Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione
Tesi di Laurea Magistrale in
Ingegneria Biomedica - Biomedical Engineering

Concept Drift Detection in High-Dimensional Spaces

Advisor: Prof. Giacomo Boracchi

Author: Michelangelo Olmo Nogara Notarianni
Student ID: 996134
Academic Year: 2022-23

Realism has nothing to do with the Real.
On the contrary, the Real is what realism has continually to suppress.

Mark Fisher

Abstract

In predictive modeling, the assumption of static relationships between input and output data often fails. The phenomenon can be addressed as *concept drift*, signifying shifts in data patterns over time. In high-dimensional multivariate data streams, common in fields like IoT and text analysis, the curse of dimensionality complicates the identification of meaningful changes. Feature reduction techniques, such as Principal Component Analysis (PCA), offer a solution but come with computational costs.

This study addresses the performance of the QuantTree (QT) algorithm in high-dimensional dataframes, extending its exploration to the generalized Kernel-QT (KQT) and its online variant, QT-EWMA. A novel addition to this landscape is proposed—Kernel-QuantTree Exponentially Weighted Moving Average (KQT-EWMA). In evaluating and building a monitoring system, we consider the balance between True Positive Rate (TPR) and False Positive Rate (FPR) to be crucial. We also consider other quantitative criteria such as execution time.

The problem formulation revolves around a change detection algorithm's three main components: a model of the initial distribution, a statistic derived from it, and a decision rule. We assume both the initial distribution and the changing distribution to be unknown. The study employs a QuantTree-like histogram fitted on the distribution to estimate the model. Two modes of drift detection - batch-wise and online - are explored. The study investigates the limitations and advantages of different dimensionality reduction techniques, including PCA and Random Projections. The study uncovers the impact of dimensionality together with the availability of training data on QT's performance. We extend the analysis to incorporate l_p norms as alternative kernel functions for KQT, considering quasi-norms with $p \leq 1$ that might be suitable in high dimensional dataframes.

Introducing KQT-EWMA, an online nonparametric change-detection algorithm, we extend some previous theoretical results and compare its performance against existing methods; KQT-EWMA stands out as comparable or superior with respect to the state of the art. The complexities addressed in this dynamic environment underscore the ongoing challenges and the need for simple solutions in the ever-evolving landscape of data analysis.

Keywords: Concept Drift Detection, QuantTree Algorithm, Kernel QuantTree (KQT), High-Dimensional Data, False Positive Rate (FPR) Control, Principal Component Analysis (PCA), Distance Metrics in High-Dimensional Space, Online Change Detection, Multivariate Data Streams

Compendio

Nel realizzare modelli di classificazione, analisi predittiva, etc., l'assunzione di relazioni statiche tra i dati di input e output - cioè che le relazioni tipo $y = f(x)$ rimangano vere nel tempo - spesso fallisce. Il fenomeno si può chiamare "concept drift", o *deriva del concetto*, indicando cambiamenti fra le quantità e relazioni statistiche. Considerando flussi di dati multivariati, comuni in settori come l'IoT e l'analisi testuale, l'identificazione di cambiamenti significativi è estremamente complicata. Tecniche di riduzione della dimensionalità come l'Analisi delle Componenti Principali (PCA) offrono - a volte - una soluzione ma comportano anche altri problemi e costi computazionali.

Questo studio osserva le analisi dell'algoritmo QuantTree (QT) di dataframe ad alta dimensionalità, estende l'osservazione alla sua versione generalizzata Kernel-QT (KQT) e alla sua variante online, QT-EWMA. Viene proposto l'algoritmo Kernel-QuantTree Exponentially Weighted Moving Average (KQT-EWMA). Nel costruire sistemi di monitoraggio, consideriamo l'equilibrio tra il tasso di veri positivi (TPR) e il tasso di falsi positivi (FPR), con particolari attenzioni per quest'ultimo, e altri criteri quantitativi come il tempo di esecuzione, che abbiamo sempre registrato lungo questi esperimenti.

La formulazione del problema si fonda sui tre elementi principali di un algoritmo per concept drift detection: un modello della distribuzione iniziale, una statistica derivata da questo e una regola decisionale che valuti un cambiamento come tale. Assumiamo sconosciute sia la distribuzione iniziale che le derive che questa subisce nel tempo. Esploriamo le due modalità batch-wise (offline) e online, indagando limiti e vantaggi delle diverse tecniche di riduzione della dimensionalità d, incluse PCA e *Random Projections*, proiezioni randomiche. Lo studio mostra l'impatto di d sulle prestazioni di diversi algoritmi insieme alla disponibilità (in generale scarsa) di dati di training.

Continuiamo l'analisi di KQT considerando quasi-norme l_p come funzioni kernel alternative, noto che valori di $p \leq 1$ possono essere adatti in spazi ad alta dimensionalità. Con l'introduzione di KQT-EWMA, un algoritmo online e non parametrico, estiendiamo alcuni risultati teorici precedenti; ne confrontiamo le prestazioni con metodi esistenti. KQT-EWMA è comparabile allo stato dell'arte, superiore in ambienti controllati (con un sufficiente numero di punti di training).

Le complessità affrontate in questo ambiente dinamico portano senz'altro la necessità di soluzioni semplici.

Contents

Chapter 1

Introduction

In predictive modeling we approximate a function f to predict, given some input data x, the output $y = f(x)$. The function f is often assumed to be static, meaning the relationship between input and output data does not change. In practice, patterns in real-world databases change over time, and predictive models become obsolete. Concept Drift Detection is the task of detecting when these changes, or concept drifts, occur, in order to keep the patterns up-to-date, possibly without inducing them each time from scratch. We call *concept*, or "hidden context", the unknown relationship between inputs and output variables. When the data that is used for inference differs from the data used during model training (data drift), the model likely perform worse than expected.

A concept drift is neither a gradual change over time nor a recurring cyclical change, nor is it a sudden or abrupt change; it means that the statistical properties of the target variable, which the model is trying to predict, change in unforseen ways, and it's important to continuously monitor the inference data and compare it to the data used during training, since in a variety of cases the underlying distribution of incoming data will unpredictably drift. We could consider as example the dynamic nature of language. Natural Language Processing is applied in a wide range of use cases, from chatbots and machine translation to speech recognition; individuals tend to change their speech patterns over time, across different geographical location, social contexts, etc. As spam e-mails content and design change, users themselves change their idea of what "spam" is, and spam detectors should be able to decide how to respond.

NO SPORT IS LESS ORGANIZED THAN CALVINBALL!
NEW RULE! NEW RULE! IF YOU DON'T TOUCH THE 30-YARD BASE WICKET WITH THE FLAG, YOU HAVE TO HOP ON ONE FOOT!
12
4
5-2 WATTERSON
© 1990 Watterson Distributed by Andrews McMeel Syndication

Early approaches to the problem were initially parametric and univariate, with control charts introduced by Shewhart in 1926-1927 being among the first methods. These are statistical tools used in quality control to monitor and control processes over time; they consist in plotting data points on a graph with upper and lower control limits, and when observations fall outside these limits, a potential issue or change in the process is flagged. However, a limitation was that this method operated on a single sample basis, making it susceptible to change with just one out-of-range value. To address this limitation, Page introduced the Cumulative Sum (CUSUM) method in 1954, a parametric drift detection method that computes the cumulative sum of $ln(p(x_t|\theta_1))$, given θ_1 the parameter of the post-change distribution, and reports a change when the sum exceeds a certain threshold. Since this method is parametric and supervised, pre- and post-change distributions ϕ_0, ϕ_1 are assumed to be known. Hotelling proposed a multivariate approach in 1947, extending Shewhart's control charts. Using Hotelling's T^2 test statistic, it assumed known mean mu and covariance matrix Σ for each variable. However, like Shewhart's univariate version, it considers only a single sample.

Most existing multivariate methods are parametric and assume a known stationary distribution ϕ_0. Some recent methods monitor similarity with respect to models trained on data drawn from ϕ_0, like Gaussian processes or Kernel Density Estimators. Non-parametric methods for detecting any change are scarce; one of them is QuantTree (QT), an histogram based method presented in Boracchi et al. [6]; we will write about it in chapter 2 and use its variants in every experiment.

In our framework, we will deal with high dimensional dataframes, describing the curse of dimensionality and studying the behavior of concept drift detection algorithms when dimensionality scales; we also will discuss how data points occupy the some space around them, and what is a "good" or "dense" coverage of the space, studying algorithms behavior when a small number of training points is provided. We will start with studying the performances of QT and its advanced version Kernel-QuantTree [25] (KQT), comparing them with other state-of-the-art methods in the same framework. We will introduce new kernel functions derived from l_k norms with $k \leq 1$ trying to adapt KQT to the curse of dimensionality. Finally, we will discuss the performances of the online version of QuantTree [10], QT-Exponentially Weighted Moving Average (QT-EWMA), which tests each new point of a datastream real-time, and implement and introduce a novel streaming algorithm, the online adaptation of KQT, and compare it with QT-EWMA and other baseline methods. Our main contributions are:

- We set an experimental benchmark for studying concept drift detection algorithm performances in high dimensional spaces when a few training points are given.
- We extend Kernel-QuantTree using distances derived from l_k (quasi-)norms, with $k \leq 1$ to adapt it to the curse of dimensionality.
- We extend Kernel-QuantTree to an online algorithm, KQT-EWMA, while addressing its theoretical properties.

1.1 Problem Formulation

We need to address concept drift to ensure accurate and reliable predictions and decisions, and despite the ever-changing nature of this topic, it is essential to try to state an understanding of the problem formulation in a formal and hopefully concise manner. For a more detailed and comprehensive description, one can refer to "Learning under Concept Drift: A Review" by J.Liu, A.Liu, F.Dong, F.Gu, J.Gama and G.Zhang [20].

A change detection algorithm has usually three main ingredients: a model $\hat{\phi}_0$ of the initial distribution, a statistic based on it, and a decision rule to report changes. After estimating ϕ_0, an online algorithm is employed to assess whether each new sequence $\{x_1, \dots, x_t\}$ contains a change point. Typically, a statistic T based on $\hat{\phi}_0$ is computed for each incoming x_t or over an entire batch W of test points. The decision rule usually involves checking whether $T > h$, where h is a certain threshold. In the streaming case, the decision time t^* is defined as the first time instant when there is enough statistical evidence to claim that the data stream $\{x_1, \dots, x_{t^*}\}$ contains a change point, i.e.:

$$t^* = \min\{t : T_t > h_t\}$$

Online and batch-wise are two modes for drift detection. In batch-wise drift detection (also referred to as two-sample test), the idea idea is to infer whether two sample sets have been selected from the same population. A batch represent a discrete chunk of data collected over a specific time interval or event. In contrast, online drift detection continuously monitors data - sometimes, real-time - and some model is updated as each data point arrives. The choices between these approaches depends on the nature of the data and the specific requirements of the application.

1.1.1 Batch-wise monitoring

Our goal is to detect changes, namely those data points for which the distribution of the data-generating process changes. We process the incoming data in batches $W = x_1, ..., x_\nu$, where ν represents the number of samples in each batch. We consider batches of fixed size, each containing independent and identically distributed (i.i.d.) samples drawn from a unique stationary distribution ϕ, which support is $\mathcal{X} \subseteq \mathbb{R}^d$. We detect changes $\phi_0 \to \phi_1 \neq \phi_0$ using a hypothesis test (HT) that assesses whether the data in W aligns with the reference stationary distribution $\hat{\phi}_0$ learned from a training set TR.

$$TR = \{x_i \in \mathcal{X}, \quad i = 1, \dots, N\}, \quad \text{where } x_i \sim \phi_0$$

Indeed, we assume that both ϕ_0 and ϕ_1 are unknown, and only the training set is provided to estimate ϕ_0. In particular, we focus on situations where no analytic distribution seems to properly match the training data. We formulate the hypothesis test HT as:

$$H_0 : W \sim \phi_0 \quad \text{vs} \quad H_1 : W \sim \phi_1 \neq \phi_0 \tag{1.1}$$

where H_0 represents the null hypothesis where W follows the distribution ϕ_0, and H_1 is the alternative hypothesis stating that W follows a different distribution ϕ_1. These test are based on a test statistic $\mathcal{T}$ defined over the learn distribution $\hat{\phi}_0$ that fits the training set. We detect a change in the incoming W when:

$$\mathcal{T}(W) = \mathcal{T}(x_1, ..., x_\nu) > \tau \tag{1.2}$$

where τ is the threshold that controls the false positive rate, namely the proportion of type I errors. For each given test statistic $\mathcal{T}$ and reference FPR value α, we define a threshold τ such that:

$$P_{\phi_0}(\mathcal{T}(W) > \tau) \leq \alpha \tag{1.3}$$

where α is the reference FPR value, when P_{ϕ_0} denotes the probability under H_0 that W contains samples generated from ϕ_0.

1.1.2 Online monitoring

A data-stream is an ‘ùnbounded sequence of multidimensional, sporadic and transient observations made available along time” [4], meaning that observations do not happen in a pattern - sporadic, i.e. they are not continuous or regular - and that they only last for a short time. We consider a virtually unlimited multivariate datastream $x_1, x_2, \ldots$ in $\mathbb{R}^d$. We assume that, in the absence of changes, all the data samples are i.i.d. realizations of a random variable with an unknown distribution ϕ_0, which support is $\mathcal{X} \subseteq \mathbb{R}^d$. We define the changepoint τ as the unknown time instant when a change $\phi_0 \to \phi_1$ takes place. The data x_t follows the distribution:

$$x_t \sim \begin{cases} \phi_0, & \text{if } t < \tau \\ \phi_1, & \text{if } t \geq \tau \end{cases}$$

Here, x_t represents the random variable that follows the distribution ϕ_0 before the changepoint τ (in the so called *in control state*), and then follows the distribution ϕ_1 for t greater than or equal to τ (*out of control state*). We make the assumption that both ϕ_0 and $\phi_1 \neq \phi_0$ are unknown. To estimate ϕ_0, a training set TR, which consists of N stationary realizations from ϕ_0, is provided. Online change detection algorithms assess, for each new incoming sample x_t, whether the sequence $\{x_1, \ldots, x_t\}$ contains a change point. Typically, a statistic T_t is computed at each incoming x_t, then a decision rule is applied. Usually, this consists in controlling whether $T_t > h_t$ for an appropriate threshold h_t, and the detection time t^* is defined as the first time index when this happens:

$$t^* = \min\{t : T_t > h_t\}$$

The detection time t^* is the first instant when there is enough statistical evidence to claim that the datastream $\{x_1, \ldots, x_t\}$ contains a change point.

A fundamental issue in change detection is to define a sequence of thresholds $\{h_t\}_t$ to control false alarms. We measure false alarms by the Average Run Length, define as:

$$ARL_0 = \mathbb{E}_{\phi_0}[t^*] \tag{1.4}$$

where the expectation is taken assuming that the whole datastream is drawn from ϕ_0. Thus, the ARL_0 is the average time before a false alarm. Ideally, the target ARL_0 is set a priori, similar to the type I error probability in hypothesis testing. The goal is to detect a distribution change as soon as possible, minimizing the detection delay $t^* - \tau$, while controlling ARL_0. This means aiming for an empirical ARL_0 that approaches the target ARL_0 established before monitoring. It is worth noting that controlling ARL_0 also provides an upper bound on the expected detection delay.

1.2 Specific challenges

1.2.1 Multivariate Datastreams and High Dimensionality

In the era of Internet-of-Things (IoT) data streams have seen an explosion of information generated from heterogeneous data sources. Streaming data in certain scenarios can exhibit high dimensionality. Laboratory instruments become more and more complex, reporting hundreds or thousands of measurements for a single experiment: in a functional genomics, usually we have much more variables than observations! But we don't have to look at bioinformatics: automatic text analysis in a simple internet search, image processing, etc. need massive high dimensional data as input. Change detection has been widely investigated on scalar (univariate) data, but not in the multivariate case; a straightforward extension would be to independently inspect each component of the datastream, but this does not clearly provide a truly multivariate solution, e.g., it is unable to detect changes affecting the correlation among the data components. Apart from issues in terms of time and memory due to the increased computational complexity, not a lot of theoretical or experimental studies investigate how the data dimension d impacts on the change detectability. Change detection is an ill-posed problem especially in high dimensional spaces. The concept of change is highly context-dependent. How much of a difference, and in what feature space, do constitute a change? For example, in comparing X-ray images, a hair-line discrepancy in a relevant segment of the image may be an important sign of a broken bone, a tumor, etc. In these cases where distances between instances grow exponentially due to the *curse of dimensionality*, algorithm's performance drops not only in terms of time and memory. In [3] it is shown that the detectability of the changes having a given magnitude progressively reduce when d increases: they refer to this phenomenon as *detectability loss*, and analytically demonstrate that, in case of Gaussian random variables, the change visibility is upperbounded by a function decaying as $1/d$. It is also shown that detectability loss is not a consequence of density-estimation problems, as it holds either when data distribution is estimated from training samples or known.

The change magnitude of $\phi_0 \to \phi_1$ can be naturally measured by the symmetric Kullback-Leibler divergence between ϕ_0 and ϕ_1:

$$sKL(\phi_0, \phi_1) := KL(\phi_0, \phi_1) + KL(\phi_1, \phi_0) \tag{1.5}$$

where

$$KL(\phi_0, \phi_1) = \int_{\mathbb{R}^d} \log\left(\frac{\phi_0(x)}{\phi_1(x)}\right) \phi_0(x)\, dx \tag{1.6}$$

The choice is supported by the Stein's Lemma which states that $KL(\phi_0, \phi_1)$ yields an upper-bound for the power of parametric hypothesis tests that determine whether a given sample population is generated from ϕ_0 (null hypothesis) or ϕ_1 [18].

