rotation matrix $Q^{(0)}$ is defined according to (10) as $Q^{(0)} = Q^{(0)}(\boldsymbol{\theta}^{(0)}, P) = PT(\boldsymbol{\theta}^{(0)})P^T$ (line 5), being $T(\boldsymbol{\theta})$ as in (9). This is a very general procedure, that can generate any rotation preserving the orientation of the coordinate system. The matrix $Q^{(0)}$ defines the rotation around the origin of the dataset, as show in Figure 1b.

Algorithm 2 CCM: initialization

```
Input: $\widetilde{\phi}_0$ and $\kappa$, the target value of sKL($\widetilde{\phi}_0, \widetilde{\phi}_1$).
Output: Initial roto-translation parameters $\boldsymbol{\theta}^{(0)}$, $P$, $\rho^{(0)}$, $\mathbf{u}$.
 1: Set $\rho^{(0)} = 1$.
 2: Randomly generate $m$ angles $\boldsymbol{\theta}^{(0)}$ in $[-\pi/2, \pi/2]^m$ and a unitary vector $\mathbf{u}$.
 3: Generate a matrix $A \in \mathbb{R}^{d \times d}$ having Gaussian entries.
 4: Set $P$ as the orthogonal matrix of the QR decomposition of $A$.
 5: Set $Q^{(0)}(\boldsymbol{\theta}^{(0)}, P) = PT(\boldsymbol{\theta}^{(0)})P^T$ and $\mathbf{v}(\rho^{(0)}, \mathbf{u}) = \rho^{(0)}\mathbf{u}$.
 6: repeat
 7:     Compute $s^{(0)} = \text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1)$, where $\widetilde{\phi}_1 = \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{v}^{(0)})$.
 8:     $\rho^{(0)} = 2\rho^{(0)}$.
 9: until $s^{(0)} > \kappa$
```

Then, we randomly generate a vector $\mathbf{u}$ having unitary norm (line 2) as described in [39, Algorithm 11]. The vector $\mathbf{u}$ actually defines the translation direction, as illustrated in Figure 1c. If the corresponding $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{u}))$ is larger than κ, the algorithm terminates and returns $Q^{(0)}$ and $\mathbf{v}^{(0)} = \mathbf{u}$. Otherwise, the length of $\mathbf{v}^{(0)}$ is increased by doubling $\rho^{(0)}$ (Figure 1d), which controls the extent of the shift in the data (line 8). This procedure is repeated until the above condition is satisfied. Convergence of Algorithm 2 is guaranteed by the following theorem, which is proved in Appendix A.

Theorem 1. *Let $\widetilde{\phi}_0$ be a Gaussian mixture. Then, for any $\kappa > 0$ and tolerance $\varepsilon > 0$, Algorithm 2 converges in a finite number of iterations.*

4.5. Iterations

The initial values of $Q^{(0)}$ and $\mathbf{v}^{(0)}$ are iteratively adjusted in Algorithm 3 until the corresponding change achieves the target magnitude κ with the desired accuracy ε (Figure 1e). In particular Algorithm 3 implements a bisection procedure that starts from $Q^{(0)} = PT(\boldsymbol{\theta}^{(0)})P^T$ and $\mathbf{v}^{(0)} = \rho^{(0)}\mathbf{u}$, and that iteratively adjusts the rotation angles $\boldsymbol{\theta}$ and the translation extent ρ. The rotation planes defined by P and the translation direction $\mathbf{u}$ are instead kept fixed.

In what follows, we use the subscripts l and u to denote the parameters yielding change magnitudes that are smaller and larger than κ, respectively. Initially, we set both $\boldsymbol{\theta}_l$ and ρ_l to 0 (line 1), while $\boldsymbol{\theta}_u = \boldsymbol{\theta}^{(0)}$ and $\rho_u = \rho^{(0)}$ (line 2). As typical in bisection methods, we set $\boldsymbol{\theta}^{(j)}$ to the average of $\boldsymbol{\theta}_l^{(j)}$ and $\boldsymbol{\theta}_u^{(j)}$ (line 5), and $\rho^{(j)}$ to the average of $\rho_l^{(j)}$ and $\rho_u^{(j)}$ (line 6).

Algorithm 3 CCM: iterations to compute the roto-translation yielding the desired change magnitude

```
Input: $\boldsymbol{\theta}^{(0)}$, $P$, $\rho^{(0)}$, $\mathbf{u}$ from Algorithm 2, $\widetilde{\phi}_0$, $\kappa$, and the tolerance $\varepsilon$.
Output: $Q$ and $\mathbf{v}$ defining the roto-translation yielding $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q \cdot + \mathbf{v})) \approx \kappa$.
 1: Set the lower bounds $\boldsymbol{\theta}_l^{(1)} = 0$, $\rho_l^{(1)} = 0$.
 2: Set the upper bounds $\boldsymbol{\theta}_u^{(1)} = \boldsymbol{\theta}^{(0)}$, $\rho_u^{(1)} = \rho^{(0)}$.
 3: Set $j = 1$.
 4: repeat
 5:     Compute $\boldsymbol{\theta}^{(j)} = (\boldsymbol{\theta}_l^{(j)} + \boldsymbol{\theta}_u^{(j)})/2$, and $Q^{(j)}(\boldsymbol{\theta}^{(j)}, P)$ as in (10).
 6:     Compute $\rho^{(j)} = (\rho_l^{(j)} + \rho_u^{(j)})/2$, and $\mathbf{v}^{(j)}(\rho^{(j)}, \mathbf{u})$ as in (11).
 7:     Compute $s^{(j)} = \text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1^{(j)})$, where $\phi_1^{(j)}(\cdot) = \widetilde{\phi}_0(Q^{(j)} \cdot + \mathbf{v}^{(j)})$.
 8:     if $s^{(j)} < \kappa$ then
 9:         $\boldsymbol{\theta}_l^{(j+1)} = \boldsymbol{\theta}^{(j)}$, $\rho_l^{(j+1)} = \rho^{(j)}$.
10:     else
11:         $\boldsymbol{\theta}_u^{(j+1)} = \boldsymbol{\theta}^{(j)}$, $\rho_u^{(j+1)} = \rho^{(j)}$.
12:     end if
13:     $j = j + 1$.
14: until $\left|s^{(j)} - \kappa\right| < \varepsilon$
```

We denote by $s^{(j)}$ the change magnitude induced by a roto-tranlsation parametrized by $\boldsymbol{\theta}^{(j)}$ and $\rho^{(j)}$ (line 7), which we compute as described in Section 4.2. When $s^{(j)} < \kappa$, we replace both $\boldsymbol{\theta}_l$ and ρ_l with $\boldsymbol{\theta}^{(j)}$ and $\rho^{(j)}$, respectively (line 9), otherwise we similarly update $\boldsymbol{\theta}_u$ and ρ_u (line 11).

