CCM: Controlling the Change Magnitude in High Dimensional Data

Cesare Alippi[1,2], Giacomo Boracchi[1], Diego Carrera[1]
[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy
[2] Università della Svizzera Italiana, Lugano, Switzerland
{firstname.lastname@polimi.it}

Abstract The effectiveness of change-detection algorithms is often assessed on real-world datasets by injecting synthetically generated changes. Typically, the magnitude of the introduced changes is not controlled, and most of experimental practices lead to results that are difficult to reproduce and compare with. This problem becomes particularly relevant when the data-dimension scales, as it happens with big data.
To enable a fair comparison among change-detection algorithms, we have designed "Controlling Change Magnitude" (CCM), a rigorous method to introduce changes in multivariate datasets. In particular, we measure the change magnitude as the symmetric Kullback-Leibler divergence between the pre- and post-change distributions, and introduce changes by applying a roto-translation directly to the data. We present an algorithm to identify the parameters yielding the desired change magnitude, and analytically prove its convergence. Our experiments show the effectiveness of the proposed method and the limitations of tests run on high-dimensional datasets when changes are injected following traditional approaches. The proposed method is implemented in a MATLAB framework, which is made publicly available for download.

1 Introduction

Change detection is an important problem in machine learning and data mining. Change-detection tests [1, 2, 3, 4] and outlier/anomaly detectors [5, 6] provide precious information for adaptation and data-analysis purposes, like revealing unforeseen evolutions of the process generating the data or anomalous events. One of the issues to be considered when using change-detection algorithms in big data applications concerns the assessment of detection performance, which requires special care when the data dimension scales. To this purpose, we present a method to yield rigorous testbeds and enable reproducible and comparable experiments on high dimensional data.

To get stable performance measures, change-detection algorithms need to be executed on a large number of datasets and, to study the detection performance when data dimension scales, datasets having different number of components are needed. Moreover, all these datasets should be affected by changes at known locations. Unfortunately, there are not many real-world datasets exhibiting real changes that satisfy these requirements. As such, experiments typically involve few real-world datasets, and results have often to be interpreted by visual inspection, as in the tests on financial time series in [2]. While these experiments indicate whether the algorithm is successful or not, visual inspection does not provide a sound quantitative assessment of the performance, and is not viable when dimension increases.

In practice, experiments are often performed on either synthetically generated data or on real-world dataset that have been manipulated introducing changes at known locations. The first option consists in choosing two different distributions, ϕ_0 and ϕ_1, to generate the pre- and post-change data, respectively (see e.g. [7, 2]). In online classification problems, changes (concept drifts [8]) can be also introduced by modifying the label assignment-criteria, rather than the distribution of the raw data [9, 10, 11]. Generating synthetic datasets is often the easiest option to arbitrarily increase the number of datasets to be tested, thus achieve the desired accuracy in the performance measures. However, synthetic data rarely represent realistic monitoring scenarios, as they follow quite simplistic distributions.

Manipulating real-world data is certainly more appealing, since this allows testing change-detection algorithms in realistic monitoring conditions. However, special care should be taken when introducing changes, to make sure that their magnitude is in a reasonable range. Controlling the change magnitude is important to make experiments reproducible, and to fairly compare the results on different datasets. In particular, controlling the change magnitude is necessary to study how change-detection performance vary when data dimension scales [12]. Most of the papers in the literature resort to introducing arbitrary shifts/scaling [7], swapping few components [3] of the original dataset, or mixing different datasets [13]. Unfortunately, these practices yield changes having magnitude that depends on the data dimension and distribution.

We here describe "Controlling Change Magnitude" (CCM), a rigorous method to introduce changes having a controlled magnitude. The method is flexible and allows to operate on real-world datasets having arbitrary dimensions. In particular, we approximate ϕ_0 by a Gaussian mixture (GM), and measure the change magnitude by the symmetric Kullback-Leibler (sKL) divergence between ϕ_0 and ϕ_1. We introduce changes by roto-transalating the data, and the transformation parameters are estimated through an iterative algorithm (Sections 3) to yield constant change magnitude.