Dimensionality reduction, i.e. mapping high-dimensional instances onto a lower-dimensional representation while conserving the distances between instances, is a well known preprocessing method to handle high dimensional data that may increase the cost of any mining algorithm. In [22] it is proposed a framework that uses Principal Component Analysis (PCA) to project the original data and it is showed that different changes in the original data variables (changes in mean, variance and correlation between features) can be observed on one or several principal components (PCs), which are orthogonal by their construction. PCA, which was first presented in 1901 by K. Pearson [21], is a remarkable and versatile technique that brings a touch of elegance in data analysis, capturing somehow the essence of variability within the datasets, unveiling latent connections between variables. However, it can be computationally expensive for large datasets, and in any case requires additional processing steps. We will describe this technique in details. In general, there exists many dimensionality reduction solutions; they could be divided into data-dependent (learned, such as PCA) and data-independent (such as random projections). Data-independent techniques might seem appropriate for evolving datastreams since they generate the projection matrices and transform data into a lower dimensional space independently from the input data, they do not suffer from the scalability problem, they are faster and less sensitive to unseen instances. However, in some domains those could be outperformed by learned feature extraction. Some techniques are meant to preserve the global relationships between data points (e.g. PCA selects the orthogonal directions explaining the variance of the whole dataset), some other preserve local relationships (e.g. hashing techniques transform the original data into a lower dimensional feature space where similar data points are likely to be mapped to nearby hash codes). We will discuss several techniques, their accuracy, processing time and memory usage, focusing on our concept drift detection task.

Curse of dimensionality: what happens if we go through higher and higher dimensional spaces? The complexity magnifies, and our ability to perceive structured clusters is lost or at least intricate. In machine learning and statistics, when referring to various phenomena that arise when working in high-dimensional spaces, we talk about the curse of dimensionality. When the dimensionalty d increases, the volume of the space increases so fast that the available data become sparse: volume size grows exponentially with d, we need 100 cm to fill a meter, 10.000 cm^2 to fill a square of 1 cm^2, and 1 million cm^3 to fill a cube with edges 1 m. Thus, in order to keep a constant "amount of knowledge" of the space, we need more data points when adding dimensions. Another interesting phenomenon arising with increasing dimensionality is that points are further from the center. As Yann Dubois write in his Github page *the volume of a high dimensional orange is mostly in its skin and not in the pulp, which means expensive high dimensional juice.* As he explains, considering a d-dimensional unit orange ($r = 1$) with skin of width $\epsilon = 0.05$, and knowing that the volume of a hypersphere is proportional to r^d ($V_d(r) = kr^d$), we have that:

$$\frac{V_{skin}}{V_{orange}} = \frac{V_{orange} - V_{pulp}}{V_{orange}} = \frac{kr^d - k(1-\epsilon)^d}{kr^d} = 1 - (1-\epsilon)^d$$

which implies that the skin will take, approximately, 10% of an orange 2D slice, 15% of a 3D orange, 40% of a 10D hyper-orange... an easier way to think about that is a $[-1, 1]^d$ hypercube: the distance from the origin to the center of each face equals 1, while the distance from the origin to each corner equals $\sqrt{d}$.

We must consider the point of view of the distance metrics, which are used to measure the similarity between objects, thus are of fundamental importance in this discussion. In *On the Surprising Behavior of Distance Metrics in High Dimensional Space* [1] it is examined the behavior of l_p norms and shown that the problem of meaningfulness in high dimensionality is sensitive to the value of p. It is explained that, under some assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target in high dimensional space is almost 1 for a wide variety of data distributions and distance functions: the nearest neighbor problem here is ill defined. Specifically, it is shown that the l_1 distance metric (Manhattan Distance metric) is the most preferable for high dimensional applications, followed by the euclidean distance (l_2 metric), the l_3 metric, ... knowing this, the authors studied fractional distance metrics (where $p < 1$) and showed that indeed these are even more effective at preserving the meaningfulness of proximity measures.

1.2.2 Few datapoints available

The other face of having to keep a constant "amount of knowledge" is working with a limited number of datapoints. A small training set (TR) impacts the robustness and generalizability of models: in our case, when attempting to fit the stationary distribution ϕ_0 from which the data are sampled, a small $|TR| = N$ may result in an inadequate coverage of the feature space; the estimation $\hat{\phi}_0$ relies on the assumption that the samples available are representative of the true underlying distribution.

The scarcity of training points becomes more pronounced when in conjunction with the challenges posed by the curse of dimensionality. To better quantify our results we will consider the N/d **ratio**, where d is the dimension of the feature space. A lower ratio is associated with sparse training data, impeding models' ability to capture patterns effectively.

1.2.3 Real-Time Processing

The unbound nature of evolving data streams poses technical and practical limitations that traditional stream algorithms often fail to address, often due to the high resource usage, such as time and memory, required to process this dynamic ever-growing size data incoming at high speed. Online algorithms must process incoming observations rapidly to keep up with the pace of data streams. Unlike processing static datasets, analyzing data from data streams cannot rely on multiple passes since the stream is unbounded. Algorithms must process each instance from the stream only once and use it to incrementally update the model or statistical information about the data. In some cases, batch-incremental algorithms process a batch or chunk of instances at once instead of processing them individually. Moreover, due to the massive amount of data in streams, it is impractical, if not impossible, to store the entire stream in memory. Stream algorithms must be able to operate under restricted memory constraints by storing only a few synopses of the processed data and the current model(s). A scalable algorithm for data streams can effectively process and analyze large volumes of incoming workload in real-time or near real-time, regardless of the data stream's size or arrival rate. It can adapt to the changing characteristics of the data stream without significant degradation in performance.

1.2.4 False Alarms control

The False Positive Rate (FPR) is a pivotal metric in statistical hypothesis testing. It represents the probability of rejecting a true null hypothesis, erroneously indicating the presence of a change or effect when there is none. In the context of change detection algorithms, a controlled FPR is crucial to prevent spurious alerts. It is worth noting that the "false positives" are alarms raised by data points sampled from the stationary distribution, the same from which the training points are take: FPR computation does not involve the post-change distribution ϕ_1, as seen in Eq. 1.8.

$$TPR = \frac{\text{\# post-change batches triggering an alarm}}{\text{\# post-change batches}} * 100 \tag{1.7}$$

$$FPR = \frac{\text{\# pre-change batches triggering an alarm}}{\text{\# pre-change batches}} * 100 \tag{1.8}$$

We remind that an analogous discussion is valid in the online streaming case when talking about the Average Run Length (ARL_0) defined in Eq. 1.4. These metrics are critical in evaluating monitoring systems. They are, indeed, synonyms of the trustworthiness of an alert, measuring the proportion of instances where the system incorrectly raises an alert, or signals an anomaly, when there is none. An uncontrolled FPR can results in unnecessary allocation of resources to investigate the alarms, leading to increasing cost. Imagine a manufacturing plant where some machine is producing high-quality silicon wafers. The machine's performance is monitored using sensors measuring parameters such as temperature, pressure, vibrations. A concept drift detection system may be implemented to identify deviations from the normal process and trigger alarms to prevent potential defects in the silicon wafers. With an effective concept drift detection system maintaining a low FPR, alarms are only triggered when there is some real deviation, indicating a potential issue with the wafer production process. This ensures that maintainance actions are taken only when necessary, optimizing the allocation of resources. A high FPR instead would result in alarms triggered when the machine is operating within acceptable parameters. Given that each false alarm prompts manual inspections or interventions, this would lead to unnecessary production stops and increased operational costs. A way more powerful example is given from healthcare: a false positive diagnosis (indicating a medical condition that the patient does not have) will lead to the unnecessary prescription of pharmaceuticals, which not only means additional healthcare costs, but also exposing patients to the risks and side effects of unnecessary medications. People would generally prefer not to know they have cancer, if there were high chances of an incorrect diagnosis: to start treatment unnecessarily, as the psychological and physical toll of undergoing treatment for a non-existent condition can be imagined, would be worse. And the prescription of an unnecessary antibiotic entails all the risks of side effects and the hundreds of millions of euros needlessly spent by healthcare systems; above all, each unnecessarily prescribed antibiotic contributes to the emergence of multi-resistant bacteria. In this kind of situations, FPR can be more critical than the True Positive Rate (TPR). For the patient's safety, for the resource utilization, etc., we built a diagnostic system which works when can actually identify medical conditions (high TPR) and can be trusted (FPR is controlled). Again, achieving a balance between these two metric is crucial for the overall effectiveness of a diagnostic system.

1.3 Related Work

Most change-detection algorithms in the literature are designed to monitor univariate datastreams. The vast majority of these methdos cannot be extendend to monitor multivariate frames, especially those leveraging nonparametric statistics. Change detection in multivariate datastreams has often been addressed in multi-stream settings, i.e., by separately analyzing each input component([6],[10],[25]). However, the hypotheses underpinning the multi-stream monitoring are fundamentally different: in particular, they often assume the components of the data points are generated by separate random variables rather than a single multivariate random vector. In multi-stream settings changes typically affect the distribution of a subset of these random variables, while in multivariate settings more general kinds of distribution changes are admissible; changes affecting the correlation between components or subtle changes involving the whole vector can be hard to detect by separately analyzing the components. A popular semiparametric approach consists of reducing the data dimensionality by monitoring the log-likelihood of the observations with respect to a density model fitted on a training set. The most common models for the initial distribution ϕ_0 are Gaussian, and Gaussian Mixture Models (GMMs). In [15], the Semiparametric Log-Likelihood

(SPLL) algorithm fits a GMM to the training set (TR) and compares incoming batches with batches from TR by a likelihood test. The main limitation of these solutions are that:

- The implicit assumption that ϕ_0 can be approximated well by a probability distribution from a known family, which is not guaranteed in general.
- None of the commonly proposed methods can be configured before deployment to operate at a target False Positive Rate (FPR) or Average Run Length (ARL_0).

Indeed, in SPLL, the adoption of a GMM to approximate ϕ_0 might not always fit real-world data; moreover, it is not possible to set a priori the detection threshold to control FPR, as the distribution of the test statistic depends on ϕ_0.

There are only a few recent multivariate methods that perform non-parametric change detection, namely, that assume ϕ_0 and ϕ_1 to be unknown. Among these, we focus on histogram-based algorithms, since these are non-parametric by design; the extension to high-dimensional multivariate datastreams is usually infeasible since number of bins scales exponentially with data dimension, but this is not always the case. Equal Intensity K-means (EIKM) [19] divides the input space using K-means clustering, resulting in bins that yield an equal probability under ϕ_0. EIKM is designed to handle multimodal distributions, e.g., Gaussian Mixtures, and the detection thresholds are given by asymptotic approximations of the Pearson test statistic. QuantTree [6] defines a partitioning S of the input space in a fixed number K of bins by axis-aligned cuts. The theoretical properties of QT guarantee that the distribution of test statistics defined over bin probabilities does not depend on ϕ_0, which allows to set detection thresholds a priori, with synthetically generated data through a very efficient scheme. Since the data splits are limited to the axis directions, the bins in QT require a preprocessing stage whose outcome, in terms of detection power, is uncertain, as demonstrated in our experiments. KQT [25] preserves the properties of QT in terms of setting detection thresholds and FPR control, and overcomes QT limitation by constructing compact bins that are not affected by roto-translations, thus better approximate the probability measure of each bin under ϕ_0. However, KQT based on Mahalanobis and Weighted Mahalanobis distances, which rely on the sample covariance matrix estimated from the training set, lose control over false positives when the ratio N/d between the number N of training points and the dimensionality of the dataset d decrease.

Histograms are commonly used not only in the offline fashion, where multiple passes over data are allowed; in 2002 were proposed [11] several incremental techniques to handle data streams, which failed in some cases where data distribution is not uniform. QT was extendend to QT-EWMA proving that it is possible to use a very efficient Monte Carlo scheme to compute thresholds [10]; its implementation comes together with theoretically procedures to extend a generic one-shot detector to monitor datastreams controlling the ARL_0.

Chapter 2

QuantTree

2.1 QuantTree algorithm

The QuantTree algorithm was first proposed in 2018 [6] to handle concept drift detection in multivariate dataframes. This paper begins stating that *histograms are very general and flexible models, which have been relatively ignored in the change-detection literature as they often require a number of bins that grows unfeasibly with the data dimension.* QuantTree is a recursive binary splitting scheme designed to dynamically adapt histogram bins for effective change detection. The greatest advantage with QT is that the distribution of any statistic defined over the resulting histogram does not depend on ϕ_0, i.e. that decision rules to be used do not depend on the data and can be numerically computed from synthetically generated univariate sequences, even in multivariate change detection problems. The fact that QT can have a pre-assigned number of bins and can be represented as a tree, enabling a very efficient computation of test statistics.

QuantTree iteratively divides the input space by employing binary splits on a single covariate, with cutting points defined by the quantiles of the marginal distributions. This strategy shares similarities with kd-trees [5], where splits are based on the median value of the marginal. The simplicity of this construction scheme enables analytical handling, e.g. it is proven that the distribution of each bin probability does not depend on ϕ_0. Experimental results show QT's capability to deliver robust detection performance in high-dimensional streams. Notably, when confronted with limited training samples, QT ensures superior false positive rate control compared to established methods like the Pearson goodness-of-fit test and tests relying on empirical thresholds computed e.g. through bootstrap techniques.

```
Algorithm 1: QuantTree
Input: Training set $TR$ containing $N$ stationary points in $\mathcal{X}$; number of bins $K$; target probabilities $\{\pi_k\}_k$.
Output: The histogram $h = \{(S_k, \widehat{\pi}_k)\}_k$.
1: Set $N_0 = N, L_0 = 0$.
2: For $k = 1, \dots, K$ do
3:     Set $N_k = N_{k-1} - L_{k-1}, \mathcal{X}_k = \mathcal{X} \setminus \bigcup_{j<k} S_j$, and $L_k = \text{round}(\pi^k N)$.
4:     Choose a random component $i \in \{1, \dots, d\}$.
5:     Define $z_n = [x_n]_i$ for each $x_n \in \mathcal{X}_k$.
6:     Sort $\{z_n\} : z_{(1)} \leq z_{(2)} \leq \dots z_{(N_k)}$.
7:     Draw $\gamma \in \{0, 1\}$ from a Bernoulli(0.5).
8:     If $\gamma = 0$ then:
9:         Define $S_k = \{x \in \mathcal{X}_k \ [x]_i \leq z_{(L_k)}\}$.
10:    Else:
11:        Define $S_k = \{x \in \mathcal{X}_k \ [x]_i \geq z_{(N_k - L_k + 1)}\}$.
12:    End if
13:    Set $\widehat{\pi}_k = L_k / N$.
14: End for
```

Table 2.1: The algorithm is built to define histograms h through a recursive binary splitting of the input space $\mathcal{X}$. It takes as input a training set TR containing N stationary points, the number of bins K in the histogram, and the target probabilities on each bin $\{\pi_k\}_{k=1,\dots,K}$, and returns a histogram $h = \{(S_k, \widehat{\pi}_k)\}_{k=1,\dots,K}$, where each $\widehat{\pi}_k$ represents an estimate of the probability for a sample drawn from ϕ_0 to fall in S_k.

The algorithm presents in detail the iterative formulation of QuantTree, which constructs a new bin of the histogram h at each step k. We denote by $\mathcal{X}_k \subseteq \mathcal{X}$ the subset of the input space that still has to be partitioned (i.e., $\mathcal{X}_k = \mathcal{X} \setminus \bigcup_{j<k} S_k$) and by N_k the number of points of TR belonging to $\mathcal{X}_k$. We compute (line 3) the number of training points that has to fall inside S_k as $L_k = \text{round}\,(\pi_k N)$. The subset S_k is then defined by splitting $\mathcal{X}_k$ along a component $i \in \{1, \dots, d\}$ that is randomly chosen with uniform probability (line 4). The splitting point is defined by sorting $z_n = [\mathbf{x}_n]_i$, i.e., the values of the i-th component for each $\mathbf{x}_n \in \mathcal{X}_k$ (lines 5). We thus obtain $z_{(1)} \leq z_{(2)} \leq \dots \leq z_{(N_k)}$ (line 6) and we define S_k by splitting $\mathcal{X}_k$ w.r.t. $z_{(L_k)}$ or $z_{(N_k - L_k + 1)}$ (lines 7-11). In both cases S_k contains L_k points among the N in $\mathcal{X}$, thus the estimated probability of S_k is $\widehat{\pi}_k = L_k/N$ (line 13). This procedure is iterated until K subsets are extracted. QuantTree divides $\mathcal{X}$ in a given number of subsets, where each S_k has an estimated probability $\hat{\pi}_k \approx \pi_k$, and the equality holds when $\pi_k N$ is integer. Since the probabilities π_k are set a priori, in what follows we use π_k in place of $\hat{\pi}_k$. Indexes i (the choice of the component) and parameter γ (the choice of the quantile) are randomly chosen at each iteration to add variability to the histogram construction.

A key feature of a histogram computed by QuantTree is that any statistic $\mathcal{T}_h$ built over it has a distribution that is independent from ϕ_0. This results come from the following theorem, which is proved in [6].

Theorem 1. *Let $\mathcal{T}_h$ be defined over the histogram h computed by QuantTree such that it uniquely depends on $\{y_k\}_{k=1,\dots,K}$, where y_k denotes the number of samples in W falling in S_k. When W ϕ, the distribution of $\mathcal{T}_h(W)$ depends only on ν, N and $\{\pi_k\}_k$.*

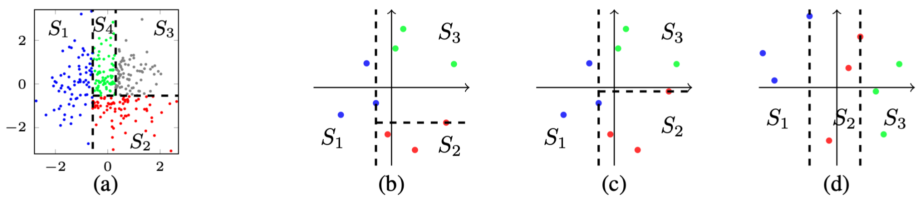


Figure 2.1: (a) A histogram $\{S_k\}_{k=1,\dots,4}$ computed by QuantTree to yield uniform density on the bins. (b)-(d) Examples of values assumed by $\tilde{L_k}$, i.e. the position of the splitting point, in three different configurations, when $N = 9$ and $L1 = L2 = L3 = 3$. In these cases $\tilde{L_1} = L_1$, while in (b) $\tilde{L_2} = 3$, in (c) $\tilde{L_2} = 5$, and in (d) $\tilde{L_2} = 6$. Note that when QuantTree chooses always the same component, we have that $\tilde{L_2} = L1 + L2$, as in (d).