These steps are iterated until the sKL achieves the target value κ up to the desired tolerance ε. Convergence of Algorithm 3 is guaranteed by the following theorem, whose proof is given in Appendix A.

Theorem 2. *Let $\widetilde{\phi}_0$ be a Gaussian mixture. Then, for any $\kappa > 0$ and tolerance $\varepsilon > 0$, Algorithm 3 converges in a finite number of iterations.*

4.6. Computational Complexity

The computational complexity of CCM is dominated by the cost of estimating the symmetric Kullback-Leibler divergence, that has to be performed at each iteration of Algorithms 2 and 3. In fact, fitting a GM or performing the QR decomposition (which are invoked in the initial steps of CCM), are not iterated in CCM. Estimating $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1)$ through (8) involves the computation of $\psi_0(\mathbf{x})$ and $\psi_1(\mathbf{x})$, which requires computing the Mahalanobis distance between $\mathbf{x}$ and the mean vector of each Gaussian in $\widetilde{\phi}_0$ for each of the k numerators in (7). Therefore, computing $\psi_0(\mathbf{x})$ and $\psi_1(\mathbf{x})$ requires $O(d^2k)$ operations that have to be computed for each $\mathbf{x} \in \mathcal{X}_0 \cup \mathcal{X}_1$. This yields the computational complexity of (8) to $O(d^2kn)$ for each iteration of Algorithms 2 and 3, being $2n$ the number of samples in $\mathcal{X}_0 \cup \mathcal{X}_1$.

5. Non Parametric CCM (NP-CCM)

The main assumption in CCM is that ϕ_0 can be well approximated by a Gaussian mixture $\widetilde{\phi}_0$. Although this is quite a reasonable assumption, there are cases where, e.g. due to the lack of training data, it is not possible to accurately approximate $\mathcal{S}$ using a GM. However, CCM uses GM only in Algorithms 2 and 3 to estimate the symmetric Kullback-Leibler divergence, and we can relax this technical assumption if an alternative method for computing $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(0)} \cdot +\mathbf{u}))$ is provided. The rationale behind Non Parametric CCM (NP-CCM) is that the parameters Q and $\mathbf{v}$ yielding the desired value of $\text{sKL}(\phi_0, \phi_0(Q \cdot +\mathbf{v}))$ can be directly computed by comparing stationary and transformed data, avoiding fitting any distribution $\widetilde{\phi}_0$.

The problem of estimating the Kullback-Leibler divergence from two populations have been recently investigated in the literature [24, 40, 41, 42], and we adopt the method in [24] that is based on the computation of the k-nearest neighbor (k-nn) distance (as recommended in [24] we use 1-nn distance). In particular $\text{KL}(\phi_0, \phi_1)$ can be estimated as

$$\widehat{\text{KL}}(\mathcal{S}_0, \mathcal{S}_1) = \frac{d}{n_0} \sum_{\mathbf{x} \in \mathcal{S}_0} \log \frac{r_0(\mathbf{x})}{r_1(\mathbf{x})} + \log \frac{n_1}{n_0 - 1}, \quad (12)$$

where $\mathcal{S}_0$ and $\mathcal{S}_1$ are two sets drawn from ϕ_0 and ϕ_1 that contain n_0 and n_1 samples, respectively. In (12), $r_0(\mathbf{x})$ denotes the distance between $\mathbf{x}$ and its nearest neighbor in $\mathcal{S}_0 \backslash \{\mathbf{x}\}$ and similarly $r_1(\mathbf{x})$ in $\mathcal{S}_1$. It was shown [24] that $\widehat{\text{KL}}(\mathcal{S}_0, \mathcal{S}_1)$ converges almost surely to $\text{KL}(\phi_0, \phi_1)$ as $n_0, n_1 \to \infty$. Moreover, we have experienced that a better estimate can be obtained by setting $n_0 < n_1$.

Algorithm 4 NP-CCM: estimation of the symmetric Kullback-Leibler without fitting $\widetilde{\phi}_0$

```
Input: Dataset $\mathcal{S}$ sampled accordingly to $\phi_0$, roto-translation parameters $Q$ and $\mathbf{v}$.
Output: Estimate of $\text{sKL}(\phi_0, \phi_0(Q \cdot +\mathbf{v}))$.
1: Randomly extracts $\mathcal{S}_0 \subset \mathcal{S}$ and $\mathcal{S}_1 \subset \mathcal{S}$ such that $\mathcal{S}_0 \cap \mathcal{S}_1 = \emptyset$ and $|\mathcal{S}_0| = n_0$, $|\mathcal{S}_1| = n_1$.
2: Apply the roto-translation given by $Q$ and $\mathbf{v}$ to each element of $\mathcal{S}_1$, obtaining $\widetilde{\mathcal{S}}_1$.
3: Estimate $\widehat{\text{KL}}(\phi_0, \phi_0(Q \cdot +\mathbf{v})) = \widehat{\text{KL}}(\mathcal{S}_0, \widetilde{\mathcal{S}}_1)$ as in (12).
4: Repeat 1-3 to estimate $\widehat{\text{KL}}(\phi_0(Q \cdot +\mathbf{v}), \phi_0) = \widehat{\text{KL}}(\widetilde{\mathcal{S}}_1, \mathcal{S}_0)$.
5: Set the estimate of $\text{sKL}(\phi_0, \phi_0(Q \cdot +\mathbf{v}))$ to $\widehat{\text{KL}}(\phi_0, \phi_0(Q \cdot +\mathbf{v})) + \widehat{\text{KL}}(\phi_0(Q \cdot +\mathbf{v}), \phi_0)$.
```

Thus, NP-CCM develops as CCM, provided that the change-magnitude is estimated by approximating sKL as in (12) rather than (8). In Algorithm 4 we precisely describe how to estimate $\text{sKL}(\phi_0, \phi_0(Q \cdot +\mathbf{v}))$ from a stationary dataset $\mathcal{S}$, provided Q and $\mathbf{v}$. At first, we randomly extract two subsets $\mathcal{S}_0$ and $\mathcal{S}_1$ from $\mathcal{S}$ (line 1) and we roto-translate samples in $\mathcal{S}_1$ using Q and $\mathbf{v}$ (line 2). These sets are used to estimate the Kullback-Leibler divergence between ϕ_0 and $\phi_0(Q \cdot +\mathbf{v})$ using (12) (line 3). The same procedure is repeated to estimate the Kullback-Leibler divergence between $\phi_0(Q \cdot +\mathbf{v})$ and ϕ_0 (line 4); then $\text{sKL}(\phi_0, \phi_0(Q \cdot +\mathbf{v}))$ is obtained by summing these two estimates (line 5).