This method was briefly introduced in [12], where it was used to generate datastreams for investigating the detectability-loss, a problem that affects change-detection algorithms monitoring the log-likelihood w.r.t. ϕ_0. We here

provide a detailed description of the method and proof the convergence of algorithms used for estimating the roto-translation parameters (Section 4). We also illustrate the main functionalities of our software framework[1] (Section 5) that can be used to prepare datastreams for change-detection purposes from datsets of popular machine learning repositories like the UCI one [14]. Finally (Section 6), we demonstrate the effectiveness of the proposed method, while changes introduces by other methods using can yield to considerably different magnitude where data dimension scales, as in big data scenario.

2 Problem Formulation

We consider a dataset S of stationary data, i.e. containing independent and identically distributed (i.i.d.) random vectors $\mathbf{s} \in \mathbb{R}^d$. We assume that S is generated by a probability density function (pdf) ϕ_0 that is possibly unknown. We want to generate a datastream $X = \{\mathbf{x}(t),\, t = 1, \dots\}$ affected by a change at time $t = \tau$ as

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

where $\mathbf{v} \in \mathbb{R}^d$ and $Q \in \mathbb{R}^{d\times d}$ is an orthogonal matrix, i.e. $Q'Q = QQ' = I$. This change model allows us to assemble X by simply selecting $\mathbf{x}(t)$ from S when $t < \tau$, and roto-translating the remaining $\mathbf{s} \in S$ after the change point, i.e. $\mathbf{x}(t) = Q'(\mathbf{s} - \mathbf{v})$ for $t \geq \tau$. This is a quite a general model which includes changes in the mean as well in the correlation among the data components.

Our goal is to define Q and $\mathbf{v}$ such that the change in (1) features a specific magnitude, which we measure by the symmetric Kullback-Leibler divergence (also known as Jeffrey divergence) between ϕ_0 and ϕ_1

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

In practice, given ϕ_0 and the desired magnitude κ, we have to estimate $\mathbf{v} \in \mathbb{R}^d$ and $Q \in \mathbb{R}^{d\times d}$ such that

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

The symmetric Kullbach-Leibler divergence is quite a natural choice in the change-detection scenario, as discussed in [12]. We observe that (2) is well posed if only if the supports of ϕ_0 and ϕ_1 coincide. For this reason we here assume that ϕ_0 is strictly positive on $\mathbb{R}^d$, and so ϕ_1.

[1] publicly available for download at: `diegopage`

Input: $\widetilde{\phi}_0$, target value κ of $\mathrm{sKL}(\widetilde{\phi}_0, \phi_1)$
Output: Roto-translation parameters $\boldsymbol{\theta}^{(0)}$, P, $\rho^{(0)}$, $\mathbf{u}$
Set $\rho^{(0)} = 1$.
repeat
  Randomly generate m angles $\boldsymbol{\theta}^{(0)}$ in $[-\pi/2, \pi/2]^m$ and a unitary vector $\mathbf{u}$.
  Generate a matrix $A \in \mathbb{R}^{d \times d}$ with Gaussian entries.
  Set P as the orthogonal matrix of the QR decomposition of A.
  Set $Q^{(0)}(\boldsymbol{\theta}^{(0)}, P) = PS(\boldsymbol{\theta}^{(0)})P'$ and $\mathbf{v}(\rho^{(0)}, \mathbf{u}) = \rho^{(0)}\mathbf{u}$.
  Compute $s^{(0)} = \mathrm{sKL}(\widetilde{\phi}_0, \phi_1)$, where $\phi_1 = \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{v}^{(0)})$
  $\rho^{(0)} = 2\rho^{(0)}$.
until $s^{(0)} > \kappa$;

Algorithm 1: Definition of initialization parameters.