Theorem 1 implies that we can numerically compute the thresholds for any statistic $\mathcal{T}_h$ defined on histograms, provided ν, N and $\{\pi_k\}$, thus disregarding ϕ_0 and the data dimension d. To this end, we synthetically generate data from a conveniently chosen distribution ψ_0, and we follow the procedure outlined in [6] and reported here as algorithm 2 to estimate the threshold τ for the hypothesis test 1.1, yielding a desired FPR α.

Algorithm 2: Numerical procedure to compute thresholds

```
Input: Test statistic $\mathcal{T}_h$; arbitrarily chosen $\psi_0$; the number $B$ of datasets and
batches to compute the threshold; the number of points $\nu$ in each batch;
$N, K$, and $\widehat{\pi}_k$ as in Algorithm 1; the desired FPR $\alpha$.
Output: The value $\tau$ of the threshold
1: for $b = 1, \dots, B$ do
2: Draw from $\psi_0$ a training set $TR_b$ of $N$ samples.
3: Use QuantTree to compute the histogram $h_b$ with
   $K$ bins and target probabilities $\{\pi_k\}_k$ over $TR$.
4: Draw a batch $W_b$ containing $\nu$ points from $\phi_0$.
5: Compute the value $t_b = \mathcal{T}_h(W)$.
6: end for
7: Compute the threshold $\tau$ as in (5).
```

At first we generate B training sets $\{TR_b\}_{b=1,\dots,B}$, sampling N points from ψ_0 and, for each training set, we build a histogram h_b using QuantTree. Then, for each h_b we generate a batch W_b of ν points drawn from ψ_0, and compute the value of the statistic $t_b = \mathcal{T}_h(W_b)$ (lines 4-5). Finally, we estimate τ (line 7) from the set $T_B = \{t_1, \dots, t_B\}$ as the $1-\alpha$ quantile of the empirical distribution of $\mathcal{T}_h$ over the generated batches, i.e.

$$\tau = \min \{t \in T_B : \# \{v \in T_B : v > t\} \le \alpha B\},$$

where $\#A$ denotes the cardinality of a set A. To take full advantage of the distribution-free nature of the procedure, we set ψ_0 to a univariate uniform distribution $U(0,1)$. This allows to obtain high accuracy on the estimation of the thresholds, since we can use very large values of B with limited computational cost. In this work, we employed the Pearson statistic:

$$\mathcal{T}_h^P(W) = \sum_{k=1}^{K} \frac{(y_k - \nu\pi_k)^2}{\nu\pi_k} \tag{2.1}$$

It is well known that, when $\{\pi_k\}_k$ are the true probabilities of the bins $\{S_k\}_k$, the statistic $\mathcal{T}_h^P(W)$ is asymptotically distributed as a χ^2_{K-1} under the null hypothesis. However, when the π_k are estimated, the threshold obtained from the χ^2_{K-1} distribution does not allow to properly control the False Positive Rate (FPR); this effect is even more evident when y_k is small. In contrast, thresholds defined by Algorithm 2 hold also in case of limited sample size, since they are not based on an asymptotic result.

2.2 Kernel-QuantTree

Kernel QuantTree (KQT) was first presented in the 2021-22 thesis work of Paolo Rizzo, then in [25]. It is a histogram based change detection method that extends QT. A fundamental limitation of QT is that splits are defined along the axis, resulting in a partitioning that does not always adhere to the input distribution. The cuts are oriented along axis, leading to bins with hyper-rectangular shapes; bins also contains large empty holes where there are no points, leading to blind spaces where changes are not detected. To mitigate this problem, a preprocessing stage is typically introduced to align the split directions to the principal components of the training set. While this procedure is often beneficial, it was observed that it can worsen the detection performance in some unpredictable cases [25]. We will show and discuss how PCA is incompatible with FPR control in high dimensional spaces when a few training points are provided. Moreover, many bins in a QT histogram have non-finite volumes, which can lead to poor estimation of bin probabilities.

In [25] is thus described Kernel QuantTree (KQT), a non-parametric and multivariate CD algorithm that constructs histogram bins via measurable kernel functions, resulting in a powerful concept drift detection test. In contrast with the QT algorithm, which constructs bins by axis-aligned splits that can be unlimited along some dimensions, KQT partitions the space in $K-1$ compact bins defined by kernel functions evaluated on the training data. An additional bin, denoted as the residual bin, is non-compact and gathers all the points that do not fall in any other bin. Figure 2.2 shows that the KQT bins are compact subsets of the domain. Among various approaches, the method that seemed most suitable was the one based on centroids: the idea is to follow QT's approach, building a completely unbalanced binary tree by slicing the data to obtain a histogram with target probabilities $\{\pi_k\}_k$. For each node, we choose a centroid from the residual training data $\mathcal{X}_k$, and compute the distances between the centroid and the data. This allows to select the closest points with respect

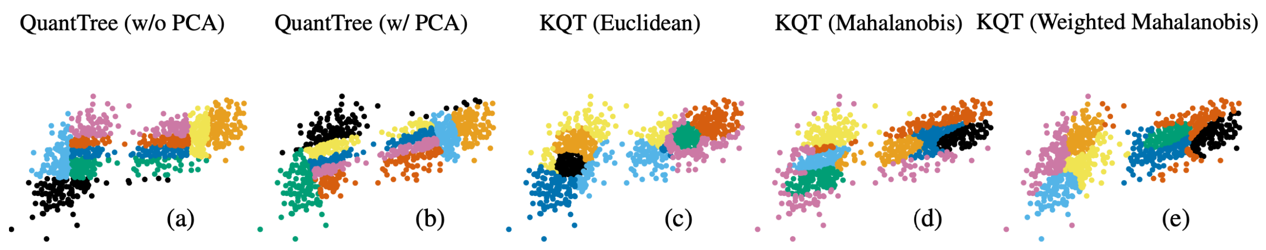


Figure 2.2: QuantTree generates bins as intersection of hyperplanes, performing cuts along the axis (a). After a preprocessing through PCA, the cuts are oriented along the principal directions (b). Kernel QuantTree generates bins that are subsets of d-dimensional spheres according to the underlying kernel functions, namely the Euclidean (c), Mahalanobis (d) and Weighted Mahalanobis (e) distances.

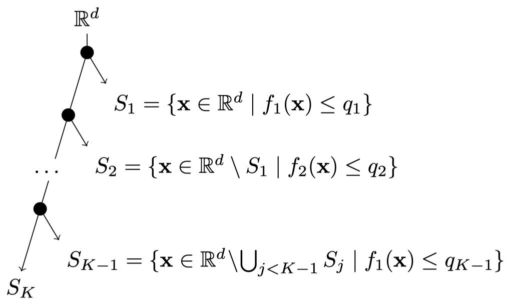
to the centroid using a quantile that is based on how many points we want far from that bin. Iteratively, we remove the selected points and continue until we have all K bins.

Again, the distribution of the test statistic $\mathcal{T}_h$ computed from a KQT histogram h does not depend on the stationary distribution ϕ_0. Consequently, detection thresholds τ can be set a priori via Monte Carlo simulations as in QT, without knowing ϕ_0. Moreover, the monitoring performed by KQT using specific kernel functions is not influenced by preprocessing based on roto-translations, including alignment to principal components. Thanks to these properties, KQT outperforms state-of-the-art alternatives on a broad experimental testbed illustrated in the paper. In particular, KQT achieves better detection performance than the alternatives independently of preprocessing steps based on roto-translations. However, we'll see that if the kernel function used is based on the training set (e.g. with Mahalanobis or Weighted Mahalanobis distances), an inconvenient ratio N/d between its cardinality and the dimensionality might ruin FPR control.

Algorithm 3: Construction of the GQT histogram

```
1: **Input**: training set $TR = \{\mathbf{x}_i\}_{i=1}^{N} \subset \mathbb{R}^d$, target probabilities $\{\pi_k\}_{k=1}^{K}$
2: **Output**: GQT histogram $h = \{(S_k, \widehat{\pi}_k)\}_{k=1}^{K}$
3: Set $\mathcal{X}_0 = \text{TR}$
4: for $k = 1, \dots, K-1$ do
5:     Compute $\widetilde{\pi}_k = \pi_k \left(1 - \sum_{j<k} \pi_j\right)^{-1}$
6:     Compute $\{f_k(\mathbf{x}_i)\}$ for $\mathbf{x}_i \in \mathcal{X}_{k-1}$
7:     Set $q_k$ as the $\widetilde{\pi}_k$-quantile of $\{f_k(\mathbf{x}_i)\}$
8:     $S_k = \left\{\mathbf{x} \in \bigcap_{j<k} \bar{S}_j \mid f_k(\mathbf{x}) \leq q_k\right\}$
9:     $\mathcal{X}_k = \{\mathbf{x} \in \mathcal{X}_{k-1} \mid f_k(\mathbf{x}) > q_k\}$
10: end for
11: $S_K = \mathbb{R}^d \setminus \bigcup_{j<K} S_j$
```

The KQT histogram is constructed by iteratively splitting the input space $\mathbb{R}^d$ into K bins $\{S_k\}$ such that the probability of a stationary sample $\mathbf{x} \sim \phi_0$ to fall in S_k is close to a target probability π_k, which are provided as input parameters. The peculiarity of KQT is that each bin S_k for $k < K$ is defined by a measurable kernel function $f_k : \mathbb{R}^d \to \mathbb{R}$ and a split value $q_k \in \mathbb{R}$, and corresponds to a compact set in $\mathbb{R}^d$. We denote as Generalized QuantTree (GQT) partitioning the resulting histogram $h = \{(S_k, \widehat{\pi}_k)\}_{k=1}^K$, which yields a partition of the input space $\mathbb{R}^d$, where $\widehat{\pi}_k$ is the empirical probability of $\mathbf{x} \sim \phi_0$ to fall in S_k.



$\mathbb{R}^d$
$S_1 = \{\mathbf{x} \in \mathbb{R}^d \mid f_1(\mathbf{x}) \leq q_1\}$
$S_2 = \{\mathbf{x} \in \mathbb{R}^d \setminus S_1 \mid f_2(\mathbf{x}) \leq q_2\}$
$\dots$
$S_{K-1} = \{\mathbf{x} \in \mathbb{R}^d \setminus \bigcup_{j<K-1} S_j \mid f_1(\mathbf{x}) \leq q_{K-1}\}$
S_K

Figure 2.3: The Generalized QuantTree histogram is a binary splitting tree where splits isolate leaves, i.e. bins of the histogram.

As illustrated in Figure 2.3, a GQT partitioning corresponds to an extremely imbalanced binary tree, where each split isolates a leaf, corresponding to a bin S_k. In [25] it is proven that the distribution of any test statistic $\mathcal{T}_h$ defined over a GQT partitioning does not depend on ϕ_0, extending the theoretical results from QT. This property enables setting the detection threshold τ a priori by Monte Carlo simulations. It is also proved that under some mild assumption on the kernel function f_k, this monitoring scheme becomes independent of any roto-translation applied to the data, including the PCA preprocessing. The scheme by KQT operates as follows: given an input batch W containing ν test samples, we compute the test statistic $\mathcal{T}_h(W)$ and detect changes when this exceeds the threshold τ. While the theoretical properties of KQT hold for all the statistics that only depend on $\{y_k\}$, the numbers of samples in W falling in bins $\{S_k\}$, we consider the Pearson χ^2 statistic:

$$\mathcal{T}_h(W) = \mathcal{T}_h\left(y_1, y_2, \dots, y_K\right) = \sum_{k=1}^{K} \frac{\left(y_k - \nu\pi_k\right)^2}{\nu\pi_k}$$

where $\{\pi_k\}$ are the target bin probabilities. The GQT extends the partitioning scheme underpinning QT, which corresponds to using linear split functions:

$$f_k(\mathbf{x}) = \pm 1 \cdot P_j \mathbf{x},$$

where P_j is the projection over a randomly selected component j and ± 1 randomly introduces a sign flip for the projection.

The criteria to select the centroids $\{\mathbf{c}_k\}$ from TR is key in KQT, as this determines both the spatial location of the bin S_k and the split value q_k associated with the kernel function f_k. For the experiments that will follow, the selection strategy adopted is maximizing the *information gain* associated with the split; measuring the decrease in the overall entropy H after a split in the data, it is typically used to assess the split quality. In particular, we will discuss the problem arising from the computation of the entropy $H(B)$ of a set of points B using the Gaussian approximation:

$$H(B) = \frac{1}{2}\log((2\pi e)^d \det(\mathrm{cov}[B]) \tag{2.2}$$

Here, $\mathrm{cov}[B]$ represents the sample covariance matrix computed over B. It is worth noting that this determinant computation could cause issues when dealing with points in high dimensional spaces. For a deeper comprehension of the importance of centroid selection criteria and an explanation of the theoretical guarantees, one should refer to the paper [25].

In the next section, we present specific measurable functions f_k that we employ in KQT.

2.2.1 Employed Kernel Functions

We define the kernel functions $f_k : \mathbb{R}^d \to \mathbb{R}$ as distances from a centroid $\mathbf{c}_k \in$ TR, selected from the training set:

$$f_k(\mathbf{x}) = (\mathbf{x} - \mathbf{c}_k)^T A\,(\mathbf{x} - \mathbf{c}_k)\,,$$

where $A \in \mathbb{R}^{d\times d}$ is the kernel matrix, which induces a distance measure in $\mathbb{R}^d$. In particular, the bins $\{S_k\}$ are subsets of d-dimensional spheres centered in $\{\mathbf{c}_k\}$ and having radii $\{q_k\}$, where the distances are measured with respect to the metric induced by A. Since spheres in $\mathbb{R}^d$ are compact sets, all the bins S_k of a KQT, but the residual S_K, are compact and have a finite volume.

In [25], KQT is computed using the Euclidean, the Mahalanobis, and the Weighted Mahalanobis [26] distances; different distances result in bins with distinct shapes, as illustrated in Figure 2.2. We obtain the **Euclidean distance** by setting $A = \mathbb{I}_d$, namely, the d-dimensional identity matrix, resulting in isotropic bins. Figure 2.2 (c) shows that these bins poorly fit the data distribution, e.g. covering different modes of the distribution. We obtain the **Mahalanobis** distance by setting $A = \Sigma^{-1}$, where $\Sigma \in \mathbb{R}^{d\times d}$ is the sample covariance matrix of TR, and in this case, the bins are anisotropic. Figure 2.2 (d) also shows that bins are elongated towards the directions with larger variance, resulting in a better fit to the data. However, these bins poorly approximate TR when this exhibits multiple clusters, since again multiple bins might span different clusters. To promote bins containing samples from a single cluster, the **Weighted Mahalanobis** distance is adopted: a Gaussian Mixture of M components is fit to TR, then a larger distance is assigned to points that belong to different components of the GMM. The Weighted Mahalanobis kernel matrix is then defined as:

$$A(\mathbf{x}) = \frac{\sum_{m=1}^{M} \rho_m \cdot i_m(\mathbf{x},\mathbf{c}) \cdot C_m^{-1}}{\sum_{m=1}^{M} \rho_m \cdot i_m(\mathbf{x},\mathbf{c})},$$

where μ_m, C_m, and ρ_m denote the mean, covariance matrix, and mixing probability of the m-th Gaussian, respectively. The matrix A represents a weighted average of the inverse covariance matrices of the GMM components. As in [26] the weights are proportional to the mixing probabilities ρ_m and $i_m(\mathbf{x}, \mathbf{c})$, which is a computationally tractable approximation of the distance between the point $\mathbf{x}$ and the bin centroid $\mathbf{c}$.

Distances derived from l_p norms

As we will show in the following chapters, Mahalanobis and Weighted Mahalanobis distances, which are based on the sample covariance matrix, ruin the FPR control in frames where too few training points fill an high dimensional space. Also the Euclidean distance (l_p norm with $p = 2$) might not be the best choice when dealing with high-dimensional data because of the curse of dimensionality. We propose to derive distances from l_p quasi-norms, with $p \leq 1$ e.g., the Manhattan distance (norm with $p = 1$) and fractional distance metrics with $p < 1$. This choice was suggested in [1]: the authors study the behavior of l_p norms and show that the problem of meaningfulness in high dimensionality is sensitive to the value of p. Specifically, it is shown that the l_1 distance metric (Manhattan Distance metric) is the most preferable for high dimensional applications, followed by the euclidean distance (l_2 metric), the l_3 metric, ... finally, it is shown that fractional distances are even more effective at preserving the meaningfulness of proximity measures. However, we must notice that only the Euclidean distance (l_2 norm) is invariant under roto-translations, and KQT loses its invariance properties in general with these metrics.