Thus, the differences between CCP and NP-CCM are in Algorithm 2 line 7, and Algorithm 3 line 7, where sKL is estimated by using Algorithm 4. Obviously, in NP-CCM there is also no need to estimate the GM $\widetilde{\phi}_0$ in Algorithm 1 line 1. We comment that removing the Gaussian-mixture assumption implies that Theorems 1 and 2 do not hold, thus the convergence of Algorithms 2 and 3 cannot be guaranteed. This has to be taken into account in the implementation of NP-CCM, which should handle cases where the algorithm does not converge. In Appendix A we discuss which assumptions on ϕ_0 would guarantee convergence of NP-CCM algorithms.

The computational complexity of NP-CCM is determined by the cost of estimating sKL through (12). When the 1-nn distances r_0 and r_1 in (12) are computed by straightforward comparisons between all the elements of $\mathcal{S}_0$ and $\mathcal{S}_1$, the cost of estimating sKL is $O(d(n_0 + n_1)^2)$. However, the complexity can be reduced by adopting advanced searching techniques, which rely for instance on the k-d tree structures [43], as we did in our MATLAB implementation.

6. Experiments

Experiments are meant to demonstrate the effectiveness of CCM and the limitations of the experimental practices adopted in the literature (Section 6.1). More precisely, in our experiments we inject changes in both synthetically generated and real world datasets, and then measure the magnitude of the changes introduced by all the different methods.

We first perform experiments on Gaussian datastreams (Section 6.2), where we can exactly measure the symmetric Kullback-Leibler divergence between the pre- and post-change distributions. Then, we introduce changes in two real world datasets (Section 6.3), where we also assess the magnitude of the injected changes by analyzing change-detection performance of hypothesis tests (Section 6.4). Section 6.5 is devoted to the empirical analysis of the convergence of CCM and NP-CCM. To avoid numerical issues, in all our experiments we standardize the components of each datasets by subtracting the sample mean and dividing by the sample standard deviation.

6.1. Considered Methods

We compare CCM and NP-CCM against other methods that used in the change-detection literature to introduce changes in real world datasets [2, 18, 44]:

- **CCM** is configured to identify roto-translation parameters yielding $\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1) = 1$.
- **NP-CCM**, where sKL is estimated using the method described in [24], configured to yield changes of magnitude 1.
- **Offset**: a 1-sigma offset is added to each component of the input data. This change model corresponds to roto-translations where $Q = I_d$ and $\mathbf{v} = [\sigma_1, \dots, \sigma_d]^T$, being σ_i the sample standard deviation of the i-th component.
- **Normalized Offset**: an offset, having magnitude that decreases as d increases, is added to each component of the input data. This change model corresponds to roto-translations having $Q = I_d$ and $\mathbf{v} = [\sigma_1/\sqrt{d}, \dots, \sigma_d\sqrt{d}]^T$. When stationary data follows a Gaussian distribution having independent components, this change model ensures $\mathrm{sKL}(\phi_0, \phi_1) = 1$.
- **Swap**: two randomly chosen components of the input data are swapped. This change model corresponds to a roto-translation where Q is a permutation matrix and $\mathbf{v} = 0$.

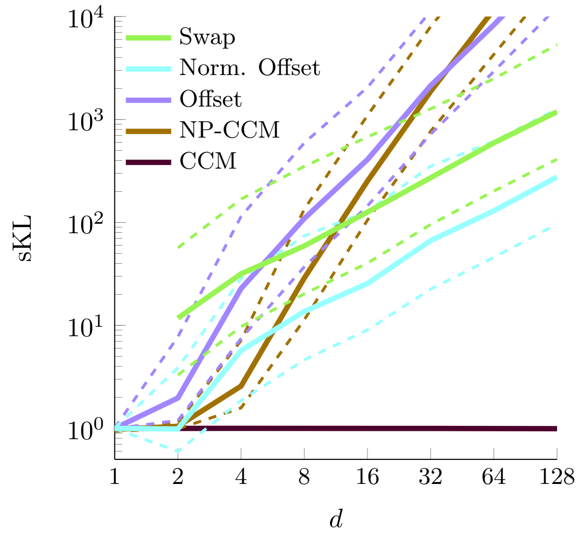
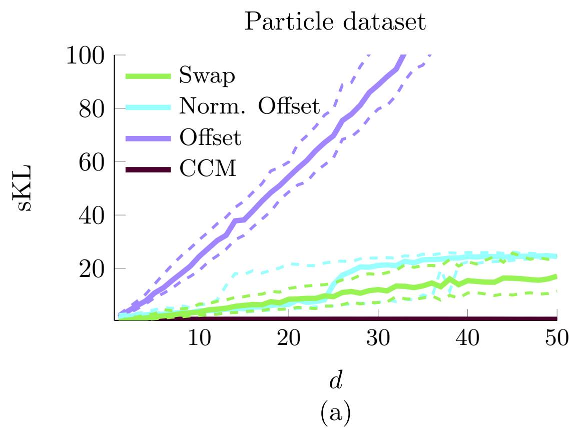


Figure 2: Values of $\mathrm{sKL}(\phi_0, \phi_1)$ obtained on Gaussian datasets using different methods to inject changes at dimension $d \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. Solid lines represent the median values, while dashed lines represent the first and the third quartiles.

While these are the most commonly adopted methods, the literature presents other solutions, such as mixing different datasets [25] or swapping classes of labeled datasets [3, 26]. These methods, however, depend on many degrees of freedom, which would yield a large variance in the results, and as such have not been considered in our analysis.

6.2. Gaussian Datastreams

We consider Gaussian datastreams generated by $\phi_0 = \mathcal{N}(\mu_0, \Sigma_0)$. The post-change distribution $\phi_1(\cdot) = \phi_0(Q \cdot + \mathbf{v})$ is still a Gaussian having mean $\mu_1 = Q^T(\mu_0 - \mathbf{v})$ and covariance matrix $\Sigma_1 = Q^T \Sigma_0 Q$. For each dimension $d \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ we generate 1000 datasets containing 20000 i.i.d. realizations of ϕ_0 each, where the values of μ_0 and Σ_0 have been randomly defined in each dataset. From each dataset, we



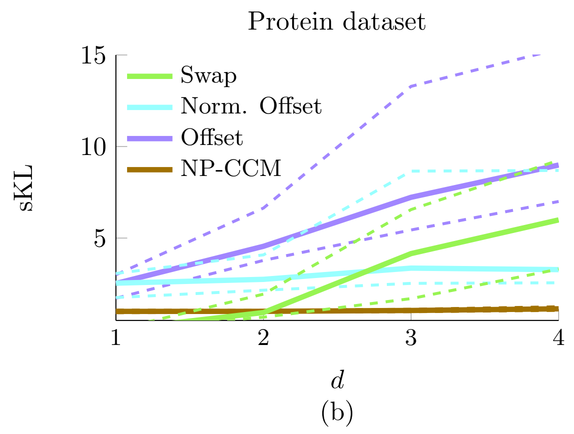


Figure 3: Values of sKL(ϕ_0, ϕ_1) obtained on (a) Particle dataset and (b) Protein dataset using different methods to introduce changes in datastreams having different dimension d. Solid lines represent the median values, while the dashed lines represent the first and the third quartiles.

generate a datastream as in (1) using the all the methods described above to introduce a change at $\tau = 1000$.