3 Method Description

In this section we describe the algorithms to iteratively compute the rotation matrix Q and the translation vector $\mathbf{v}$ yielding $\mathrm{sKL}(\phi_0, \phi_1) \approx \kappa$. To compute $\mathrm{sKL}(\phi_0, \phi_1)$ in (2) both ϕ_0 and ϕ_1 are necessary; however, since ϕ_0 is typically unknown, we replace it with a GM estimate $\widetilde{\phi}_0$, which is computed on the whole dataset S. We also adopt a suitable parametrization for Q and $\mathbf{v}$, that are randomly initialized as in Algorithm 1. These parameters are then adjusted using a bisection method, described in Algorithm 2, to achieve $\mathrm{sKL}(\phi_0, \phi_1)$ that is close to the target value κ up to a tolerance ε.

Fitting Pre-Change Distribution: To define ϕ_1 satisfying (3) we need to know ϕ_0, which is typically unknown. Therefore, we compute its estimate $\widetilde{\phi}_0$ by fitting a GM on the whole dataset of stationary data S. We adopt GM since these are flexible models that can well approximate skewed and multimodal distributions even in high dimensions [3].

The pdf $\widetilde{\phi}_0$ of a GM is a convex combination of k Gaussian functions

$$\widetilde{\phi}_0(\mathbf{x}) = \sum_{i=1}^{k} \frac{\lambda_i}{(2\pi)^{d/2}\det(\Sigma_i)^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)}, \tag{4}$$

where $\sum_{i=1}^{k} \lambda_i = 1$ and $\lambda_i \geq 0$, $i \in 1, \ldots, k$ are the weights of each Gaussian having mean μ_1 and covariance Σ_i. We estimate the weights λ_i and the parameters of the Gaussians using an Expectation-Maximization (EM) algorithm [15], and select the best value of k via K-fold cross validation.

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

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

Algorithm 2: Computation of the roto-translation yielding sKL $\approx \kappa$.

$$S(\boldsymbol{\theta}) = \begin{bmatrix} R(\theta_1) & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & R(\theta_m) & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}, \qquad R(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}. \tag{5}$$

where $R(\theta_i) \in \mathbb{R}^{2\times 2}$ defines a counterclockwise rotation of angle θ_i around the origin in $\mathbb{R}^2$. Note that, when d is even, the last column and the last row of $S(\boldsymbol{\theta})$ are dropped. The matrix $S(\boldsymbol{\theta})$ 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 $S(\boldsymbol{\theta})$ against an orthogonal matrix $P \in \mathbb{R}^{d\times d}$. Thus, the rotation matrix is parametrized as

$$Q(\boldsymbol{\theta}, P) = PS(\boldsymbol{\theta})P'. \tag{6}$$

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

$$\mathbf{v}(\rho, \mathbf{u}) = \rho\mathbf{u}, \qquad \text{where } \|\mathbf{u}\|_2 = 1,\ \rho = \|\mathbf{v}\|_2. \tag{7}$$

Initialization: Algorithm 1 is meant to define $Q^{(0)}$ and $\mathbf{v}^{(0)}$ satisfying $\mathrm{sKL}(\phi_0, \phi_0(Q^{(0)} \cdot + \mathbf{v}^{(0)})) > \kappa$. We initially set $\rho^{(0)} = 1$ (line 2) and randomly generate a vector $\mathbf{u}$ having unitary norm (line 4), as in [16]. Moreover, we randomly define the angles of rotation $\boldsymbol{\theta}^{(0)}$ in $[-\pi/2, \pi/2]^m$ (line 4) and randomly select an orthogonal matrix P (lines 5-6). We then compute the rotation matrix $Q^{(0)}(\boldsymbol{\theta}^{(0)}, P)$ and translation vector $\mathbf{v}^{(0)}(\rho^{(0)}, \mathbf{u})$ as in (6) and (7), respectively (line 7). If the corresponding $\mathrm{sKL}(\phi_0, \phi_1)$ is larger than κ, the algorithm terminates and we use $Q^{(0)}$ and $\mathbf{v}^{(0)}$ to initialize Algorithm 2.