2.3 QT-EWMA

QT-EWMA algorithm was first introduced in [10] together with the procedure to define its thresholds controlling the ARL_0. The algorithm leverages a novel online statistic T_t defined over a QuantTree histogram, which is constructed given the training set TR and K target probabilities $\{\pi_j\}_{j=1}^{K}$; it returns an histogram defined by K bins $\{S_j\}_{j=1}^{K}$, where each $S_j \subset \mathbb{R}^d$ is set to contain $\pi_j N$ training samples. The statistic T_t monitors the proportion of samples in the datastream that fall in each bin S_j. In particular, for each x_t we define K binary statistics $\{y_{j,t}\}_j$ as the indicator functions of each bin S_j, namely:

$$y_{j,t} = \mathbb{1}(x_t \in S_j), \quad j \in \{1, ..., K\} \tag{2.3}$$

to track in which bin the input sample x_t falls. It is possible to show that, when $x_t \sim \phi_0$ and TR $\sim \phi_0$ then:

$$\mathbb{E}\left[y_{j,t}\right] \approx \hat{\pi}_j := \frac{N\pi_j}{N+1}, \quad j < K \quad \text{ and } \quad \mathbb{E}\left[y_{K,t}\right] \approx \hat{\pi}_K := \frac{N\pi_K + 1}{N+1} \tag{2.4}$$

The statistics $y_{j,t}$ are evaluated for each incoming sample x_t and then the EWMA statistic $Z_{j,t}$, $j \in \{1, ..., K\}$ is computed to monitor the proportion of data that falls in each bin S_j:

$$Z_{j,t} = (1-\lambda)Z_{j,t-1} + \lambda y_{j,t} \quad \text{where} \quad Z_{j,0} = \hat{\pi}_j \tag{2.5}$$

Since, under ϕ_0, the expected value $\mathbb{E}[Z_{j,t}] \approx \hat{\pi}_j$ for $j = 1, ..., K$, the QT-EWMA change-detection statistic is defined as follows:

$$T_t = \sum_{j=1}^{K} \frac{(Z_{j,t} - \hat{\pi}_j)^2}{\hat{\pi}_j} \tag{2.6}$$

Similarly to the Pearson statistic, T_t measures the overall difference between the proportion of points in each bin S_j, represented by $Z_{j,t}$, and their approximated expected values $\hat{\pi}_j$ under ϕ_0. This difference naturally increases as a consequence of a change $\phi_0 \rightarrow \phi_1$ that modifies the probability of some bin S_j. The statistic is computed at each incoming sample and then compared against the corresponding threshold h_t to detect changes. The overall procedure is shown in Algorithm 4.

```
Algorithm 4: QT-EWMA
Input: datastream $x_1$, $x_2$, ..., target $\{\pi_j\}_{j=1}^K$, threshold $\{h_t\}_t$, TR
Output: detection flag ChangeDetected, detection time $t^*$
1: ChangeDetected ← False, $t^* \leftarrow \infty$;
2: estimate QT histogram $\{S_j, \pi_j\}_{j=1}^K$ from TR and define $\{\hat{\pi}_j\}_{j=1}^K$ as in 2.4;
3: $Z_{j,0} \leftarrow \hat{\pi}_j \forall j = 1, \ldots, K$;
4: for $t = 1, \ldots$ do:
5:     $y_{j,t} \leftarrow \mathbb{1}(x_t \in S_j)$
6:     $Z_{j,t} \leftarrow (1-\lambda)Z_{j,t-1} + \lambda y_{j,t}, \quad j = 1 \ldots, K$;
7:     $T_t \leftarrow \sum_{j=1}^K (Z_{j,t} - \hat{\pi}_j)^2 / \hat{\pi}_j$;
8:     if $T_t > h_t$ then
9:         ChangeDetected ← True,    $t^* \leftarrow t$;
10:        break
11:    end
12: end
13: return ChangeDetected, $t^*$
```

Table 2.2: QT is constructed from the training set TR and K target probabilities (line 2); defined statistics $y_{j,t}$ are evaluated for each incoming sample x_t (line 5), and then EWMA statistic $Z_{j,t}$ is computed (line 6) and compared against the corresponding threshold h_t to detect changes (line 9).

QT-EWMA algorithm inherits from QuantTree the fundamental property that the distribution of the statistics 2.5 and 2.6 - like any other statistic entirely defined over QuantTree bins - does not depend on ϕ_0, so the thresholds $\{h_t\}_t$ can be defined a priori to guarantee the ARL_0 on any datastream. The sequence of thresholds has to be properly defined to guarantee the given $ARL_0 = \mathbb{E}[t^*]$, where the expected value is computed assuming that the whole datastream is drawn from ϕ_0. We set the thresholds to guarantee a constant false alarm probability α at each time instant t. When this is satisfied, the detection time t^* is a Geometric random variable with parameter α and expected value given by:

$$ARL_0 = \mathbb{E}_{\phi_0}[t^*] = \frac{1}{\alpha} \tag{2.7}$$

To guarantee the constant false alarm probability, the thresholds must satisfy the following equation:

$$\mathbb{P}(T_t > h_t | T_k \leq h_k \forall k < t) = \alpha \quad \forall t \geq 1 \tag{2.8}$$

Since it is infeasible to exactly compute the conditional probabilities in 2.8, we resort to Monte Carlo simulations. Since thresholds do not depend on ϕ_0, we can conveniently generate 1-dimensional Gaussian streams and perform MC simulations to compute $\{h_t\}_t$ efficiently. In particular, we generate $1,000,000$ training sets of $N = 4096$ normal realizations $x_t \sim \mathcal{N}(0,1)$. For each TR we construct a QuantTree histogram and then generate 5000 samples from $\mathcal{N}(0,1)$ that we use to compute the statistics $\{T_t\}_{t=1}^{5000}$. Then, we define the threshold h_1 yielding the target ARL_0 as the empirical $(1-\alpha)$-quantile of T_1 values, bearing in mind that $\alpha = 1/\text{ARL}_0$. Similarly, all the thresholds h_t are computed as the $(1-\alpha)$-quantiles of the values T_t, but when $t > 1$ we compute the empirical quantiles only considering those sequences whose statistic has never exceeded any of the previous thresholds $h_1, \dots, h_{t-1}$, namely having $T_k \leq h_k, \forall k < t$. Thresholds $\{h_t\}_t$ computed in this way guarantee 2.8 to hold, so the target ARL_0 is preserved. We compute all the thresholds $\{h_t\}_{t=1}^{5000}$ and then fit a polynomial to these values. This allows both to estimate h_t for $t > 5000$ and to improve the estimates $\{h_t\}_{t=1}^{5000}$ by leveraging correlation among thresholds. In particular, we estimate a polynomial in powers of $1/t$ that returns h_t for a given t. In our experiments we employ the following target ARL_0 values: 500, 1000, 2000, 5000. An important consequence of setting a constant false alarm probability in 2.8 is that, being t^* a Geometric random variable with parameter α, the probability of having a false alarm before t by the geometric sum:

$$\mathbb{P}(t^* \leq t) = \sum_{k=1}^{t} \alpha(1-\alpha)^{k-1} = \alpha \cdot \frac{1-(1-\alpha)^t}{\alpha} = 1-(1-\alpha)^t \tag{2.9}$$

where the probability $\mathbb{P}$ is computed under ϕ_0.

2.4 KQT-EWMA

We propose Kernel-QuantTree Exponentially Weighted Moving Average (KQT-EWMA), a novel online nonparametric change-detection algorithm for multivariate datastreams. It combines a generalized QT histogram [25], used as a model $\hat{\phi}_0$, and a novel statistic T_t based on Exponentially Weighted Moving Average.

The theoretical properties of Kernel-QuantTree guarantee that KQT-EWMA is completely nonparametric since the distribution of our statistic does not depend on ϕ_0, hence its thresholds $\{h_t\}_t$ controlling the ARL_0 can be set a priori. Moreover, these thresholds guarantee by design a constant false alarm probability over time and, consequently, a fixed false alarm rate at any time instant during monitoring. Thus, kQT-EWMA controls both ARL_0 and false alarm (FA) rate. The distribution of the test statistic computed over stationary data is independent of ϕ_0, and this can be exploited to compute detection thresholds by Monte Carlo simulations such that the empirical ARL_0 matches any target value. Additional material, with theoretical properties and proofs, can be found in [25] and in Chapter 3. The algorithm leverages a novel online statistic T_t defined over a Generalized QuantTree histogram, which is constructed given the training set TR and K target probabilities $\{\pi_j\}_{j=1}^{K}$. The procedure for the construction of the GQT was shown in Algorithm 3. The statistic T_t monitors the proportion of samples in the datastream that fall in each bin S_j. In particular, for each x_t we define K binary statistics $\{y_{j,t}\}_j$ as the indicator functions of each bin S_j. We evaluate the statistics as in Equations 2.3, 2.4, and 2.5; also in this case, we evaluate the statistics $y_{j,t}$ for each incoming sample x_t and then we compute the EWMA statistic $Z_{j,t}$, $j \in \{1, ..., K\}$, to monitor the proportion of data that falls in each bin S_j. Since, under ϕ_0, the expected value $\mathbb{E}[Z_{j,t}] \approx \hat{\pi}_j$ for $j = 1, ..., K$, we define the kQT-EWMA change-detection statistic as before (Eq. 2.6):

$$T_t = \sum_{j=1}^{K} \frac{(Z_{j,t} - \hat{\pi}_j)^2}{\hat{\pi}_j} \tag{2.10}$$

Similarly to the Pearson statistic, T_t measures the overall difference between the proportion of points in each bin S_j, represented by $Z_{j,t}$, and their approximated expected values $\hat{\pi}_j$ under ϕ_0. This difference naturally increases as a consequence of a change $\phi_0 \rightarrow \phi_1$ that modifies the probability of some bin S_j. The statistic is computed at each incoming sample and then compared against the corresponding threshold h_t to detect changes. The overall procedure is shown in Algorithm 5.

GQT-EWMA algorithm inherits from Generalized QuantTree the fundamental property that the distribution of the statistics 2.10 does not depend on ϕ_0, so the thresholds $\{h_t\}_t$ can be defined a priori to guarantee the ARL_0 on any datastream. The sequence of thresholds has to be properly defined to guarantee the given $ARL_0 = \mathbb{E}[t^*]$, where the expected value is computed assuming that the whole datastream is drawn from ϕ_0. We set the thresholds to guarantee a constant false alarm probability α at each time instant t. When this is satisfied, the detection time t^* is a Geometric random variable with parameter α and expected value given by Eq.2.7: To guarantee the constant false alarm probability, the thresholds must satisfy Equation 2.8.

As described for QT-EWMA thresholds computation we resort to Monte Carlo simulations. Since thresholds do not depend on ϕ_0, we can conveniently generate 1-dimensional Gaussian streams and perform MC simulations to compute

```
Algorithm 5: GQT-EWMA
Input: datastream $x_1$, $x_2$, ..., target $\{\pi_j\}_{j=1}^K$, threshold $\{h_t\}_t$, TR
Output: detection flag ChangeDetected, detection time $t^*$
1: ChangeDetected $\leftarrow$ False, $t^* \leftarrow \infty$;
2: estimate GQT histogram $\{S_j, \pi_j\}_{j=1}^K$ from TR and define $\{\hat{\pi}_j\}_{j=1}^K$ as in 2.4;
3: $Z_{j,0} \leftarrow \hat{\pi}_j \forall j = 1, \dots, K$;
4: for $t = 1, \dots$ do:
5:    $y_{j,t} \leftarrow \mathbb{1}(x_t \in S_j)$
6:    $Z_{j,t} \leftarrow (1-\lambda) Z_{j,t-1} + \lambda y_{j,t}, \quad j = 1 \dots, K$;
7:    $T_t \leftarrow \sum_{j=1}^K (Z_{j,t} - \hat{\pi}_j)^2 / \hat{\pi}_j$;
8:    if $T_t > h_t$ then
9:       ChangeDetected $\leftarrow$ True,    $t^* \leftarrow t$;
10:      break
11:   end
12: end
13: return ChangeDetected, $t^*$
```

Table 2.3: GQT is constructed as in Algorithm 3 from the training set TR and K target probabilities (line 2); defined statistics $y_{j,t}$ are evaluated for each incoming sample x_t (line 5), and then EWMA statistic $Z_{j,t}$ is computed (line 6) and compared against the corresponding threshold h_t to detect changes (line 9).

$\{h_t\}_t$ efficiently. Moreover, the generated threshold estimated for QT-EWMA can be applied in this generalized case effectively.

2.5 Computational remarks

QuantTree

It is important to note that since the histogram h computed by QuantTree is exclusively defined on the marginal probabilities of single components, the dimensionality of the input data d does not impact the overall computational cost. In fact, the cost of building a QT is dominated by sorting the covariates (Algorithm 1 line 6), which is performed K times on a progressively smaller number of samples at each iteration. Therefore, the overall complexity of constructing a QuantTree is $\mathcal{O}(KNlogN)$. Since any histogram h computed by QT can be represented as a tree structure, it is very efficient to identify the bin where any testing point belongs to. In fact, during monitoring, at most K if-then operations have to be performed for each input sample $\mathbf{x}$. Moreover, in contrast with histograms based on regular grids, the number of bins K is here a priori defined, and does not need to grow exponentially with d.

Kernel-QT

The trianing of a kQT comprises *i)* the projection of TR by f_k, whose cost depends on the specific kernel function, *ii)* the computation of the split value, which costs $\mathcal{O}(N)$, and *iii)* the centroid selection. The cost of computing the Euclidean distance - or other distances based on l_p norms such as the Manhattan and the fractional $l_{0.1}$ - is $\mathcal{O}(d)$, while the Mahalanobis costs $\mathcal{O}(d^2)$ and the Weighted Mahalanobis costs $\mathcal{O}(Md^2)$, where M is the number of Gaussian components fitted to TR. The cost of computing the information gain is dominated by the computation of the determinant (see Eq. 2.2), which costs $\mathcal{O}(d^3)$, while computing the Gini index only requires the distances between the training samples and the centroids, already computed to define S_k. Overall, the cost of the index computation is multiplied by the number of centroids T tested during the selection procedure; therefore, an upper bound for the cost of KQT construction is $\mathcal{O}(KT(N+MNd^2+d^3))$ when using the Weighted Mahalanobis distance and the information gain. During monitoring, the only operation performed is the projection by f_k of the samples of a batch W, resulting in a cost of $\mathcal{O}(\nu KMd^2)$ in case of the Weighted Mahalanobis distance.

QT-EWMA and KQT-EWMA

As QuantTree, QT-EWMA is extremely efficient from both computational and memory points of view. Both place the incoming sample x_t in the corresponding bin of the histogram, resulting in $\mathcal{O}(K)$ operations, where K is the number of bins. Then, QT-EWMA computes the test statistics and these operations have a constant cost that falls within $\mathcal{O}(K)$. QT computes the Pearson statistic at the end of each batch, and this does not increase the order of computational complexity either. In terms of memory requirements, QT-EWMA update the statistics $Z_{j,t}$ at each new sample x_t, which requires storing only the K values $Z_{j,t-1}$, $j = 1, ..., K$. The same can be stated for KQT-EWMA: still, placing the incoming sample in the right bin, and computing the test statistic, have the costant complexity of order $\mathcal{O}(K)$.

Chapter 3

Theoretical Results

3.1 KQT with p-norms

Let $p \geq 1$ be a real number. The p-norm of a vector $\mathbf{x} = (x_1, \dots, x_n)$ is:

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{3.1}$$

For $p = 2$ we get the Euclidean norm, for $p = 1$ we get the Manhattan norm. The definition is still of some interest for $0 < p < 1$, but the resulting function does not define a norm, because it violates the triangle inequality. In linear algebra and functional analysis, this is called a quasi-norm; it is similar to a norm since it satisfies the norm axioms, except that the triangle inequality is replaced by:

$$\|x + y\| \leq K(\|x\| + \|y\|)$$

for some $K > 1$. A quasi-norm on a vector space (such as $\mathbb{R}^d$) is a real-valued map. In this framework, a p-quasinorm with $0 < p < 1$ is still a real-valued function from $\mathbb{R}^d$ to $\mathbb{R}$.

Here, we report some theoretical results from the paper introducing KQT [25] to extend the validity of the FPR control to these quasi-norms derived distances to be used as kernel functions. The proofs are in the supplementary material of the paper. The Generalized QuantTree (GQT) histogram $h = \{(S_k, \hat{\pi}_k)\}$ partitions the input space $\mathbb{R}^d$ such that the probability $\hat{\pi}_k$ of a stationary sample $\mathbf{x} \sim \phi_0$ to fall in bin S_k is close to a target probability π_k provided as an input parameter. During testing, GQT monitors batches W of ν samples by computing a test statistic $\mathcal{T}_h$ whose value only depends on the number of samples of W falling in each bin. Then, the test statistic is compared against a detection threshold $\tau \in \mathbb{R}$, and a change is detected when $\mathcal{T}_h(W) > \tau$. A peculiarity of GQT is that each bin S_k is defined as a subset of the sublevel set of a *measurable* kernel function $f_k : \mathbb{R}^d \to \mathbb{R}$.

Proposition 1. *Let (X, Σ) and (Y, T) be measurable spaces, meaning that X and Y are sets equipped with respective σ-algebras Σ and T. A function $f : X \to Y$ is said to be measurable if for every $E \in T$ the pre-image of E under f is in Σ; that is, for all $E \in T$*

$$f^{-1}(E) := \{x \in X | f(x) \in E\} \in \Sigma$$

Proposition 2. *In the case $f : X \to \mathbb{R}$, continuous functions are all measurable.*

This follows from the fact that, for every $t \in \mathbb{R}$, the set $E = \{x \in X : f(x) > t\}$ is open and therefore measurable.

Theorem 2. *Let $h = \{(S_k, \hat{\pi}_k)\}_{k=1}^{k}$ be a Generalized QuantTree histogram constructed using measurable functions $f_k : \mathbb{R}^d \to \mathbb{R} \forall k$. Let $\mathcal{T}_h$ be a statistic defined over batches W such that $\mathcal{T}_h(W)$ only depends on the number of samples $y_1, \ldots, y_K$ of W falling in the bins of h. Then, the distribution of $\mathcal{T}_h$ over stationary batches $W \sim \phi_0$ depends only on the batch size ν, the number of training points N and target probabilities $\{\pi_k\}_k$*

Theorem 2 implies that the distribution $\mathcal{T}_h$ computed over stationary batches by a GQT is independent of ϕ_0, d or $\{f_k\}$, thus allowing us to empirically estimate its distribution and compute a threshold τ such that the FPR achieved by GQT is controlled. It is a generalization of Theorem 1, reported from [6], which is still valid when adopting distances derived from l_p norms and quasi-norms (when $p \leq 1$) since these are measurable functions given Prop. 2. Indeed, as long as f_k is measurable for the considered space and X is a continuous random variable (r.v.) in $\mathbb{R}^d$, by the properties of continuous r.v., we have that $Z = f_k(X)$ is also a continuous r.v. in $\mathbb{R}$. The proof in [25] is thus unchanged.

3.2 Online implementation of KQT

We report some definitions used to define the EWMA statistic T_t in [10]. T_t monitors the proportion of samples falling in each bin of the histogram constructed over TR. In particular, when a new sample x_t is acquired, we define K binary statistics from the indicator functions of each bin S_j, namely

$$y_{j,t} = \mathbb{I}(x_t \in S_j), \quad j = 1, \ldots, K \tag{3.2}$$

to track in which bin x_t falls. Denoting the true bin probabilities $p_j = \mathbb{P}_{\phi_0}(S_j)$, i.e. the set of probabilities of a point sampled from ϕ_0 to belong to S_j, we have that:

$$\mathbb{E}_{\phi_0}[y_{j,t}] = p_j, \quad j = 1, \ldots, K \tag{3.3}$$

where the expected value $\mathbb{E}_{\phi_0}$ is computed under the assumption that $x_t \sim \phi_0$. Since ϕ_0 is unknown, so are the bin probabilities $(p_1, \ldots, p_K)$, which are a realization of a random vector and can be approximated by $\widetilde{\pi}_j \approx p_j$, where $\widetilde{\pi}_1, \ldots, \widetilde{\pi}_K$ are defined as:

$$\hat{\pi}_j := \frac{N\pi_j}{N+1}, \quad j < K \quad \text{and} \quad \hat{\pi}_K := \frac{N + 1\pi_K}{N+1} \tag{3.4}$$

The distribution of any statistic defined over a GQT histogram does not depend on ϕ_0 nor on d, thus KQT-EWMA is a nonparametric change detection algorithm. We report Proposition 1 from the paper [10] and show that the proof is also valid in our framework adopting Generalized QT.