On Gaussian datastreams we can exactly compute the magnitude of the introduced changes, since the sKL is given by

$$\mathrm{sKL}(\phi_0, \phi_1) = \frac{1}{2}\Big[\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + \mathrm{tr}(\Sigma_0^{-1}\Sigma_1) + \\ + (\mu_1 - \mu_0)^T(\Sigma_1^{-1} + \Sigma_0^{-1})(\mu_1 - \mu_0) - 2d\Big]. \tag{13}$$

This expression is used as a reference to assess the magnitude of the introduced changes.

Figure 2 shows that CCM generates changes yielding sKL(ϕ_0, ϕ_1) equals to 1 (the target value) for each dimension d. This result is not surprising, since we set $k = 1$ and the assumption that ϕ_0 can be well approximated by a GM is here perfectly met. In contrast, NP-CCM cannot successfully control the change magnitude, which reaches 2 when $d = 4$. This is due to the fact that 20000 samples are not sufficient to obtain an accurate estimate of sKL by (12) when d increases, as noted in [24]. Other experimental practices can not preserve the change magnitude, since sKL steadily increases with d. Remarkably, also the magnitude of normalized offset increases, and this indicates that a coarse approximation of ϕ_0, which in this case ignores the correlation among components, does not allow to control the change magnitude. Figure 2 shows that also the dispersion of the change magnitude increases (note the logarithmic scale in the axis) thus that the introduced changes have considerably different magnitudes when d increases.

Such an increasing magnitude prevents the use of these techniques to correctly study the change-detection performance when data dimension scales. In practical applications, in fact, it is not expected that the change magnitude necessarily increases with d. For instance, when the dimension of the change-detection problem increases because of components that *i)* are not affected by the change, and *ii)* are independent from components that change, the change magnitude does not increase with d.

6.3. Real World Datastreams

In the second experiment we generate datastreams from two real world datasets from the UCI repository [22]. The *MiniBooNE Particle* dataset contains measurements from a physical experiment designed to distinguish electron neutrinos from muon neutrinos. We consider data from the muon class and obtain a dataset having dimension $d_{max} = 50$. We adopt a mixture of $k = 4$ Gaussians to approximate ϕ_0. The second dataset is the *Physicochemical Properties of Protein Tertiary Structure* (Protein) dataset, which has dimension $d_{max} = 9$. By analyzing the marginal distribution over components and pairs of components, we conclude that a GM seems not to properly fit this latter dataset, and we have adopt NP-CCM with a maximum of $d = 4$ components. Larger values of d would have required too many samples for correctly estimating the symmetric Kullback-Leibler divergence by (12).

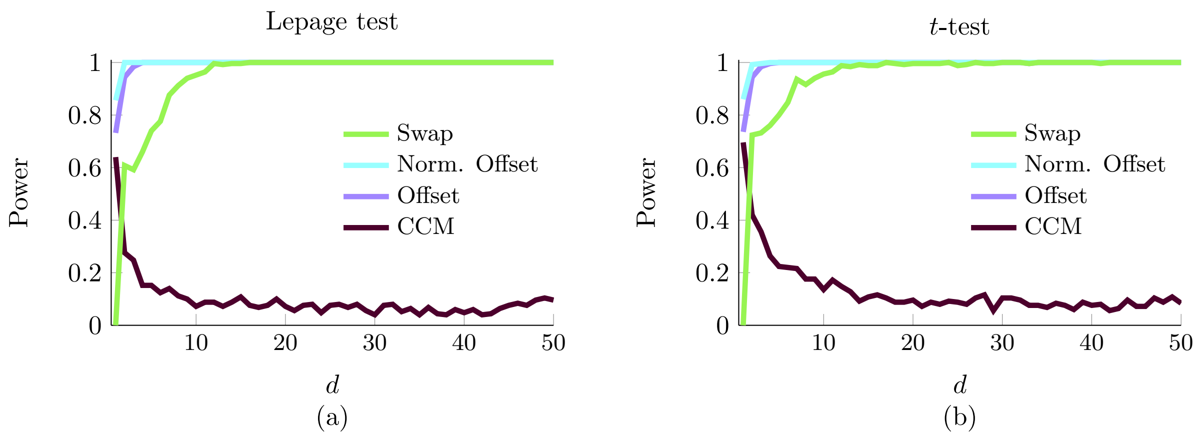


Figure 4: Estimated powers of the Lepage (a) and t-test (b) used to detect changes by comparing the log-likelihood values over W_P and W_R extracted from the Particle dataset. The samples in W_R are affected by different types of changes, and among them only CCM can preserve the change magnitude. In contrast, the other methods inject changes that are more apparent in higher dimensions, as confirmed by the power of both tests, that increases as d increases.

In this experiment we cannot exactly compute the symmetric Kullback-Leibler divergence between the pre- and post- change distributions, since ϕ_0 is unknown. Therefore, we resort to estimating the change magnitude as follows. We split each dataset $\mathcal{S}$ in two set $\mathcal{T}$ (training) and $\mathcal{V}$ (validation): the set $\mathcal{T}$ contains approximately one third of the samples of $\mathcal{S}$ and it is used for estimating the roto-translation parameters by all the methods, while $\mathcal{V}$ is exclusively used for estimate the sKL corresponding to the identified changes. From $\mathcal{T}$ we generate 1000 datastreams for each dimension $d = 1, \ldots, d_{max}$ by randomly choosing d components out of the d_{max} available and compute Q and $\mathbf{v}$ using the considered methods[2]. Then we estimate the corresponding $\text{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q \cdot + \mathbf{v}))$ over $\mathcal{V}$. More precisely, in the experiments on the Particle dataset we fit $\widetilde{\phi}_0$ on $\mathcal{T}$, and we split the set $\mathcal{V}$ in two subsets $\mathcal{X}_0$ and $\mathcal{X}_1$. Then we estimate the symmetric Kullback-Leibler divergence using (8), where the ψ_0 and ψ_1 are defined from $\widetilde{\phi}_0$ (which is fitted on $\mathcal{T}$) and $\widetilde{\phi}_0(Q \cdot + \mathbf{v})$, respectively. In the experiments on the Protein dataset, the symmetric Kullback-Leibler divergence is estimated using Algorithm 4, where $\mathcal{S}_0$ and $\mathcal{S}_1$ in (12) are extracted exclusively from $\mathcal{V}$.

Figures 3 shows the values sKL obtained when increasing d, and demonstrates that CCM and NP-CCM can generate changes having controlled magnitude. The Swap, Offset and Normalized Offset do not preserve the change magnitude, and yield changes that are more apparent when d increases. These results are consistent with those emerging from the experiments on the Gaussian dataset, and further indicate that controlling the change magnitude is very important when manipulating real-world datasets.