Otherwise, we double $\rho^{(0)}$ and repeat this procedure until the condition is satisfied. Convergence of Algorithm 1 is proved in Section 4.

Iterations: Algorithm 2 implements a bisection method to compute $\boldsymbol{\theta}$ and ρ yielding the desired sKL value. The algorithm is initialized from $\boldsymbol{\theta}^{(0)}$, P, $\rho^{(0)}$, and $\mathbf{u}$ provided by Algorithm 1, and iteratively adjusts $\boldsymbol{\theta}$ and ρ, while P and $\mathbf{u}$ are preserved. We use the subscripts l and u to denote the parameters yielding $\mathrm{sKL}(\widetilde{\phi}_0, \phi_1)$ values that are smaller and larger than κ, respectively.

Initially, $\boldsymbol{\theta}_l$ and ρ_l are set to 0 (line 2), while $\boldsymbol{\theta}_u = \boldsymbol{\theta}^{(0)}$ and $\rho_u = \rho^{(0)}$, (line 3). As in any bisection method, we set $\boldsymbol{\theta}^{(j)}$ to the average of $\boldsymbol{\theta}_l^{(j)}$ and $\boldsymbol{\theta}_u^{(j)}$ (line 6), and $\rho^{(j)}$ to the average of $\rho_l^{(j)}$ and $\rho_u^{(j)}$ (line 7). Then, if the corresponding value of sKL, namely $s^{(j)}$ (line 8), is smaller than the target value κ, we update $\boldsymbol{\theta}_l$ and ρ_l (line 10), otherwise we update $\boldsymbol{\theta}_u$ and ρ_u (line 12). These steps are iterated until the sKL achieves the target value κ up to the desired tolerance ε. Convergence of Algorithm 2 is demonstrated in Section 4.

4 Proofs of Convergence

To proof the convergence of the proposed method we demonstrate first that Algorithm 1 terminates after a finite number of iterations (Theorem 1), and then that Algorithm 2 actually converges (Theorem 2).

Theorem 1. *Let $\widetilde{\phi}_0$ be a Gaussian mixture. Then, for any $\kappa > 0$, Algorithm 1 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 $\|\mathbf{v}\|_2 \to \infty$. The lower bound follows from the definition of sKL and the Jensen's inequality:

$$\begin{aligned} \mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q^{(0)} \cdot + \mathbf{v})) &\geq \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log\left(\frac{\widetilde{\phi}_0(\mathbf{x})}{\widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v})}\right) d\mathbf{x} \\ \geq \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \log(\widetilde{\phi}_0(\mathbf{x})) dx &- \log\left(\int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{x}\right). \end{aligned} \tag{8}$$

We now define $f(\mathbf{v}) := \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{x}$, which we observe is a continuous function (see Lemma 16.1 of [17]). We prove the theorem by demonstrating that $f(\mathbf{v}) \in L^1(\mathbb{R}^d)$, as this implies that $f(\mathbf{v}) \to 0$ when $\|\mathbf{v}\|_2 \to \infty$, thus that the lower bound in (8) diverges:

$$\begin{aligned} \int_{\mathbb{R}^d} |f(\mathbf{v})| \, d\mathbf{v} = \int_{\mathbb{R}^d} f(\mathbf{v}) d\mathbf{v} = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{x} d\mathbf{v} = \\ = \int_{\mathbb{R}^d} \widetilde{\phi}_0(\mathbf{x}) \left[\int_{\mathbb{R}^d} \widetilde{\phi}_0(Q^{(0)}\mathbf{x} + \mathbf{v}) d\mathbf{v}\right] d\mathbf{x} = 1. \end{aligned} \tag{9}$$

□

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

Proof. The thesis follows from the Intermediate Value Theorem [18] applied to the function used in the bisection procedure, i.e.

$$sKL(s) := \mathrm{sKL}(\widetilde{\phi}_0, \widetilde{\phi}_0(Q(\boldsymbol{\theta}(s), P) \cdot + \mathbf{v}(s))), \tag{10}$$