Proposition 3. *Let $\{S_j\}_{j=1}^K$ be a partitioning built by the (Generalized) QuantTree algorithm with target probabilities $\{\pi_j\}_{j=1}^K$ on a training set $TR \sim \phi_0$ of size N. Then, the bin probability vector $(p_1, \dots, p_K)$ is drawn from the Dirichlet distribution:*

$$(p_1, \dots, p_K) \sim \mathcal{D}(\pi_1 N, \pi_2 N, \dots, \pi_K N + 1) \tag{3.5}$$

Proof. We leverage the results in [9], linking the Dirichlet distribution to the *stick-breaking process.* In particular, the process generates a sequence of K random variables $q_1, \dots, q_K$ as:

$$q_j = \prod_{k=1}^{j-1}(1 - \widetilde{q}_k) \cdot \widetilde{q}_j, \quad j < K, \qquad q_K = 1 - \sum_{j=1}^{K-1} q_j \tag{3.6}$$

where $\widetilde{q}_j$ for $j = 1, \dots, K-1$ are defined as:

$$\widetilde{q}_j \sim \text{Beta}(\gamma_j, \sum_{k=j+1}^{K} \gamma_j) \tag{3.7}$$

and $\gamma_1, \dots, \gamma_K$ are the parameters that define the stick-breaking process. In [9] it has been shown that:

$$(q_1, \dots, q_k) \sim \mathcal{D}(\gamma_1, \dots, \gamma_K) \tag{3.8}$$

To prove the proposition it is enough to show that there exists a specific configuration of γ_j that the bin probabilities p_j of a (Generalized) QT histogram can be expressed as q_j in 3.6. To this purpose, we recall three Propositions from [25].

Proposition 4. *Let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$ be i.i.d. realizations of a continuous random vector $\mathbf{X}$ defined over $\mathcal{D} \subset \mathbb{R}^d$. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a measurable function, and let $Z = f(\mathbf{X})$. We denote with $z_{(1)} \leq z_{(2)} \leq \dots \leq z_{(M)}$ the sorted images of $\{\mathbf{x}_j\}$ through f. For any $L \in \{1, 2, \dots, M\}$, we define the sublevel sets:*

$$Q_{f,L} := \{\mathbf{x} \in \mathcal{D} : f(\mathbf{x}) \leq z_{(L)}\} \tag{3.9}$$

Then the random variable $p = P_{\mathbf{x}}(Q_{f,L})$ is distributed as Beta$(L, M - L + 1)$

Proposition 5.

$$p_k = \widetilde{p}_k \cdot \left(1 - \sum_{j<k} p_j\right) = \widetilde{p}_k \prod_{j<k}(1 - \widetilde{p}_j) \tag{3.10}$$

Proposition 6. *For a Generalized QuantTree histogram, the random variables $\{\widetilde{p}_k\}$ are independent.*

These results from [25] show that p_j can be written as in 3.10, where $\widetilde{p}_j$ are independent and follow Beta distributions:

$$\widetilde{p}_j \sim \text{Beta}(\pi_j N, (1 - \sum_{k=1}^{j} \pi_k)N + 1), \quad j = 1, \dots, K-1 \tag{3.11}$$

Now, we only need to find a suitable choice of $\gamma_1, \dots, \gamma_K$ to express the $\widetilde{p}_j$ as the $\widetilde{q}_j$ in 3.6. If we define $\gamma_j = \pi_j N$ for $j < K$ as in 3.11 and $\gamma_K = \pi_K N + 1$, we obtain that:

$$\sum_{k=j+1}^{K} \gamma_k = \sum_{k=j+1}^{K} \pi_k N + 1 = (1 - \sum_{k=1}^{j} \pi_k)N + 1 \tag{3.12}$$

where the last equality follows from $\sum_{j=1}^{K} \pi_j = 1$. Equation 3.12 ensures the correspondence between $\widetilde{p}_j$ in 3.11 and $\widetilde{q}_j$ in 3.7, which implies the thesis.

□

Chapter 4

Methods

4.1 Other Drift Detection strategies

In the field of concept drift detection, various methods are employed: some are based on the data distribution, using distance functions or metrics to measure the dissimilarity between the distribution of historical data and new data. Parametric methods assume a known probability distribution, tend to be quite fast and also require significantly less data compared to non-parametric methods. Additionally, since they tend to be less flexible and suitable for less complex problems, they are more interpretable. However, these methods are not suitable for real-world data streams, where ϕ_0 is unknown; as non-parametric methods make fewer assumptions, their applicability is much more general. Also, due to the reliance on fewer assumptions, non-parametric methods are more robust. Histograms provide a simple and flexible non-parametric modeling approach for describing the underlying distribution ϕ_0.

4.1.1 EI-kMeans

Equal Intensity k-means (EIKM) Space Partitioning is a cluster-based histogram proposed in 2019 [A.Liu, J. Lu, G. Zhang, *Concept Drift Detection via Equal Intensity k-means Space Partitioning* [19]] "to address the problems caused by irregular partitions". EIKM builds a histogram that dynamically partitions the data into an appropriate number of small clusters, searching for the best centroids to create partitions; it then applies Pearson's chi-square test (χ^2 test) to conduct the null hypothesis test. Pearson's test ensures that the test statistics remain independent of the sample distribution and the sample size; moreover, the drift threshold can be calculated directly according to Chi-square distribution and it can be implemented in an offline manner. The overall EIKM complexity is $O(nK)$.

The algorithm can be summarized into 3 steps:

1. Initialize the greedy equal-intensity cluster centroids.
2. Segment the feature space as small clusters. This step is based on k-means clustering, which divides the datasets into a set of individual clusters. This ensures no partition will step across clusters. The number of partitions is continuously reduced if the number of samples in each partition does not satisfy the desired values.
3. Detect drifts with Pearson's chi-square test

4.1.2 SPLL

Semiparametric log-likelihood detector (SPLL) was proposed in L. I. Kuncheva, "Change Detection in Streaming Multivariate Data Using Likelihood Detectors" (2013) [15]. It models the stationary distribution Φ_0 as a Gaussian Mixture Model (GMM) (semiparametric approximation), and number of components in the mixture can be small as $C = 2$. During inference, the test statistic associated with a batch W is computed by an upper bound of the log-likelihood of its samples, which is proportional to the sum of the negative squared Mahalanobis distances between each observation and its closest mean. The expression proposed is:

$$SPLL = -\frac{\overline{LL}}{\nu} \propto \frac{1}{\nu_2} \sum_{\mathbf{x} \in W} \left(\mathbf{x} - \mu_{i*}\right)^T \Sigma^{-1} \left(\mathbf{x} - \mu_{i*}\right).$$

where ν is the cardinality of the batch W and $\overline{LL}$ is the scaled upper bound.

The adaptation of the one-shot change detection algorithm SPLL [15] also comes from [10], which present an adaptation scheme for both batch-wise and element-wise algorithm to the streaming case. SPLL fits a Gaussian Mixture $\hat{\phi}_0$ to approximate the stationary distribution. Upon this, a test statistic computed batch-wise T^ν is defined. The algorithm detects a change as soon as $T^\nu(W_t) > h^\nu$, where the threshold h^ν is set to control the probability of false alarm over each individual batch. The very same monitoring scheme can be adopted to monitor datastreams at a controlled ARL_0, as long as the threshold h^ν is accordingly modified. Let the threshold h^ν be such that:

$$\mathbb{P}(T^\nu(W) > h^\nu) = \alpha \tag{4.1}$$

where W is a batch of ν samples drawn from ϕ_0. Then, the monitoring scheme $T^\nu(W) > h^\nu$ yields $ARL_0 \geq \nu/\alpha$. This implies that, when we set $\alpha = \nu/ARL_0$, our online change detection algorithm is conservative, since its ARL_0 can be greater than the target. In some cases, however, we can provide guarantees that $\alpha = \nu/ARL_0$, to exactly control the ARL_0, as shown in the following proposition, which is proved in the supplementary material of [10]. Let the threshold h^ν be such that:

$$\mathbb{P}(T^\nu(W) > h^\nu | TR) = \alpha \tag{4.2}$$

where W is a batch of ν samples drawn from ϕ_0. Then, the monitoring scheme $T^\nu(W) > h^\nu$ yields $ARL_0 = \nu/\alpha$. In the batch-wise SPLL detector, we compute the thresholds to guarantee the FPR using bootstrap over the training set TR since the distribution of the statistics depends on ϕ_0. We are under the hypothesis of this last proposition since the false positive probability is conditioned on the training set realization. Therefore, by adopting the same monitoring scheme and setting h^ν to guarantee $\mathbb{P}(T^\nu(W) > h^\nu | TR) = \nu/ARL_0$ we can obtain the target ARL_0.

Computational Complexity - A comparison

A comparison of the computational complexity of QT and the other considered methods is reported in the following table, where M is the number of Gaussian components employed by KQT with the Weighted Mahalanobis distance.

Method	Training Cost	Inference Cost
KQT (Weighted Maha.)	$O\left(KT\left(N + MNd^2 + d^3\right)\right)$	$O\left(\nu KMd^2\right)$
KQT (l_p (quasi)-norms)	$O\left(KT\left(N + Nd + d^3\right)\right)$	$O\left(\nu Kd\right)$
QuantTree	$O(KN \log N)$	$O(\nu K)$
EIKM	$O\left(K^2 N \log N\right)$	$O(\nu K)$

SPLL test need to compute the log-likelihood of each sample with respect to each of the m components of a GMM $\hat{\phi}_0$ fit on TR. This results in $\mathcal{O}(md)$ operations per sample. In case of batch-wise monitoring, the log-likelihood is averaged over the batch and only 1 value has to be stored. Details can be read in the original paper [15]. To monitor the log-likelihood of each new observation x_t with respect to $\hat{\phi}_0$. The statistic used requires sorting the entire sequence of log-likelihood values, which results in additional $\mathcal{O}(t \log t)$ operations. In this case, all the t values of the sequence have to be analyzed and stored, thus computational complexity and memory requirement steadily increase over time. Since this is not appropriate for online monitoring, the algorithm operates over a window of most recent w samples.

A comparison of the computational complexity of QT-EWMA and the other considered methods is reported in the following table, where M is the number of Gaussian components employed by KQT with the Weighted Mahalanobis distance, m is the number of Gaussian components used by SPLL, and w is the cardinality of the window of most recent samples employed to compute the test statistic.

Method	Training Cost	Inference Cost (per sample)
KQT-EWMA (W.M.)	$O\left(KT\left(N + MNd^2 + d^3\right)\right)$	$O\left(KMd^2\right)$
KQT-EWMA (l_p)	$O\left(KT\left(N + Nd + d^3\right)\right)$	$O\left(Kd\right)$
QT-EWMA	$O(KN \log N)$	$O(K)$
SPLL (online)	$O\left(mNd^2\right)$	$O\left(md + w \log w\right)$

4.2 Dimensionality Reduction techniques

Dimensionality reduction (DR) is a fundamental technique for improving the performance of machine learning algorithms. It is defined as the projection of high dimensional data into a low dimensional space by reducing the input features to the most relevant ones. There are two main methods: feature transformation and feature selection. Feature transformation, which we'll call DR for simplicity, creates new feature combinations by identifying patterns and reducing noise in the original dataset. On the other hand, feature selection retains the most relevant features from the original dataset without changing them. In a more formal sense, DR is about finding a function or *map* $A : \mathbb{R}^d \rightarrow \mathbb{R}^p$, where $p \ll d$, to be applied to each instance (X_i) of a dataframe. The work by Bahri, Bifet, et al. (2020), presents a taxonomy that categorizes DR techniques into three main groups:

1. Data-Dependent Techniques: These methods rely on the entirety of the input data to perform the transformation. They analyze the data's intrinsic characteristics to achieve the dimensionality reduction.

2. Data-Independent Techniques: In contrast, data-independent techniques are based on random projections. These techniques do not rely on the specific properties of the input data for the transformation. Instead, they employ random projections to reduce dimensionality without considering the underlying structure of the data.

3. Graph-Based Techniques: This category consists of data-dependent techniques that employ a graph-based approach. The primary objective of graph-based techniques is to preserve the local structure or neighborhood relationships of the data even after the dimensionality reduction process.

Data-dependent techniques construct a projection function – or matrix – from the data. This requires the presence of the entirety – or at least a part of – the dataset. In the streaming context, where data are potentially infinite, the classical techniques from this category are therefore limited, since keeping the entire data stream in memory is impractical. Principal Components Analysis is one important example. Data-independent techniques are mainly based on the principle of random projections. These techniques are therefore appropriate for evolving streams because they generate the projection matrices (or functions), and transform data into a low-dimensional space, independently from the input data.

An interesting property of any dimensionality reduction technique is to consider its stability. In this context, a technique is said to be ϵ stable, if for any two input data points, x_1 and x_2 , the following inequation holds:

$$(1-\epsilon) * ||x-y||^2 \leq ||\Phi(x)-\Phi(y)||^2 \leq (1+\epsilon) * ||x-y||^2$$

that is, the Euclidean distances in the original input space are relatively conserved in the output feature space.

PCA can be computationally expensive for large data and feature selection requires additional processing steps. Random Projections or hashing can be much faster and still achieve good results in reducing dimensionality. We will experiment with different techniques to identify the most relevant features in the dataframes given the purposes and methods we discussed, considering accuracy and processing time.

4.2.1 Johnson-Lindenstrauss Lemma and Random Projections

The Johnson-Lindenstrauss Lemma is a fundamental result in mathematics and computer science, particularly in the field of dimensionality reduction and high-dimensional data analysis. It addresses the problem of reducing the dimensionality of data points while approximately preserving their pairwise distances. This lemma is crucial in various applications such as machine learning, data compression, and data visualization.

Statement

Given a set of N data points in a d-dimensional Euclidean space, the Johnson-Lindenstrauss Lemma (JLL) states that for any $\epsilon > 0$ and a sufficiently large number of dimensions m (with $m \leq d$), there exists a linear mapping (or projection) Φ from the high-dimensional space to a lower-dimensional space of dimension m such that the pairwise distances between the projected points are approximately preserved. In other words, for all data points x and y:

$$(1-\epsilon) * ||x-y||^2 \leq ||\Phi(x) - \Phi(y)||^2 \leq (1+\epsilon) * ||x-y||^2 \tag{4.3}$$

Where $||x - y||$ is the Euclidean distance between data points x and y in the original high-dimensional space, $\Phi(x)$ and $\Phi(y)$ are the projections of x and y onto the lower-dimensional space. To obtain a good approximation, m (the dimension of the lower-dimensional space) needs to be sufficiently large, but it is much smaller than the original dimension d. The exact value of m depends on ϵ and the number of data points N. There is a trade-off between dimensionality reduction (m) and the quality of the approximation (ϵ). In particular, a set of N points in a high dimensional euclidean space can be mapped into a $\mathcal{O}(\log N/\epsilon^2$ dimensional euclidean space such that distance between any two points change only by a factor $1 \pm \epsilon$. We notice that the new dimensionality m is independent of d, all pairwise distances are maintained within an arbitrarily small ϵ.

Random projections

While principal component analysis (PCA) finds an embedding with optimal leverage distortion of instances between the original and embedded vectors, the Johnson-Lindenstrauss Lemma bounds the worst case distortion between the distances within the original space and distances within the embdedded space. JLL is powerful not only because it guarantees the existence of a suitable mapping Φ for dimensionality reduction but also because it doesn't prescribe a specific form for Φ. Instead, it asserts that such a mapping can be found with certain properties. This flexibility in choosing the mapping makes the lemma practically useful. *Random projections* (RPs) involve mapping high-dimensional data onto a lower-dimensional subspace using a random linear transformation. This transformation is typically achieved through the multiplication of the data matrix by a random matrix with appropriately scaled entries. Random projections are appealing due to their computational efficiency and ability to approximate pairwise distances between data points. Moreover, they work well across various types of data and do not rely on data-specific characteristics. In their paper "*Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*" [12], Halko, Martinsson, and Tropp provide a comprehensive overview of the theory and applications of RPs, particularly in the context of approximating matrix decompositions, such as the Singular Value Decomposition (SVD) / Principal Component Analysis (PCA). RPs can be used to construct low-rank approximations of matrices. This is particularly valuable when dealing with massive datasets where exact factorizations are computationally expensive. The authors provide theoretical guarantees for the quality of the low-rank approximations and introduce randomized algorithms that leverage random projections to efficiently compute approximate matrix factorizations. These algorithms significantly reduce computational complexity compared to traditional deterministic methods. Additionally, they present empirical results and emphasize that random projection-based methods offer a flexible framework that can be adapted to different problem domains and data types.

4.2.2 Eckart-Young Theorem and Principal Components Analysis

Principal component analysis (PCA) stands as one of the most widely used and straightforward unsupervised techniques. A lower dimensional basis is sought in such a way to minimize the sum of squared distances between the original data points and their projections. Essentially, PCA aims to make these distances as close to zero as possible while simultaneously maximizing the variances between the first few components. In mathematical terms, PCA seeks to discover a linear mapping created by a set of orthogonal linear combinations, often referred to as eigenvectors or principal components (PCs). However, it's important to note that PCA calculates these eigenvectors and eigenvalues based on a computed covariance matrix, relying on the entire dataset. This approach poses challenges for streaming data scenarios because it necessitates re-estimating the covariance matrix entirely when new observations arrive, which can be computationally inefficient.