6.4. *Change-Detection Performance*

The procedure used in Section 6.3 to assess the performance of all the methods is somehow biased towards CCM and NP-CCM. In fact, as far as CCM is concerned, the performance in term fo the change-magnitude is computed by fitting a GM over the whole dataset, and CCM also fits a GM over the training set $\mathcal{T}$ to determine the change magnitude. Similar concerns hold for NP-CCM, which uses the same expression (12) to estimate sKL but on different datasets. To provide an unbiased assessment, in the following experiments we measure the change-detection performance of a simple algorithm over the generated datastreams, to show that CCM and NP-CCM can better control of the change-magnitude. As a meaningful example of change-detection algorithms, we consider the monitoring of the log-likelihood w.r.t. $\widetilde{\phi}_0$, as a typical approach to perform change detection on multivariate datastreams, see [2, 3, 14].

We use the Particle dataset and for each $d = 1, \ldots, d_{max}$ we generate 1000 datastreams affected by changes introduced from those roto-translations

[2]Note that the swap method does not need training data to define the permutation matrix.

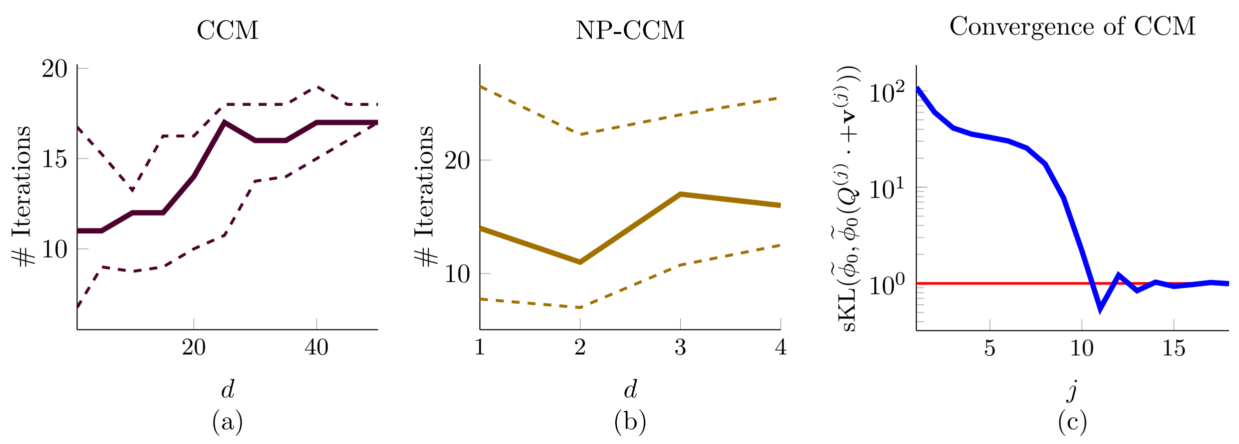


Figure 5: Number of iterations required for the convergence of Algorithm 3 for (a) CCM and (b) NP-CCM, in case of the Particle and the Protein dataset, respectively. For each considered dimension, less the 20 iterations are sufficient to achieve convergence, with a tolerance $\varepsilon = 0.01$, i.e. 1% of the target sKL. The solid lines represent the median computed over 50 realizations, while the dashed lines represents the first and the third quartiles. (c) An examples of the values achieved by the target functional sKL$(\widetilde{\phi}_0, \widetilde{\phi}_1^{(j)})$ in Algorithm 3 of CCM. The horizontal red line indicates the target value $\kappa = 1$ of the symmetric Kullback-Leibler divergence.

computed in Section 6.3. Each sequence contains a training set of $200d$ stationary samples, which is used to estimate a Gaussian mixture $\widehat{\phi}_0$. Test data are divided in two windows W_P and W_R containing 50 pre- and post-change samples, respectively. The log-likelihood, i.e. $\log\left(\widehat{\phi}_0(\cdot)\right)$, computed over W_P and W_R is compared by a Lepage [45] and a t-test. This experimental framework is consistent with the one in [14].

Figure 4 reports the estimated power of both hypothesis tests, when changes are introduced by different methods and for different dimensions d. Changes introduced by CCM yield a decreasing power of the test and this is coherent with the study in [14] which analytically demonstrates the detectability-loss problem when monitoring the log-likelihood of Gaussians and Gaussians Mixtures. In contrast, the power of hypothesis tests used to detect changes introduced by other experimental practices increases with d. This is due to the fact that the magnitude of the introduced changes is actually increasing and indeed prevents to correctly study how the data dimension influences the change-detection performance.

6.5. Execution times

To estimate the execution times of CCM and NP-CCM, we count the number of iterations that are required to CCM and NP-CCM to converge on the Particle and Protein datasets, respectively. These numbers are shown in Figure 5(a) and 5(b) as function of d. We observe that, in general, less than 20 iterations are enough to converge to the target value $\kappa = 1$ with a tolerance of $\varepsilon = 0.01$. An example of how the value of sKL$(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(j)} \cdot + \mathbf{v}^{(j)}))$ changes through all the iterations j is shown in Figure 5(c): in this case, after 18 iterations CCM converges. This translates in execution times that are reported in Figure 6 and show that CCM can be used to generate a large number of datastreams for the evaluation of change-detection algorithms. This times are measured using our MATLAB implementation of CCM and NP-CCM on a PC mounting an Intel Core i5 2.30 GHz CPU and 12 GB RAM. Computational times of the other approaches are not reported even though these are substantially smaller since the Swap and Offset methods do not require any computation, while Normalized Offset method has only to compute the standard deviation of each components over the entire dataset.

7. MATLAB Framework

We have implemented CCM in a MATLAB package that is publicly available for download[3]. This package allows to import any numerical dataset $\mathcal{S}$ and generates an arbitrarily number of datastreams of the same length, containing a change