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 (Algorithm 2, lines 2-3). We observe that $\mathrm{sKL}(0) < \kappa$ (since $sKL(0) = 0$) and $sKL(1) > \kappa$, thus it is enough to show that $sKL(\cdot)$ is continuous in $[0,1]$ to prove that there exist a value of $\bar{s}$ such that $sKL(\bar{s}) = \kappa$, thus that $\boldsymbol{\theta}(\bar{s})$ and $\rho(\bar{s})$ yield sKL equal to κ, thus, that bisection method converges (see [19], Theorem 2.1).

To show that $sKL(\cdot)$ in (10) is continuous, we show that $KL(\cdot)$ is continuous which we write as $KL(s) := \int_{\mathbb{R}^d} g(s, \mathbf{x}) d\mathbf{x}$, being

$$g(s, \mathbf{x}) := \widetilde{\phi}_0(\mathbf{x}) \log \left(\frac{\widetilde{\phi}_0(\mathbf{x})}{\widetilde{\phi}_0(Q(\boldsymbol{\theta}(s), P)\mathbf{x} + \mathbf{v}(\rho(s), \mathbf{u}))} \right), \tag{11}$$

It is clear that g is continuous in $[0,1] \times \mathbb{R}^d$, thus $g(\cdot, \mathbf{x})$ is continuous in $[0,1]$: to show that $KL(\cdot)$ is also continuous, we leverage Lemma 2.1 in [17] and prove that $|g(s, \mathbf{x})|$ admits a dominant integrable function that does not depend on s. To this purpose, we exploit the following upperbound [2]

$$\left| \log(\widetilde{\phi}_0(\mathbf{x})) \right| \leq c_1 + c_2 \left\| \mathbf{x} \right\|_2^2, \qquad \forall\, \mathbf{x} \text{ s.t. } \left\| \mathbf{x} \right\|_2 > r. \tag{12}$$

that holds for some constants $c_1, c_2, r > 0$. Then, for a sufficiently large r

$$\begin{aligned} |g(s, \mathbf{x})| &= \widetilde{\phi}_0(\mathbf{x}) \left(\log(\widetilde{\phi}_0(\mathbf{x})) - \log(\widetilde{\phi}_0(Q(\boldsymbol{\theta}(s), P)\mathbf{x} + \mathbf{v}(\rho(s), \mathbf{u}))) \right) \\ &\leq \widetilde{\phi}_0(\mathbf{x})(c_1 + c_2 \left\| \mathbf{x} \right\|_2^2)(c_1 + c_2 \left\| Q(\boldsymbol{\theta}(s), P)\mathbf{x} + \mathbf{v}(\rho(s), \mathbf{u}) \right\|_2^2). \end{aligned} \tag{13}$$

The exponential decay of GM implies that

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

for some $c > 0$. Thus, the function that dominates $|g(\cdot, \mathbf{x})|$ for $\left\| \mathbf{x} \right\|_2 > r$ is $c\sqrt{\widetilde{\phi}_0(\mathbf{x})}$ which obviously belongs[3] to $L^1(\mathbb{R}^d)$. □

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

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

5 The Framework

We implement the proposed method in a MATLAB framework which is publicly available for download at `http://home.deib.polimi.it/carrerad`. The main function of the framework is `generateDatastreams`, which manipulates a dataset S containing i.i.d. data and generates multiple datastreams affected by a change as in (1). At first, this function performs a random shuffling of the dataset to obtain different datastreams everytime the function is invoked, then it estimates the pdf $\widetilde{\phi}_0$ over the whole S and generates a change of controlled magnitude κ at time τ by performing a roto-translation of the data (Section 2).