PCA can be seen as a specific application of *Singular Value Decomposition* (SVD), which is a more general factorization technique that decomposes a matrix into its constituent singular values and vectors, enabling applications in data compression, noise reduction, information retrieval, ... Given a matrix A, SVD decomposes it into $A = U\Sigma V^T$, where U and V are called respectively "left" and "right" singular vectors matrix and Σ is a pseudo-diagonal matrix containing the singular values. In 1936, Carl Eckart and Gale Young published a seminal paper titled "The approximation of One Matrix by Another of Lower Rank". This laid the foundation of understanding how to approximate a given matrix with a low-rank one, which is a core concept in SVD. The Eckart-Young Theorem states that for any rank-k approximation of a matrix A, where $k < rank(A)$, the best approximation (in terms of the Frobenius or the operator norm) is achieved by using the first k singular values and their corresponding singular vectors from the SVD decomposition of A:

$$A_k = U_k \Sigma_k V_k^T$$

where A_k is the best rank-k approximation of matrix A, U_k, Σ_k and V_k^T are the first k singular vectors and values of the decomposition. Eckart and Young's work is fundamental to understand how SVD can be applied to approximate and reduce data dimensionality and their theorem provides a rigorous mathematical framework.

PCA aim is to maximize the variance and minimize the covariance of data features in this new base, which thus depends on the data itself.

1. Let A be our data (N, d) matrix. We define $\bar{A}$ as its centered version with respect to the mean of A columns: $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ such that $\bar{A} = HA$
2. We define the sample covariance matrix as $S = \frac{\bar{A}^T \bar{A}}{N-1}$, thus S is symmetric and positive definite
3. We solve $S = VDV^T$; $D = V^T S V$ contains the eigenvalues of S and V vectors are the principal components

The problem of finding k orthonormal directions $\mathbf{v}_i$ is the same as to minimize the representation error:

$$J_{PCA} = \mathbb{E}||\mathbf{x} - \sum_{i=1}^{k} \langle \mathbf{v}_i, \mathbf{x} \rangle \mathbf{v}_i||^2$$

Randomized SVD/PCA

Randomized SVD is an innovative and efficient technique for approximating the dominant singular values and vectors of a matrix, making it particularly valuable in scenarios where traditional SVD methods are computationally infeasible. It leverages randomization and sampling to approximate the dominant singular values and vectors of a matrix efficiently. It offers several advantages:

- Scalability: Randomized SVD is particularly well-suited for large-scale datasets. By using random projections and sampling, it significantly reduces the computational complexity compared to traditional SVD methods.
- Memory Efficiency: Unlike traditional SVD, which requires storing the entire matrix, it operates on matrix sketches or subsets, making it memory-efficient for high-dimensional data.

- Speed: The use of randomization techniques enables faster computation of singular values and vectors, making it suitable for real-time or near-real-time applications.

We report here the algorithm shown in Halko et al. (2010) work.

1. Let A be a matrix with intrinsic low rank r. We build a random projection matrix: $A \in \mathbb{R}^{\mathbb{N}*}$, $P \in \mathbb{R}^{*\setminus}$ then $Z = AP \in \mathbb{R}^{\mathbb{N}*\setminus}$
2. We perform QR factorization of Z, $Z = QR$; then Q will be an orthonormal basis for A
3. We project A on Q: $W = Q^T A$
4. We compute SVD decomposition for W: $W = U_w \Sigma_w V_w^T$
5. Then, $A \approx U_r \Sigma_r V_r^T \approx U \Sigma_w V_w^T$ where $U = QU_w$

That is, we first construct a low dimensional space that captures the action of the data matrix, then we restrict the matrix to that subspace and compute a standard factorization (QR, SVD, ...) here.

Sample-based PCA vs "theoretical" PCA

To study the algorithm behavior under different conditions, as an alternative approach and mostly as a "sanity check", we explored another similar technique which we refer to as "theoretical" version of PCA. The preprocessing usually operates on the training data, employing SVD to derive the principal components. This process, being sample-dependent, is inherently influenced by the peculiarities of the observed dataset. Rather than relying on the sample-specific covariance matrix of the training data, our "theoretical" algorithm leverages the covariance matrix derived from the original distribution used to generate both training and testing datasets: by utilizing the eigenvectors and eigenvalues of the theoretical covariance matrix, we mitigate potential issues arising from outliers and get a broader perspective by aligning the DR with the underlying statistical properties of the data generation process. It is worth noting that while sample-based PCA offers valuable insights into the algorithm's behavior in realistic settings, theoretical PCA serves as a benchmark for understanding fundamental characteristics and assessing the algorithm's robustness. The comparison between these two approaches is not fair, since in practical situations the true underlying distribution of the data is unknown or highly complex: theoretical PCA cannot be applicable, it does not mirror real-world conditions, but in our case is a valuable instrument for assessing the potential impact of extreme cases or outliers in the algorithm behavior, acting as a stress test. We explored different configurations:

1. (Centering (removing the mean))
2. Rotation (projection on eigenvectors)
3. Centering + Rotation

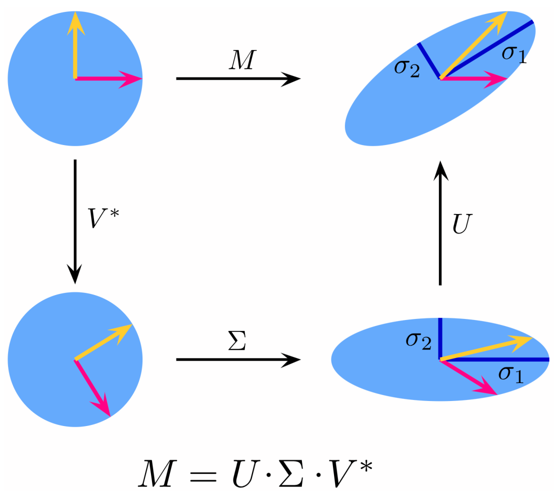


Figure 4.1: from Wikipedia: Illustration of the singular value decomposition $U\Sigma V^*$ of a real 2×2 matrix M. **Top**: The action of M, indicated by its effect on the unit disc D and the two canonical unit vectors e1 and e2. **Left**: The action of V^*, a rotation, on D, e_1, and e_2. **Bottom**: The action of Σ, a scaling by the singular values σ_1 horizontally and σ_2 vertically. **Right**: The action of U, another rotation.

4. Rotation + Scaling (projection on eigenvectors, multiplication by eigenvalues)

5. Centering + Rotation + Scaling

It's worth noting that QT performance is robust to any translation, since the histogram spans the entire space guided by the distribution's quantiles. Although translation-invariant, the transformation induced by centering before PCA may alter the subsequent rotation and scaling components.

Robust PCA

If the input vectors are contaminated by outliers, the estimation of mean and covariance may be wrong, resulting in very poor performance of the PCA. There are approaches to have a robust PCA which basically amounts to have robust estimates of the mean and covariance. In 2009, Emmanuel J. Candes, Xiaodong Li, Yi Ma, and John Wright published "a paper about a curious phenomenon: recovering the superposition of a low rank matrix and a sparse matrix as components of the data matrix"[8]:

$$A = L_0 + S_0$$

with L_0 low rank and S_0 sparse. Classical PCA seeks the best rank-r matrix in a l^2 sense, by solving:

$$\begin{aligned} &\text{minimize} \quad ||A - L|| \\ &\text{subject to} \quad \text{rank}(L) \leq r \end{aligned}$$

"however, its brittleness with respect to grossly corrupted observations often puts its validity in jeopardy - a single corrupted entry in A could render the estimated $\hat{L}$ arbitrarily far from the true L_0. The authors aim to recover a low-rank matrix L_0 from highly corrupted measurements $A = L_0 + S_0$ where entries in S_0 can have arbitrarily large magnitude, and their support is assumed to be sparse, but unknown. Their proposed solutions *principal component pursuit* (PCP) was used for our experiments:

$$\begin{aligned} &\text{minimize} \quad ||L||_* + \lambda ||S||_1 \\ &\text{subject to} \quad L + S = A \end{aligned} \tag{4.4}$$

where $|| \quad ||_*$ represents the nuclear norm, and $|| \quad ||_1$ the l_1 norm.

4.2.3 Kernel PCA

Kernel Principal Component Analysis was presented during the International Conference on Artificial Neural Networks (ICANN) in 1997 by B. Schölkopf, A. Smola, and K.R. Müller. They generalized PCA to a nonlinear setting. As reported in the paper [23], kPCA first maps the data nonlinearly into a feature space F by

$$\Phi : \mathbb{R}^N \to F, \quad \mathbf{a} \mapsto \mathbf{A}$$

Even if F has arbitrarily large dimensionality, for certain choices of Φ:, PCA can still be performed in F, by the use of kernel functions known from Support Vector Machines. Kernel functions calculates the dot product of data samples under Φ: $\kappa(\mathbf{a_i}, \mathbf{a_j}) = \phi(\mathbf{a_i})\phi(\mathbf{a_j})^T$. In our case, we used a *radial basis function* (RBF) kernel , i.e.:

$$\kappa(\mathbf{a_i}, \mathbf{a_j}) = \exp\left(-\gamma \|\mathbf{a_i} - \mathbf{a_j}\|_2^2\right)$$

This kernel is computed for every pair of points, projecting our dataset to this new subspace. Then, dimensionality reduction is conducted to obtain the principal components through SVD/PCA.

4.2.4 Independent Component Analysis

A simple problem used to describe ICA might be considering a recording studio with a single microphone positioned at its center, and various musicians playing different instruments around it. Each instrument's sound reaches the microphone and the recorded audio contains a blend of all the different sounds. The challenge is to find a way to extract individual recordings of each instrument from this mixed audio. Independent Component Analysis is a technique that transforms a dataframe into a set of maximally independent components. When applied to our waveform, ICA could effectively separate the different instruments into distinct, unmixed audio recordings. While PCA looks for uncorrelated factors (a constraint on the second-order statistic), ICA looks for independent factors (a constraint on all the moments). We must notice that after PCA we usually decide to retain a subset of the components as a reduced representation of the data; ICA, on the other hand, typically preserves the same number of inputs and outputs - since outputs are mutually independent, there is no obvious way to drop components. Still, ICA can potentially reduce dimensionality in the scenarios where there are few latent factors that explain data's variability while being independent from each other. As explained in A. Hyvärinen, E. Oja (2000) paper "Independent component analysis: algorithms and applications" an ICA model is strictly related with *non-gaussianity* [13]. The Central Limit Theorem, a fundamental concept in probability theory, states that when certain conditions are met, the distribution of the sum of independent random variables approaches a Gaussian distribution. Consequently, when you add two independent random variables together, the resulting distribution generally exhibits greater resemblance to a Gaussian distribution than either of the original two random variables individually. The paper outlines the algorithm we employed, providing details and recommendations (e.g. preprocessing steps).

Chapter 5

Batch-wise experiments

5.1 Datasets

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 should be affected by changes at known locations. Unfortunately, there are not many real-world datasets exhibiting real changes that satisfy these requirements. In practice, experiments are often performed on either synthetically generated data or on real-world datasets that have been manipulated introducing changes at known locations.

5.1.1 Synthetically generated data

Generating synthetic data, or manipulating real-world dataframes, is usually the most straightforward way 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. Most of the papers in the literature resort to introducing arbitrary shift/scaling, swapping few components of the original dataset, or mixing different datasets; obviously, these practices yield changes having magnitude that depends on the data distribution and dimension. 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. For example, in [16], the authors wrote that *it is difficult to find an acid test for change detection in unlabeled multidimensional data*; for their experiment to test SPLL with and without PCA, they chose two *change heuristics which could be regarded as instances of equipment failure*. In the first one (*Shuffle Values*), a random integer $k, 1 \leq k \leq d$ was generated to determine how many features/dimensions out of d will be affected; k random features are chosen, and the values of each feature are randomly permuted within the post-change dataframe. In the second one (*Shuffle Features*), again a random integer

$k, 1 \leq k \leq d$ is sampled to determine how many features/dimensions will be affected, then the chosen k features are randomly permuted within the post-change dataframe. As they state, *the shuffle values change resembles a case where a group of sensors stop working as a result of a technical fault and produce random readings within the sensor ranges. The shuffle features change can be likened to bleeding of signals into one another.* This way is not erroneous and provides results, but does not seem to be the best approach to generate a synthetic dataset. To enable a fair comparison among change-detection algorithms, C.Alippi, G.Boracchi and D.Carrera designed "Controlling Change Magnitude" (CCM), a rigorous method to introduce changes in multivariate datasets [2]. In particular, they measure the change magnitude as the symmetric Kullback-Leibler divergence (see Eq. 1.5) between the pre- and post-change distributions, and introduce changes by applying a rototranslation directly to the data. An algorithm to iteratively identify the rototranslation parameters yielding the desired change magnitude is presented in the paper.

Monomodal Gaussian

We consider a stationary distribution ϕ_0 which is a null-mean Gaussian with a random covariance matrix. The post-change distribution ϕ_1 is obtained by roto-translation using the CCM framework, such that the symmetric Kullback-Leibler distance (sKL) between ϕ_0 and ϕ_1 is fixed. If not specified, in our experiments the target sKL is set to 1. N points are sampled from ϕ_0 to generate the training set, and a number ν of samples is drawn to generate each one of the n_b batches both from ϕ_0 (pre-change test data, to compute FPR) and ϕ_1 (post-change test data, to compute TPR). In Figure 5.1 are shown two examples of

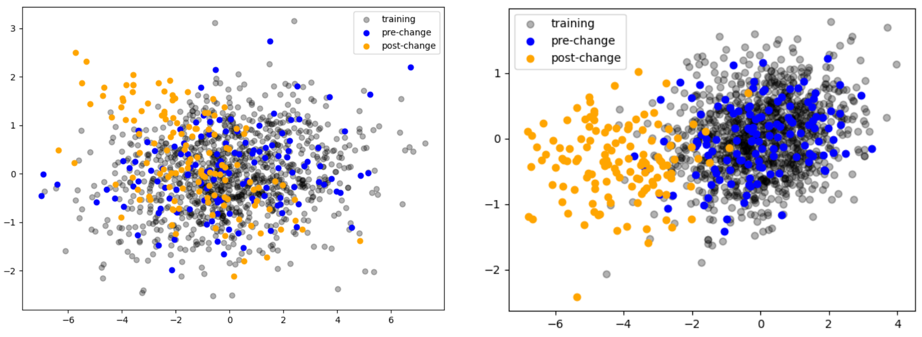


Figure 5.1: $N = 1024$ training points and one batch of 128 testing points drawn from a bidimensional monomodal gaussian ϕ_0 (respectively in gray and blue), and one batch of 128 testing points drawn from the post-change distribution ϕ_1 obtained through CMM such that sKL $= 1$ (a) and sKL $= 10$. (b)

this dataframe, with ϕ_1 built such that sKL $= 1$ and sKL $= 10$ respectively. As we observe the impact of increasing sKL on our synthetic datasets, discerning

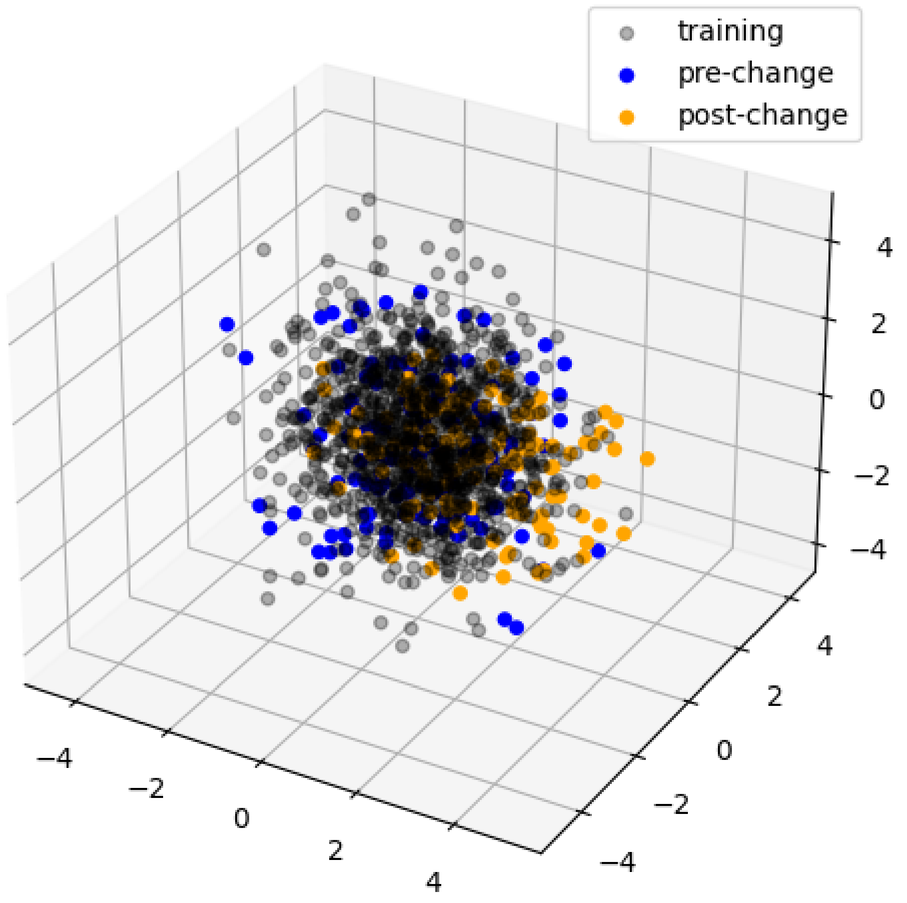


Figure 5.2: $N = 1024$ training points and one batch of 128 testing points drawn from a tridimensional monomodal gaussian ϕ_0 (respectively in gray and blue), and one batch of 128 testing points drawn from the post-change distribution ϕ_1 obtained through CMM such that sKL $= 1$.

distinct clusters among data points becomes evident. This can be observed both in two dimensions and three (as in Fig. 5.2) - even if in the latter case might be struggling since I had to print the dataframe on a 2D thesis page.

5.1.2 Real-world dataframes

In [6], the authors employ real-world datasets from the UCI Machine Learning Repository with dimensions ranging from $d = 5$ to $d = 50$. These datasets contain no distribution changes, thus stationary samples are drawn by sampling the datasets, while the post-change distribution ϕ_1 is generated by shifting stationary data in a random direction with a magnitude proportional to the variance of each component. In [25], a new UCI high dimensional dataframe (*Swarm dataframe*, with $d = 2400$) is used to assess KQT's performances. We rely on this dataframe because of its high dimensionality. Also, we don't have to construct or synthetically generate the post-change distribution, as we will describe.