[3]`home.deib.polimi.it/carrerad`

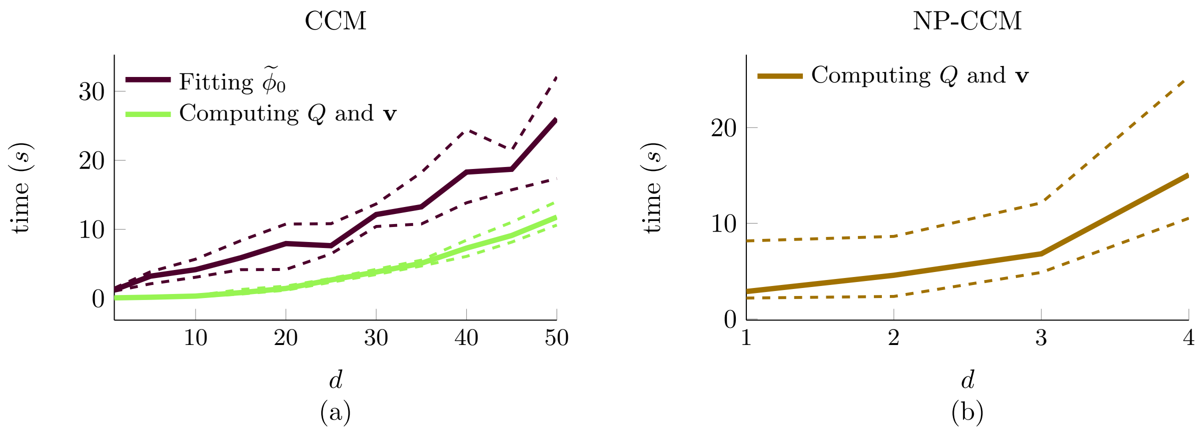


Figure 6: Execution times of (a) CCM and (b) NP-CCM required to compute the roto-translation parameters on the Particle and Protein datasets, respectively. The solid lines represent the median computed over 50 realizations, while the dashed lines represents the first and the third quartiles. CCM requires an additional step to fit the Gaussian Mixture $\widetilde{\phi}_0$. Although this step is more computationally demanding than the computation of the rotation matrix Q and the shift vector $\mathbf{v}$, it has to be performed only once when computing several change parameters for the same dataset.

of a desired magnitude at a given location. The main class of the framework is `CCMframework`, which implements all the functionalities used in Algorithm 1, namely: fitting of the GM $\widetilde{\phi}_0$ to the whole dataset $\mathcal{S}$, computation of the roto-translation, and preparation of the datastreams. The `CCMframework` class has two methods: the constructor and `generateDatastreams`.

The constructor of `CCMframework` takes as input the dataset $\mathcal{S}$ and a flag parameter that specifies which version of the framework to use (CCM or NP-CCM). In case of CCM, the constructor fits a GM to the dataset (Algorithm 1, line 1) for a given number k of Gaussians, which has to be defined by the user, for instance through the procedure described in in Section 4.1. Then an object of the class `gmDistr` is accordingly instantiated. When the flag parameter indicates NP-CCM, the constructor instantiates an object of the class `empiricalDistr`, which relies on an `KDTreeSearcher` MATLAB object to efficiently compute the 1-nn distance in (12). Both of `gmDistr` and `empiricalDistr` implement the method `symmetricKullbackLeibler`, which is used to estimate the symmetric Kullback-Leibler divergence between the pre- and post-change distributions. The `gmDistr` class implements the procedure described in Section 4.2, while the class `empiricalDistr` implements Algorithm 4.

The method `generateDatastreams` implements Algorithm 2 to initialize the roto-transalation parameters, and then runs the bisection method in Algorithm 3 until the matrix Q and the vector $\mathbf{v}$ provide the desired change magnitude or when the number of iterations reaches a maximum value. Finally, the datastream is assembled by randomly selecting samples from $\mathcal{S}$ and by applying the computed roto-translation to samples that are located after the specified change-point location.

8. Conclusions

We present CCM (Controlling Change Magnitude), a rigorous method to introduce changes in real-world datastreams by roto-translating stationary data. We prove that CCM can successfully control the magnitude of the introduced changes when data can be approximated by a Gaussian mixture. Our experiments show that experimental practices commonly adopted in the literature for introducing changes cannot control the change magnitude, which typically increases with d. This fact prevents a fair performance assessment of change-detection algorithms, especially when data become high-dimensional. Our experiments also show that the simplistic approach that aim at preserving the change magnitude approximating ϕ_0 by ignoring correlations among data-component is not successful, and that CCM, which adopts GM as a more general model to describe stationary data, is indeed necessary to control the change magnitude.

CCM is flexible and can be extended using non-parametric techniques to estimate the density of the data or the sKL between the pre- and post-change distributions. However, the presented NP-CCM inherits the limitations of techniques used to measure the sKL, and in particular the need for large amount of data when d increases. Thus, when d is large, it is necessary to resort to CCM and fit parametric models on data.

Ongoing work concerns the extension of CCM to other change models, such as changes in the dispersion of the original data. We will also investigate how to handle categorical data, which would necessarily require a different change model.

Appendix A.

Proofs of Convergence for CCM

To prove the convergence of CCM (Algorithm 1) we demonstrate first that Algorithm 2 terminates after a finite number of iterations, and then that Algorithm 3 actually converges. This is what Theorem 1 and Theorem 2 respectively show.

Theorem 1. *Let $\widetilde{\phi}_0$ be a Gaussian mixture. Then, for any $\kappa > 0$ and tolerance $\varepsilon > 0$, Algorithm 3 converges in a finite number of iterations.*

Proof. It is enough to show that $\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{v})) \to \infty$ as $\|\mathbf{v}\|_2 \to \infty$, or that it admits a lower bound that diverges as $\|\mathbf{v}\|_2 \to \infty$. We here pursue the latter approach and define the lower bound as follows

$$
\begin{aligned}
\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{v})) &\geq \mathrm{KL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{v})) \\
&= \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log \left(\frac{\widetilde{\phi}_0(\mathbf{x})}{\widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v})} \right) d\mathbf{x} \\
&= \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log(\widetilde{\phi}_0(\mathbf{x})) d\mathbf{x} + \\
&\quad - \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log(\widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v})) d\mathbf{x} \\
&\geq \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log(\widetilde{\phi}_0(\mathbf{x})) d\mathbf{x} + \\
&\quad - \log \left(\int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{x} \right).
\end{aligned}
\tag{A.1}
$$

which follows from the fact that the KL is nonnegative and from the Jensen's inequality.

The first term $\int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log(\widetilde{\phi}_0(\mathbf{x})) d\mathbf{x}$ is finite, since the following upper bound[4]

$$
\left| \log(\widetilde{\phi}_0(\mathbf{x})) \right| \leq c_1 + c_2 \|\mathbf{x}\|_2^2, \qquad \forall \mathbf{x} \text{ s.t. } \|\mathbf{x}\|_2 > r. \tag{A.2}
$$

holds for suitable constants $c_1, c_2, r > 0$.