The pdf $\widetilde{\phi}_0$ is implemented as the object `gmDistr`, which includes `fit` and `symmetricKullbackLeibler` among its methods. The method `fit` estimates $\widetilde{\phi}_0$ over the dataset S and is based on the MATLAB implementation of the EM algorithm [15]. The method `symmetricKullbackLeibler` takes as input the ϕ_1 and estimates $\mathrm{sKL}(\widetilde{\phi}_0, \phi_1)$ via a Monte Carlo simulation which involves the computation of the log-likelihood with respect to $\widetilde{\phi}_0$ and ϕ_1 on synthetically generated samples. To prevent severe numerical raising when computing the log-likelihood of a GM via (4) (in particular when $d \gg 1$), we adopt the following upper-bound that approximates $\widetilde{\phi}_0(\mathbf{x})$ as in [12]

$$\widetilde{\phi}_0(\mathbf{x}) \leq -\frac{k\lambda_{i^*}}{2}\Big(\log\left((2\pi)^d \det(\Sigma_{i^*})\right) + (\mathbf{x}-\mu_{i^*})'\Sigma_{i^*}^{-1}(\mathbf{x}-\mu_{i^*})\Big) \qquad (14)$$

where $i^* = \underset{i=1,\ldots,k}{\operatorname{argmax}}\left(\frac{\lambda_i}{(2\pi)^{d/2}\det(\Sigma_i)^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)'\Sigma_i^{-1}(\mathbf{x}-\mu_i)}\right)$. The rotation matrix Q and the translation vector $\mathbf{v}$ are computed by `computeRotoTranslation`, which implements both Algorithms 1 and 2, as in Section 3.

6 Experiments

Experiments are meant to demonstrate the effectiveness of CCM and the importance of controlling the change magnitude, showing the limitations of methods typically used in the literature which do no allow such a control.

In our experiments we use the *MiniBooNE Particle Dataset* from the UCI repository [14], which contains measurements from a physical experiment designed to distinguish electron neutrinos from muon neutrinos. We consider only the data from the muon class, obtaining a dataset of dimension 50 with 93108 samples. We fit a mixture $\widetilde{\phi}_0$ of $K = 2$ Gaussians, where K is selected by 5-fold cross-validation. To avoid numerical issues, we preprocess the dataset by standardizing each component (i.e. subtracting the sample sample mean and dividing by the sample standard deviation). We generate 1000 datasets for each value of $d = 1, \ldots, 50$, by randomly choosing d components out of the 50 available.

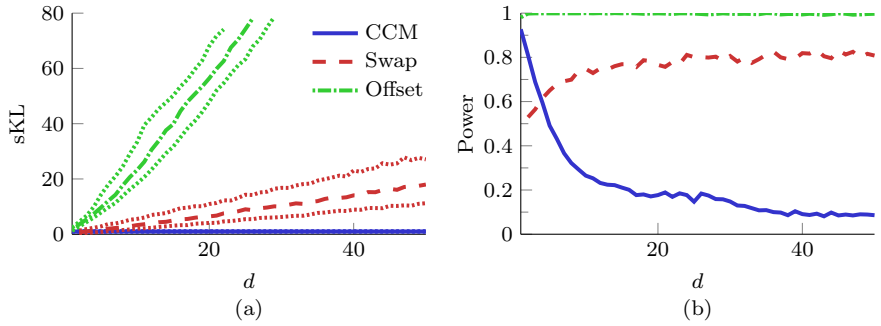

Fig. 1 (a) Values of $\text{sKL}(\widetilde{\phi}_0, \phi_1)$ obtained using CCM, Swap and Offset methods to introduce changes when varying d. Solid, dashed and dash-dotted lines represent the median values, while the dotted lines represent the first and the third quartiles. (b) The estimated powers of t-tests on the log-likelihood values.

We consider three methods to synthetically introduce changes:
CCM: which was configured to yield constant $\text{sKL}(\widetilde{\phi}_0, \phi_1) = 1 \, \forall d$.
Offset: an offset $\nu = 1$ is added to each component of the data (after preprocessing ν equals the standard deviation of each component). With respect to (1) this change model has $Q = I_d$ and $\mathbf{v} = \nu \mathbf{1}_d$.
Swap: two components, randomly chosen, are swapped. With respect to (1) this change model has Q equal to the corresponding permutation matrix and $\mathbf{v} = 0$.