Swarm Behavior dataset

The Swarm Behavior classification dataset from the UCI Machine Learning Repository comprises high-dimensional data ($d = 2400$) describing the motion of large groups of animals, which are labeled as *flocking* or *not-flocking.* Flocking behavior refers to the way that groups of birds, insects, fish or other animals, move close to each other. They are able to move as a group with the same velocity, yet without running into each other. Even though it is quite simple for an individual to move in a flock, the behavior of the whole group can appear complex to an observer. The University of New South Wales (UNSW Canberra) ran a survey explaining that "humans easily recognise flocking in nature, although sometimes it is hard for them to explain why. However, it is very hard for a machine to recognise flocking. We want to understand how humans do it". The dataset achieved from the survey contains 24017 instances with 2400 features each and has no missing values. The attributes are x_m, y_m as the position of each boid (*Boids* is an artificial life program, developed by Craig Reynolds in 1986, which simulates the flocking behaviour of birds. The name "boid" corresponds to a shortened version of 'bird-oid object"), then contains the velocity vector, an alignment vector, a separation vector, a cohesion vector, ... several attributes repeated for all m boids, where $m = 1, \dots, 200$. Class labels are binary; 1 refers to flocking, grouped, aligned, and 0 refers to not flocking, not grouped, and not aligned. In [25] the authors define the stationary distribution ϕ_0 as the distribution of data describing flocking groups of animals; in contrast, post-change distribution ϕ_1 is defined by data corresponding to non-flocking animals. An exemplary representation is given in Figure 5.3, where sequences of (x, y) positions of several individuals belonging to flocking (in orange) and non-flocking (in blue) groups are shown as a scatterplot.

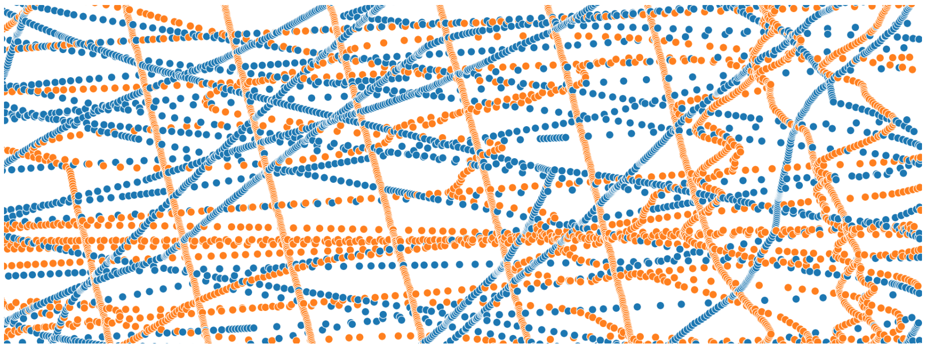


Figure 5.3: Series of datapoints representing position as (x, y) scattered for several individuals belonging to flocking (in orange) and non-flocking (in blue) groups. The dataframe can be found in the UC Irvine Machine Learning Repository.

5.2 Figures of Merit

As we wrote in the introduction (see Equations 1.7 and 1.8), True Positive Rate (TPR) and False Positive Rate (FPR) are important metrics used to evaluate the performance of a binary classification model. We assess the performance of batch-wise CD algorithms with these two standard figures of merit, which we report here.

$$TPR = \frac{\text{\# post-change batches triggering an alarm}}{\text{\# post-change batches}} * 100 \tag{5.1}$$

$$FPR = \frac{\text{\# pre-change batches triggering an alarm}}{\text{\# pre-change batches}} * 100 \tag{5.2}$$

We set the detection thresholds in our experiments to yield an empirical FPR of $\alpha = 5\%$. To compare the detection power, we rank the algorithms according to their TPR or the difference $\Delta(TPR - FPR)$. We write again that FPR is a critical metric in monitoring systems, a synonym of the trustworthiness of an alert, measuring the proportion of instances where the system incorrectly raises an alert, or signals an anomaly, when there is none; it can be more critical than the True Positive Rate (TPR). Quantitative criteria also include the execution time and the memory usage. Qualitative criteria, such as robustness, also play a significant role in evaluating algorithm performance.

5.3 Detectability loss with QuantTree and kQT

In [3] the authors show that the magnitude of a change can be naturally measured by the symmetric Kullback-Leibler divergence (Eq. 1.5); they consider change-detection problems in $\mathbb{R}^d$ and investigate how d affects the detectability of a change when monitoring the log-likelihood of a datastream, and introduce the

Signal-to-Noise Ratio of the change (SNR) to quantitatively assess the change detectability when monitoring the log-likelihood:

$$\mathrm{SNR}\,(\phi_0 \to \phi_1) := \frac{\left(\underset{\mathbf{x}\sim\phi_0}{E}[\mathcal{L}(\mathbf{x})] - \underset{\mathbf{x}\sim\phi_1}{E}[\mathcal{L}(\mathbf{x})]\right)^2}{\underset{\mathbf{x}\sim\phi_0}{\mathrm{var}}[\mathcal{L}(\mathbf{x})] + \underset{\mathbf{x}\sim\phi_1}{\mathrm{var}}[\mathcal{L}(\mathbf{x})]}$$

where var$[\cdot]$ denotes the variance of a random variable. In particular, SNR$(\phi_0 \to \phi_1)$ measures the extent to which $\phi_0 \to \phi_1$ is detectable by monitoring the expectation of $\mathcal{L}$, as they considered the log-likelihood of the analyzed data with respect to the stationary distribution ϕ_0 (which, we remind, is often unknown and has to be preliminarily estimated from a training set of stationary data.

In this empirical analysis, we generated training sets with $N \in \{1024, 2048, 4096, 8192, 16384\}$ and test batches ($\nu = 128$) from monomodal gaussian distributions with zero mean in $\mathbb{R}^d$ with $d \in \{2, 4, 8, 16, 32, 64, 128, 256\}$. The post change distribution ϕ_1 was built such that the symmetric KL is unitary. To test the effects of increasing dimensionality, we measure TPR and FPR obtained from post-change and stationary batches respectively with QuantTree and kernel QT histograms with $K = 64$ bins built, in the latter case, using Mahalanobis and Weighted Mahalanobis distances. The FPR desired value is set to $\alpha = 0.05$. Pearson statistic was chosen for the hypothesis tests. The data was generated 100 times and each time TPR and FPR were computed over 100+100 test batches. We plot the mean values of our results in Fig. 5.4.

The results reveales that with increasing dimensionality, the TPR converges to the FPR, even if the sKL divergence between stationary and post-change distributions is fixed. We can see that regardless of the dimensionality of the space it is built in, QuantTree (QT) histogram has effective control over the False Positive Rate at the specified threshold α. In contrast, both the Mahalanobis and Weighted Mahalanobis methods exhibited challenges in maintaining control with high dimensional data, leading to an undesirable escalation of both True and False Positive Rates to 100%. This observed outcomes can be changed employing a training set of increased cardinality $|TR|$ as it's evident in Figure 5.5. With the Mahalanobis distance, control over FPR is a little more stable. It is worth noting that FPR does not depend on sKL since it is only measured on batches drawn from the stationary distribution ϕ_0.

We provide a rigorous study of the challenges that change-detection methods have to face when data dimension scales. Our empirical analysis, performed by keeping the change-magnitude constant when scaling dimensionality, confirms detectability loss when using synthetically generated data from monomodal gaussians. Ongoing works concern extending this study to other change-detection approaches; we will try several dimensionality reduction techniques that can be applied to these algorithms, considering that one of the most important characteristics of QT is its small computational requirement, making it suitable for online applications.

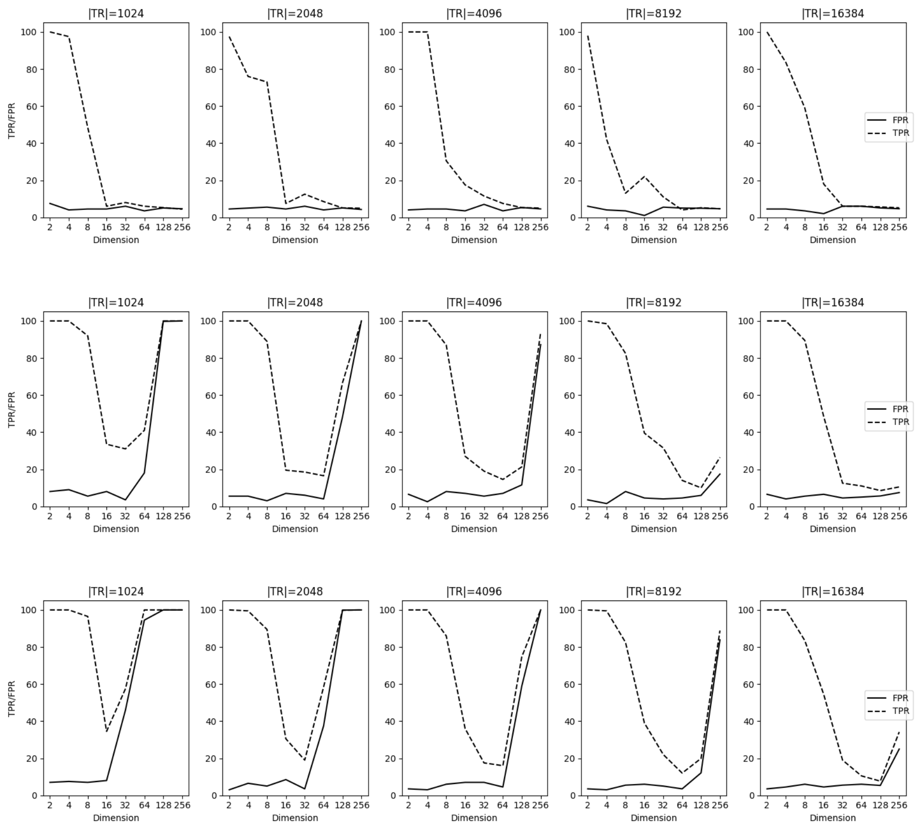


Figure 5.4: TPR and FPR measured on data from monomodal gaussians with increasing dimensionality and $sKL = 1$ fixed, by: (a) QuantTree; (b) Mahalanobis kQT; (c) Weighted Mahalanobis QT given training sets TR of increasing cardinality from left to right. In these experiments, the True Positive Rate (TPR) reaches the False Positive Rate (FPR) when the dimensionality of the data increases. A greater number of training points can delay this phenomenon. Moreover, while the QuantTree (QT) algorithm seems to effectively control the False Positive Rate at the given threshold at any dimension, both the Mahalanobis and Weighted Mahalanobis methods cannot. Consequently, in the latter cases, both the True Positive Rate and False Positive Rate escalate to 100%. Also this undesirable outcome can be mitigated and improved with a training set of increased cardinality. The data was generated 100 times and each time TPR and FPR were computed over 100+100 test batches.

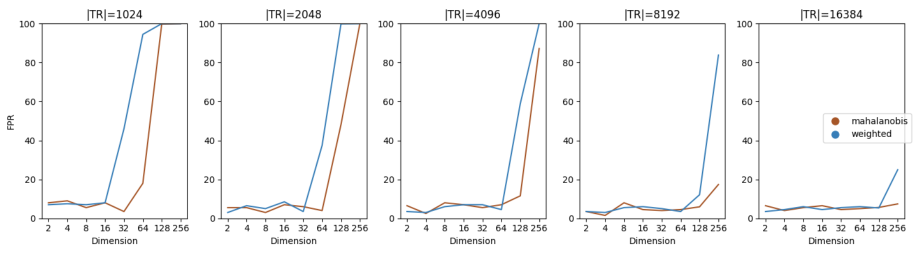


Figure 5.5: FPR obtained by kQT with weighted-M. and Mahalanobis distances over dataframes of increasing dimensionality with $sKL = 1$ fixed. Given the Mahalanobis distance, control over FPR is a little more stable. The data was generated 100 times and each time TPR and FPR were computed over 100+100 test batches.

Increasing sKL

What if we increase the distance between the stationary distribution and the post-change distribution? We try to perform the same experiment with QuantTree: all the parameters are the same, but sKL divergence (see Eq. 1.5) is gradually, bus substantially, incremented. We expect the FPR to not change, since it is an expected value computed on samples drawn from the stationary ϕ_0 (it is, indeed, completely independent from the post-change distribution, which QT has never seen being built), but the TPR to increase with the distances between the two distributions.

In Fig. 5.6 and 5.7 and we show this behavior. Even if these results are averaged over 5 experiments only, thus are noisy, the trend we expected is evident. QT achieves a 100% TPR on 4-dimensional Gaussians with $sKL(\phi_0 \rightarrow \phi_1) = 1$, but doubling the number of dimensions halves the TPR. In higher-dimensional spaces, is it necessary to increase the sKL distance between distributions? How much can QT performance improve with data dimensionality reduction techniques?

5.4 EIKM

In addition to QuantTree, the concept drift detection algorithm *Equal Intensity K-Means* (EIKM) Space Partitioning was employed for comparative analysis. This approach coinsists in dynamically dividing the data into small clusters, determining optimal centroids to create partitions. The algorithm then utilizes Pearson's chi-square test (χ^2 test) for conducting the null hypothesis test, ensuring the test statistics remain independent of the sample distribution and size.

The conducted experiments, illustrated in Figures 5.8 and 5.9, reveal that EIKM struggles to control the False Positive Rate (FPR) when dealing with high-dimensional data. Moreover, QuantTree generally achieves a higher power

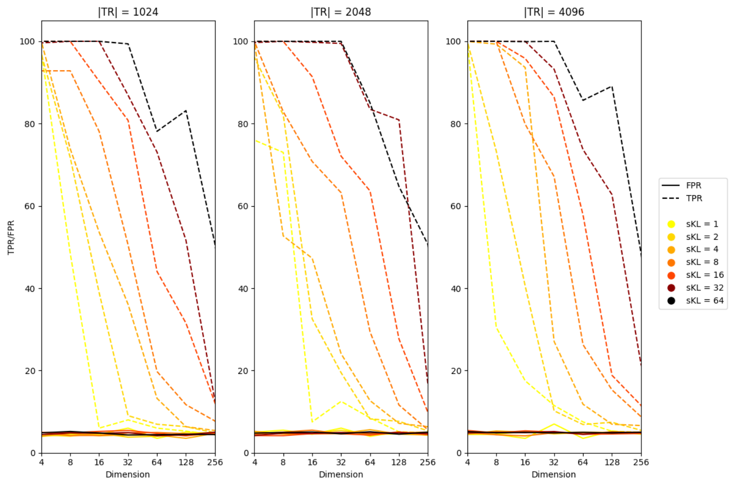


Figure 5.6: TPR and FPR obtained by QT over dataframes of increasing dimensionality with the distance between pre- and post-change distributions $sKL \in \{1, 2, 4, 8, 16, 32, 64\}$. The data was generated 5 times and each time TPR and FPR were computed over 512+512 test batches; we see the effects of *detectability loss* when data dimension scales, contrasting an increasing sKL divergence.

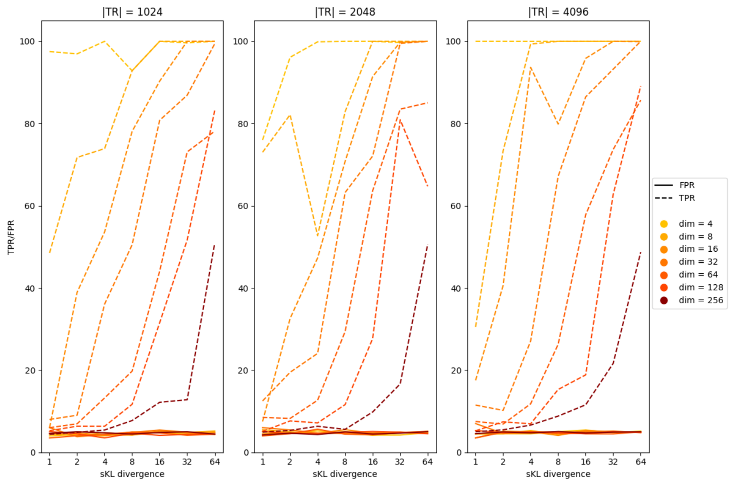


Figure 5.7: Another representation of Fig. 5.6: TPR and FPR obtained by QT over dataframes of increasing dimensionality with the divergence between pre- and post-change distributions $sKL = x$, given different data dimensionalities. The data was generated 5 times and each time TPR and FPR were computed over 512+512 test batches.

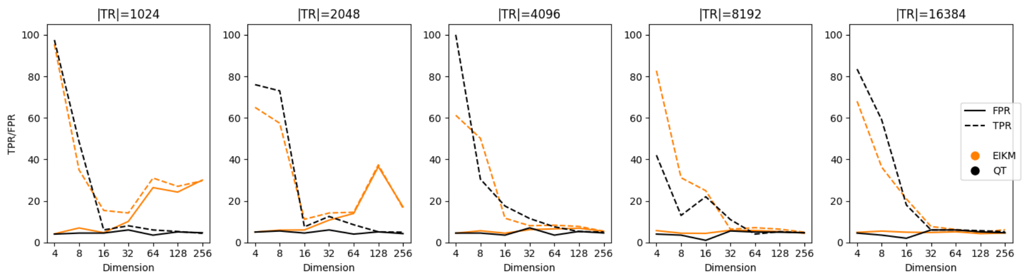


Figure 5.8: TPR and FPR achieved on data from monomodal gaussians with increasing dimensionality and $sKL = 1$ fixed, given training sets TR of increasing cardinality from left to right, by QT and EIKM algorithms. No preprocessing were applied to the dataframes. These experiments were repeated 10 times and TPR and FPR values are averaged over 500+500 test batches.

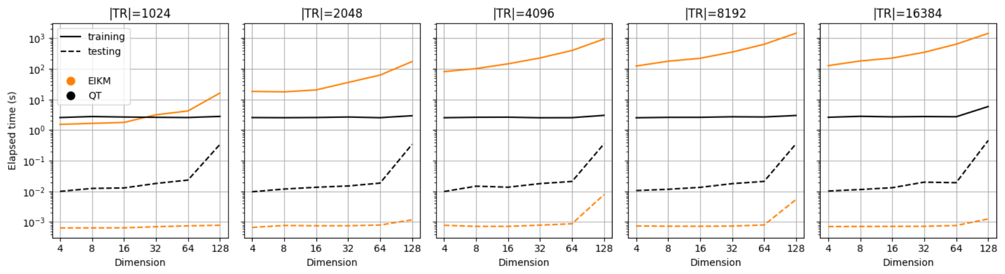


Figure 5.9: Training and testing time comparison between QT and EIKM given dataframes of different dimension d and different number of training points N. It is worth noting that QT processing time is independent from data dimensionality, and EIKM processing time is not. These experiments were repeated 10 times and TPR and FPR values are averaged over 500+500 test batches.