Then, we have to prove that the second term in the last row of (A.1) diverges. To this purpose we define $f(\mathbf{v})$ as

$$
f(\mathbf{v}) := \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{x}, \tag{A.3}
$$

and we observe that f is a continuous positive function. The integrand in (A.3) in fact admits as dominant function

$$
\widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) \leq \widetilde{\phi}_0(\mathbf{x}) \sup_{\mathbf{y} \in \mathbb{R}^d} \widetilde{\phi}_0(\mathbf{y}), \tag{A.4}
$$

which is integrable and independent from $\mathbf{v}$ (see Lemma 16.1 of [46]), since $\sup_{\mathbf{y} \in \mathbb{R}^d} \widetilde{\phi}_0(\mathbf{y})$ is finite and $\widetilde{\phi}_0$ is a Gaussian Mixture. We now demonstrate that $f \in L^1(\mathbb{R}^d)$, as this implies that $f(\mathbf{v}) \to 0$ when $\|\mathbf{v}\|_2 \to \infty$, thus that the lower bound (A.1), which is equal to $-\log f(\mathbf{v})$, diverges. From basic calculus it follows that:

$$
\begin{aligned}
\int_{\mathbb{R}^d} |f(\mathbf{v})| \, d\mathbf{v} &= \int_{\mathbb{R}^d} f(\mathbf{v}) d\mathbf{v} \\
&= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{x} d\mathbf{v} = \\
&= \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \left[\int_{\mathbb{R}^d} \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{v} \right] d\mathbf{x} = 1
\end{aligned}
\tag{A.5}
$$

and this implies $f \in L^1(\mathbb{R}^d)$, which proves the theorem. □

Theorem 2. *Let $\widetilde{\phi}_0$ be a Gaussian mixture. Then, for any $\kappa > 0$, Algorithm 3 converges in a finite number of iterations.*

Proof. The thesis follows from the Intermediate Value Theorem [47] (Theorem 4.23) applied to the function used in the bisection procedure, i.e.

$$
skl(a) := \mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q(\boldsymbol{\theta}(a), P) \cdot + \rho(a)\mathbf{u})), \tag{A.6}
$$

where $\boldsymbol{\theta}(s) = (1-s)\boldsymbol{\theta}_l^{(1)} + s\boldsymbol{\theta}_u^{(1)}$ and $\rho(s) = (1-s)\rho_l^{(1)} + s\rho_u^{(1)}$ are convex combinations of the initialization parameters (defined in Algorithm 3,

[4]This bound is trivial for Gaussian pdfs and follows from basic algebra in case of GM.

lines 1-2). We observe that $skl(0) = 0 < \kappa$ since $\boldsymbol{\theta}_l^{(1)} = 0$ and $\rho_l^{(1)} = 0$, thus the corresponding roto-translation is the identity transformation and $\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0) = 0$. Moreover, $skl(1) > \kappa$, thus it is enough to show that $skl(\cdot)$ is continuous in $[0,1]$ to prove that the bisection method converges (see [48], Theorem 2.1) to a value of $\bar{s}$ such that $skl(\bar{s}) = \kappa$. Thus, $\boldsymbol{\theta}(\bar{s})$ and $\rho(\bar{s})$ yield $\mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_1)$ equal to κ.

To show that $skl(\cdot)$ in (A.6) is continuous, we show that both summands of sKL are continuous, which can be proved by defining

$$kl(\cdot) = \int_{\mathbb{R}^d} g(\cdot, \mathbf{x}) d\mathbf{x} \tag{A.7}$$

where

$$g(a, \mathbf{x}) := \widetilde{\phi}_0(\mathbf{x}) \log \left(\frac{\widetilde{\phi}_0(\mathbf{x})}{\widetilde{\phi}_0(Q(\boldsymbol{\theta}(a), P)\mathbf{x} + \mathbf{v}(\rho(a), \mathbf{u}))} \right). \tag{A.8}$$

The function $g(\cdot,\cdot)$ in (A.7) is continuous in $[0,1] \times \mathbb{R}^d$, thus $g(\cdot, \mathbf{x})$ is continuous in $[0,1]$. Then, to prove that $kl(\cdot)$ is also continuous, we leverage Lemma 16.1 in [46] and prove that $|g(a,\mathbf{x})|$ admits a dominant integrable function that does not depend on s. To this purpose, we exploit (A.2) and, for a sufficiently large r,

$$\begin{aligned} |g(a,\mathbf{x})| &= \widetilde{\phi}_0(\mathbf{x}) \Big| \log\left(\widetilde{\phi}_0(\mathbf{x})\right) + \\ &\quad - \log\left(\widetilde{\phi}_0(Q(\boldsymbol{\theta}(a), P)\mathbf{x} + \mathbf{v}(\rho(a), \mathbf{u}))\right)\Big| \\ &\leq \widetilde{\phi}_0(\mathbf{x}) \Big(c_1 + c_2 \|\mathbf{x}\|_2^2 + c_1 + c_2 \big\| Q(\boldsymbol{\theta}(a), P)\mathbf{x} + \\ &\quad + \mathbf{v}(\rho(a), \mathbf{u}) \big\|_2^2 \Big). \end{aligned} \tag{A.9}$$

The exponential decay of GM implies that

$$\sqrt{\widetilde{\phi}_0(\mathbf{x})} \Bigg(2c_1 + c_2 \Big(\|\mathbf{x}\|_2^2 + \\ + \left\| Q(s\boldsymbol{\theta}^{(0)}, P)\mathbf{x} + \mathbf{v}(s\rho^{(0)}, \mathbf{u}) \right\|_2^2 \Big) \Bigg) < c$$

for some $c > 0$. Thus, the function that dominates $|g(\cdot, \mathbf{x})|$ for $\|\mathbf{x}\|_2 > r$ is $c\sqrt{\widetilde{\phi}_0(\mathbf{x})}$ which obviously belongs[5] to $L^1(\mathbb{R}^d)$. $\square$

Convergence of NP-CCM

Theorems 1 and 2 do not guarantee the convergence of Algorithms 2 and 3 when adopting a nonparametric formula to estimate the change magnitude like (12). In fact, both theorems rely on the assumption that $\widetilde{\phi}_0$ is a GM. When using (12), we need additional assumptions on ϕ_0 to prove CCM convergence. More precisely, Theorem 1 holds if we assume that ϕ_0 is bounded over $\mathbb{R}^d$. In fact, in this case the integrand in (A.3) admits a dominant function independent from $\mathbf{v}$, thus f is continuous, proving Theorem 1. In case of Theorem 2, we have to assume the existence of a function $h \in L^1(\mathbb{R}^d)$ such that for every rotation matrix Q and translation vector $\mathbf{v}$, the following condition holds:

$$\left| \phi_0(\mathbf{x}) \log \frac{\phi_0(\mathbf{x})}{\phi_0(Q\mathbf{x} + \mathbf{v})} \right| \leq h(\mathbf{x}). \tag{A.10}$$

In this case the $g(a,\mathbf{x})$ in (A.8) is dominated by $h(\mathbf{x})$ and the function $kl(\cdot)$ in (A.7) is continuous. The assumption in (A.10) holds when the data-generating distribution is a GM, but it is difficult to verify for a more general class of distributions.

[5] There is no need to find a dominant function for $\|\mathbf{x}\|_2 \leq r$ since there $g(\cdot, \mathbf{x})$ is bounded.

References

[1] M. Basseville, I. V. Nikiforov, Detection of abrupt changes: theory and application, Prentice Hall Englewood Cliffs, 1993.

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

[3] X. Song, M. Wu, C. Jermaine, S. Ranka, Statistical change detection for multi-dimensional data, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD), 2007, pp. 667–676.

[4] M. Harel, S. Mannor, R. El-Yaniv, K. Crammer, Concept drift detection through resampling, in: Proceedings of the International Conference on Machine Learning (ICML), 2014, pp. 1009–1017.