To measure the magnitude of the introduced changes, we compute $\text{sKL}(\widetilde{\phi}_0, \phi_1)$ by fitting $\widetilde{\phi}_0$ ($K = 2$) on each d-dimensional dataset and setting ϕ_1 as in (1) using the parameters defined by the change model. Figure 1(a) shows the distribution of $\text{sKL}(\widetilde{\phi}_0, \phi_1)$ when varying d. While CCMclearly preserves $\text{sKL} = 1$, the Offset and Swap methods do not, and sKL increases with d. Moreover, the dispersion of sKL also increases, indicating that arbitrarily introducing changes using Offset and Swap methods can lead to considerably different magnitudes when d increases.

Controlling the change-magnitude is important as this determines the change-detection performance, to a large extent. As an example, we perform t-test on the mean value of the log-likelihood computed over two windows W_P and W_R containing 200 pre- and post-change samples, respectively. The log-likelihood is computed with respect to a GM estimated on a training set of pre-change data having $200d$ samples. Figure 1(b) reports the estimated power of the t-test when changes are introduced by the different methods and for different values of d. While changes introduced by CCMreport a decreasing power, coherently with the detectability-loss studies [12], the power in case of changes introduced by Swap and Offset increases with d, preventing to study how d influences the change-detection performance.

7 Conclusions

We have presented CCM, a rigorous method to control the change magnitude in multivariate datasets. The algorithms at the core of CCMare sound and a proof of their convergence is given. Our tests remark the importance of using CCMto control the change magnitude, rather than more heuristic experimental practices, as this enables to fairly assess the detection performance and investigate the change-detection challenges raising in high dimensional data.

References

1. Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "Hierarchical change-detection tests," *IEEE Trans. on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2016.
2. Gordon J Ross, Dimitris K Tasoulis, and Niall M Adams, "Nonparametric monitoring of data streams for changes in location and scale," *Technometrics*, vol. 53, no. 4, 2011.
3. Ludmila I Kuncheva, "Change detection in streaming multivariate data using likelihood detectors," *IEEE Trans. on Knowledge and Data Engineering*, vol. 25, no. 5, 2013.
4. Maayan Harel, Shie Mannor, Ran El-yaniv, and Koby Crammer, "Concept drift detection through resampling," in *Proc. of ICML*, 2014, pp. 1009–1017.
5. Varun Chandola, Arindam Banerjee, and Vipin Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, July 2009.
6. Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.
7. Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "A Just-In-Time adaptive classification system based on the Intersection of Confidence Intervals rule," *Neural Networks*, vol. 24, no. 8, pp. 791 – 800, 2011.
8. João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, 2014.
9. Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues, "Learning with drift detection," in *Proc. of Brazilian Symposium on Artificial Intelligence (SBIA)*, 2004.
10. Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 4, 2013.
11. W Nick Street and YongSeog Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proc. of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.
12. Cesare Alippi, Giacomo Boracchi, Diego Carrera, and Manuel Roveri, "Change detection in multivariate datastreams: Likelihood and detectability loss," in *Proc. of IJCAI*, 2016, http://arxiv.org/abs/1510.04850.
13. G. Boracchi and M. Roveri, "Exploiting self-similarity for change detection," in *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2014.
14. M. Lichman, "UCI machine learning repository," 2013.
15. Geoffrey McLachlan and David Peel, *Finite mixture models*, John Wiley & Sons, 2004.
16. Cesare Alippi, *Intelligence for Embedded Systems, a Methodological Approach*, Springer, 2014.
17. Heinz Bauer, *Measure and integration theory*, Walter de Gruyter, 2001.
18. Walter Rudin, *Principles of mathematical analysis*, McGraw-Hill New York, 1964.
19. Richard L Burden and J Douglas Faires, *Numerical analysis*, 2001.