[5] C. Alippi, G. Boracchi, M. Roveri, Hierarchical change-detection tests, IEEE Transactions on Neural Networks and Learning Systems 28 (2) (2017) 246–258.

[6] G. J. Ross, D. K. Tasoulis, N. M. Adams, Nonparametric monitoring of data streams for changes in location and scale, Technometrics 53 (4) (2011) 379–389.

[7] C. Alippi, G. Boracchi, M. Roveri, Change detection tests using the ici rule, in: Proceedings of the IEEE International Joint Conference of Neural Networks (IJCNN), 2010, pp. 1–7.

[8] C. Alippi, G. Boracchi, M. Roveri, Just-in-time classifiers for recurrent concepts, IEEE Transactions on Neural Networks and Learning Systems 24 (4) (2013) 620–634.

[9] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Computing Surveys (CSUR) 46 (4) (2014) 1–44.

[10] L. L. Minku, X. Yao, Ddd: A new ensemble approach for dealing with concept drift, IEEE Transactions on Knowledge and Data Engineering 24 (4) (2012) 619–633.

[11] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of the Brazilian

Symposium on Artificial Intelligence (SBIA), 2004, pp. 286–295.

[12] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: Proceedings of the International Workshop on Knowledge Discovery from Data Streams, 2006, pp. 1–10.

[13] G. Ditzler, R. Polikar, Hellinger distance based drift detection for nonstationary environments, in: Proceedings of the IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2011, pp. 41–48.

[14] C. Alippi, G. Boracchi, D. Carrera, M. Roveri, Change detection in multivariate datastreams: Likelihood and detectability loss, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Vol. 2, 2016, pp. 1368–1374.

[15] A. Zimek, E. Schubert, H.-P. Kriegel, A survey on unsupervised outlier detection in high-dimensional numerical data, Statistical Analysis and Data Mining: The ASA Data Science Journal 5 (5) (2012) 363–387.

[16] D. Carrera, B. Rossi, D. Zambon, P. Fragneto, G. Boracchi, Ecg monitoring in wearable devices by sparse models, in: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), 2016, pp. 145–160.

[17] A. Adler, M. Elad, Y. Hel-Or, E. Rivlin, Sparse coding with anomaly detection, in: Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 2013, pp. 1–6.

[18] L. I. Kuncheva, W. J. Faithfull, Pca feature extraction for change detection in multidimensional unlabeled data, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 69–80.

[19] A. M. Narasimhamurthy, L. I. Kuncheva, A framework for generating data to simulate changing environments., in: Proceedings of the IASTED International Multi-Conference: artificial Intelligence and Applications (AIA), 2007, pp. 384–389.

[20] L. L. Minku, A. P. White, X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Transactions on Knowledge and Data Engineering 22 (5) (2010) 730–742.

[21] P. Lindstrom, S. J. Delany, B. Mac Namee, Autopilot: Simulating changing concepts in real data, in: Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science (AICS), 2008, pp. 1–10.

[22] M. Lichman, UCI machine learning repository (2013). URL `http://archive.ics.uci.edu/ml`

[23] C. Alippi, G. Boracchi, D. Carrera, CCM: Controlling the change magnitude in high dimensional data, in: Proceedings of the INNS Conference on Big Data, 2016, pp. 216–225.

[24] F. Pérez-Cruz, Estimation of information theoretic measures for continuous random variables, in: Advances in Neural Information Processing Systems (NIPS), 2009, pp. 1257–1264.

[25] G. Boracchi, M. Roveri, Exploiting self-similarity for change detection, in: Proceedings of the IEEE International Joint Conference onNeural Networks (IJCNN), 2014, pp. 3339–3346.

[26] H.-V. Nguyen, J. Vreeken, Linear-time detection of nonlinear changes in massively high dimensional time series, in: Proceedings of the 2016 SIAM International Conference on Data Mining, 2016, pp. 828–836.

[27] W. N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2001, pp. 377–382.

[28] K. B. Dyer, R. Capo, R. Polikar, Compose: A semisupervised learning framework for initially labeled nonstationary streaming data, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 12–26.

[29] F. Gustafsson, F. Gustafsson, Adaptive filtering and change detection, Wiley New York, 2000.

[30] C. Alippi, G. Boracchi, M. Roveri, An effective just-in-time adaptive classifier for gradual concept drifts, in: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 2011, pp. 1675–1682.

[31] L. Ljung, System identification, in: Signal analysis and prediction, Springer, 1998, pp. 163–173.

[32] C. Alippi, G. Boracchi, B. Wohlberg, Change detection in streams of signals with sparse representations, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 5252–5256.

[33] B. Zhao, L. Fei-Fei, E. P. Xing, Online detection of unusual events in videos via dynamic sparse coding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 3313–3320.

[34] D. Carrera, G. Boracchi, A. Foi, B. Wohlberg, Scale-invariant anomaly detection with multiscale group-sparse models, in: Proceedings of the IEEE International Conference on Image Processing (ICIP), 2016, pp. 3892–3896.

[35] D. Carrera, G. Boracchi, A. Foi, B. Wohlberg, Detecting anomalous structures by convolutional sparse models, in: Proceedings of the IEEE-INNS International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8.

[36] T. M. Cover, J. A. Thomas, Elements of information theory, John Wiley & Sons, 2012.

[37] G. McLachlan, D. Peel, Finite mixture models, John Wiley & Sons, 2004.

[38] L. Trefethen, D. Bau, Numerical linear algebra, Siam, 1997.

[39] C. Alippi, Intelligence for Embedded Systems: A Methodological Approach, Springer, 2014.

[40] N. Leonenko, L. Pronzato, V. Savani, et al., A class of rényi information estimators for multidimensional densities, The Annals of Statistics 36 (5) (2008) 2153–2182.

[41] X. Nguyen, M. J. Wainwright, M. I. Jordan, Estimating divergence functionals and the likelihood ratio by convex risk minimization, IEEE Transactions on Information Theory 56 (11) (2010) 5847–5861.

[42] Q. Wang, S. R. Kulkarni, S. Verdú, A nearest-neighbor approach to estimating divergence between continuous random vectors, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), 2006, pp. 242–246.

[43] J. H. Friedman, J. L. Bentley, R. A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Transactions on Mathematical Software 3 (3) (1977) 209–226.

[44] T. Dasu, S. Krishnan, S. Venkatasubramanian, K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in: Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications, 2006.

[45] Y. Lepage, A combination of wilcoxon's and ansari-bradley's statistics, Biometrika 58 (1) (1971) 213–217.

[46] H. Bauer, Measure and integration theory, Walter de Gruyter, 2001.

[47] W. Rudin, et al., Principles of mathematical analysis, McGraw-Hill New York, 1964.

[48] R. L. Burden, J. D. Faires, Numerical analysis. 2001, Brooks/Cole, USA, 2001.