Δ(TPR-FPR). These experiments were repeated 10 times and TPR and FPR values are averaged over 500+500 test batches.

As we can see in Fig. 5.8, while in general EIKM cannot control the FPR at the 5% threshold, the performances are comparable with the ones of QT given a sufficient number of training data. Also the comparison of computational efficiency (Fig 5.9) is noteworthy. Despite QuantTree exhibiting a testing time approximately 10 times higher than EIKM, its processing time remains within the range of 10-100 milliseconds. On the other hand, EIKM demonstrates a consistently higher training time, reaching durations of 100-1000 seconds: its computation is slightly faster than QT's in small dimensional dataframes with a few training points, but QT's processing time is independent of data dimensionality while EIKM processing time is not.

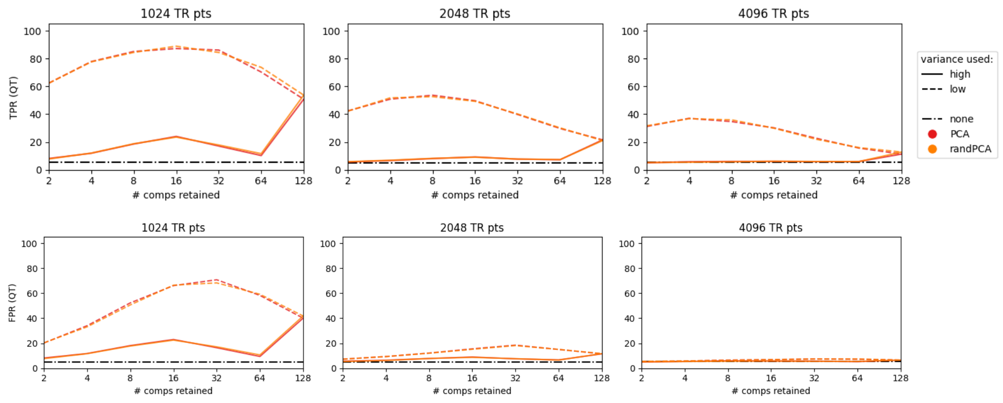
5.5 PCA and randomizedPCA

Our first study of the effect of dimensionality reduction over the performance of QuantTree and its variants was with Principal Components Analysis and its randomized version proposed in [12]. We proceeded on the same dataframe generated from monomodal gaussian distributions in $\mathbb{R}^d$ with $d \in \{2, 4, 8, 16, 32, 64, 128, 256\}$. Preprocessing was applied to select principal components and our models QuantTree and kQT with Mahalanobis and Weighted Mahalanobis distances were fed those. Data was generated and preprocessed, as we decided to run our CD algorithms on the first $d' \in \{2, 4, 8, 16, 32, 64, 128, 256\}$ principal components and on the last d' selected components, i.e. the ones explaining a minimal amount of the initial d-dimensional dataframe. As before, the post change distribution ϕ_1 was built such that the symmetric KL is unitary, and TPR and FPR were obtained from post-change and stationary batches respectively with QuantTree and kernel QT histograms with $K = 64$ bins built, in the latter case, using Mahalanobis and Weighted Mahalanobis distances. We separate the results obtained with Euclidean, Manhattan, and KQTs built on other l_p (quasi)norms. Again, the FPR desired value was set to $\alpha = 0.05$ and Pearson statistic was chosen for the hypothesis tests. The data was generated 20 times and each time TPR and FPR were computed over 1000+1000 test batches.

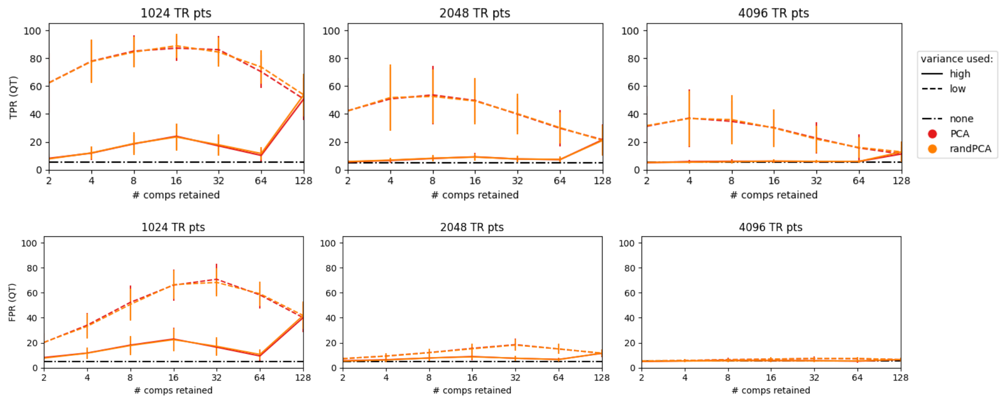
QuantTree

In Figure 5.10 are shown TPR (top) and FPR (bottom) measured with QuantTree on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \rightarrow \phi_1) = 1$ preprocessed with PCA and randomizedPCA, given a different number x of "principal" components retained. To link to the discussion([17],[22]) on whether it is better to keep principal components or the low-variance components (which are usually discarded after PCA), we notice that, when keeping low-variance components (dashed lines), QT achieves a better TPR but FPR increases likewise. The expectation that concept drifts may manifest in the "low variance components" is indeed intuitive, considering that a drift implies some change in the underlying data distribution. However, even points from the pre-change distributions may trigger an alarm. In fact, these points are projected onto low-variance components that are computed given an insufficient number of training points, thus are strongly biased. The effectiveness of low-variance components in detecting concept drift is intricately tied to the robustness of their computation, emphasizing the importance of an adequately sized and representative training set.

We notice that QT's control of the FPR is lost - if too few training points are drawn from a high-dimensional distribution - after a PCA-like rotation, even with no dimensionality reduction. The full-space based results in terms of TPR and FPR computed on datasets with increasing dimensionality $d \in \{4, 8, ..., 256\}$ are reported in Figure 5.11, where we can confront QT's outcomes after PCA rotation with no DR with respect to the ones on the same datasets not preprocessed.



(a)



(b)

Figure 5.10: **Keeping high variance or low variance components?** TPR (top) and FPR (bottom) measured with QuantTree on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \rightarrow \phi_1) = 1$ fixed. Data is preprocessed with PCA and the first $d' \in \{2, 4, 8, 16, 32, 64, 128 = d\}$ components are retained both from the top and the bottom in order of variance explained. We can observe that while low-variance components (dashed lines) achieve an higher TPR, QT cannot maintain the FPR anymore. Both TPR and FPR are compared with the reference dash-dot black line (QT performance on the same 128-dimensional dataset, with no preprocessing). It's important to notice that even if PCA preprocessed comes with no dimensionality reduction, FPR control is lost if an insufficient number of training points is provided. The expectation that concept drifts may manifest in the "low variance components" is indeed intuitive, considering that a drift implies some in the underlying data distribution. However, even points from the pre-change distributions may trigger an alarm. TPR and FPR are averaged over (512+512) test batches.

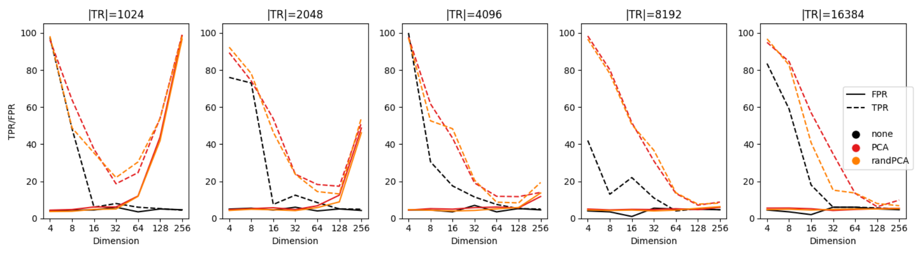


Figure 5.11: TPR and FPR achieved by QT on x-dimensional gaussian dataframes given $sKL = 1$ fixed. We can confront QT performance after PCA rotation with no DR with its performance on the same datasets with no preprocessing: if N is not large enough, the estimated principal components badly represents the original data distribution and FPR control is lost, as with Mahalanobis KQT. This bad behavior, which is almost unpredictable given different experiments with different dataframe, is one of the main point of this thesis work. We will study other PCA-like rotations and decompose PCA into different steps to try to understand what is the cause and what can be done to avoid falling into this difficult situation.

These experiments aim to show how PCA, a common preprocessing technique for high-dimensional data influences QT's behavior. QuantTree has shown robustness in maintaining a low FPR under fixed thresholds, even in challenging conditions; however a PCA preprocessing, even without dimensionality reduction, badly influences this QT's capability. In the scatterplot in figure 5.12, we again analyze the performance of the QT algorithm on datasets of increasing dimensionality, varying the number of training points N from 1024 to 16384, and compare the True Positive Rate (TPR) and False Positive Rate (FPR) under these different conditions, with and without PCA preprocessing. When the data undergoes no preprocessing, we know there is a noticeable decline in the TPR when dimensionality scales. However, the FPR consistently remains below the predefined threshold of $\alpha = 5\%$ independently of d, where traditional methods often struggle. After applying PCA rotation to the data, the False Positive Rate (FPR) of QT increases with the dimensionality (d) of the data. These changes might be influenced by the presence of outliers within the data; indeed, the effect is mitigated by an increased cardinality of the training set. In the context of the TPR/FPR scatterplot of fig. 5.12 (b), we observe that the larger dots, which correspond to higher dimensions of the dataframe, tend to shift closer to the diagonal line, i.e. as dimensionality increases FPR tends to approach TPR. Experiments conducted with a greater number of training points, represented by rose and violet dots in the scatterplot, exhibit a lower FPR. We will discuss other methods to mitigate this effect probably given by the presence of outliers, or by bad estimation of low-variance components of the original distribution.

Our experimental analyses will mostly explore this FPR, remarking some

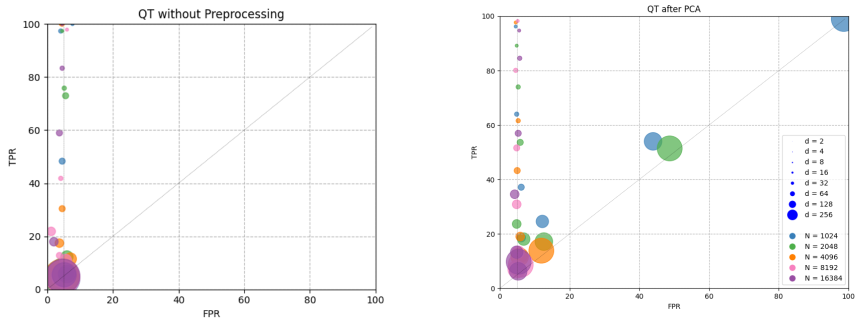


Figure 5.12: (a) QT generally excels in maintaining a low FPR. While PCA rotation generally improves detection capabilities, in specific scenarios with inconvenient N/d ratios, QuantTree's performance may be compromised, particularly regarding FPR control.

topics discussed in the concept drift detection literature, remembering that while TPR signifies a potent ability to detect changes, it is the controlled FPR that ensures the reliability. With QuantTree, where the thresholds (see Algorithm 2) of the test statistics can be numerically computed from univariate and synthetically generated data, the controlled FPR is guaranteed. This method not only is as robust as traditional methods such as the Pearson goodness-of-fit test and tests based on empirical thresholds computed through bootstrap, but even outperforms those in cases with limited training samples. Indeed, QT maintains this control across the entire spectrum of challenging scenarios with a few training points sampled from high-dimensional distributions. However, we acknowledge that data preprocessing may, at times, compromise that property.

PCA is associated to a *whitening transformation* (or sphering transformation). If X is a random vector with non-singular covariance matrix Σ and mean 0, then the transformation $Y = WX$ yields the whitened random vector Y with unit diagonal covariance when W is a whitening matrix. There are infinitely many possible whitening matrices satisfying the above condition: if $W = \Sigma^{-1/2}$ we talk about Mahalanobis whitening, while PCA whitening is given by the eigensystem of Σ. To select components, we are not using Σ, from which we generated the training and test datapoints, but the sample covariance matrix, which was estimated from the training set. Is this estimation, very poor from e.g. 1024 points lying in a 256-dimensional space, the cause of these results?

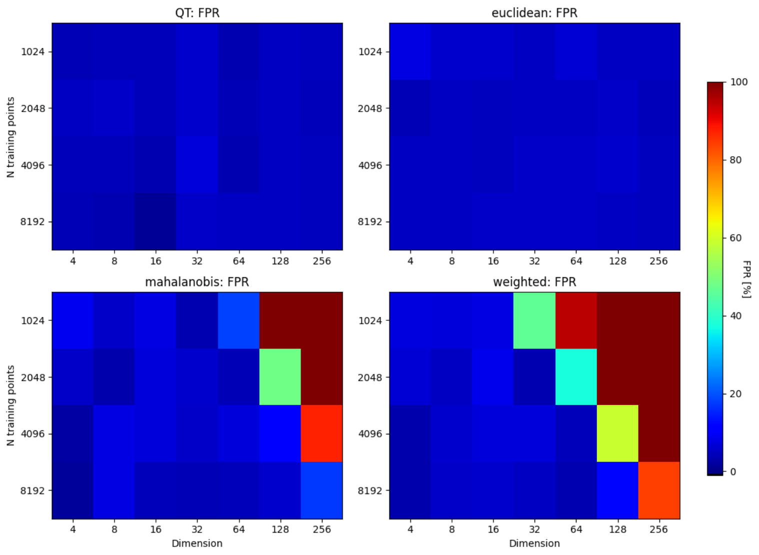
In the following experiments, we will try to understand and show the causes behind this issues, to refine the algorithms' applicability in diverse scenarios. Are there specific configurations or adaptations in the preprocessing pipeline that avoid the FPR control? We remark that where the algorithm's robustness falters, leading to an improved ability to detect changes at the expense of FPR control, it become pointless: if nearly each new data point is treated as a dire emergency, with both True Positive Rate (TPR) and False Positive Rate (FPR) consistently high, it is much like the boy who cried wolf ('al lupo al lupo'). The same discussion also considers Kernel-QuantTree, which partitions the space in $K-1$ compact bins defined by kernel functions evaluated on the training data, and estimates the sample covariance matrix when using the Mahalanobis and Weighted Mahalanobis distances.

Kernel QuantTree

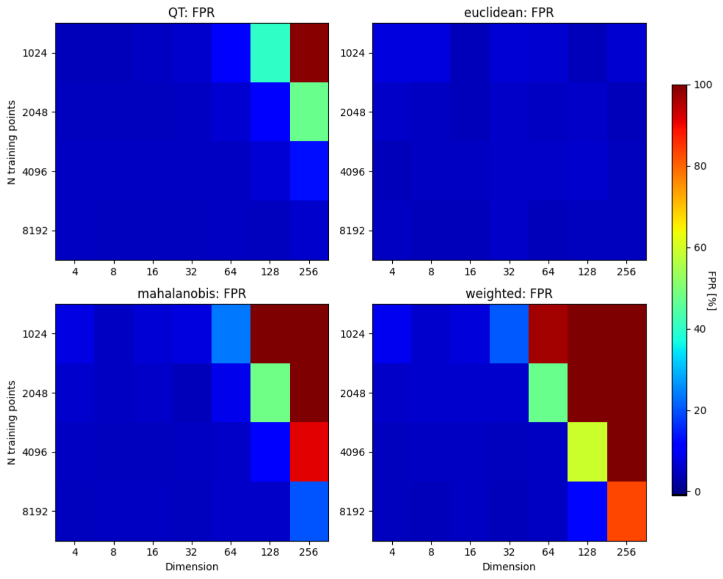
We remark a baseline experiment in Figure 5.13a; data was sampled from the same monomodal multivariate Gaussian distributions with zero mean in d dimensional spaces (dataframe dimensionality is represented on the x-axis). Different numbers N of training points (on the y-axis) were used to construct QuantTree histograms with 64 bins, which were then tested on batches drawn from the same distribution to compute the FP Rate. We have seen that, no matter neither the dimensionality of the dataframe nor the number of training points, both QuantTree and Kernel QuantTree based on the Euclidean distance between pair of points can control the FPR, i.e. can keep the number of false alarms around the user-specified value of 5%. In this same setup KernelQuantree based on Mahalanobis and Weighted Mahalanobis (WM) distances cannot maintain FPR at the desired value when data dimensionality increases, given that is not provided a proper number N of points in the training set. WM-KQT struggle to control the FPR when d increases was recognized and noted in [25] and explained as a consequences of the fact that, in high-dimensional settings, the estimated Gaussian Mixture Model (GMM) can become poorly conditioned if the training set TR is not sufficiently large. The authors proposed the use of KQT with the Mahalanobis distance in these cases when the GMM fit from TR yields Gaussian having covariances with large condition numbers. In the paper it is also affirmed that FPR control worsens in general when d increases, but that using a large training set heavily mitigates this problem. Moreover, it is noted that the issue can be avoided by employing dimensionality reduction techniques, such as PCA.

We plot again our mean FPR values in diverse N/d configurations as in Figure 5.13a. Comparing Figure 5.13a and 5.13b, we see that KQT performances are practically unchanged: it is proven [25] that, thanks to the kernel functions adopted, the KQT monitoring scheme is invariant to any roto-translation of the input data, while QT loses FPR control in unfavorable N/d ratios.

If we try to apply Principal Component Analysis to our datapoints sampled from the 256-dimensional gaussian distribution and keep a number x of principal components to pass to QuantTree, as we've already seen, it cannot control the FPR anymore. This is true if we decide to keep as "principal" components the directions explaining the greatest variance of the training set, and particularly if we consider "principal" components the ones explaining the lowest variance, the ones usually discarded. We can imagine that, since these "last" components were computed over a limited amount of points, might not truly explain a small amount of variance of the original gaussian distribution, that is: a new unseen test point might be far from its nearest training neighbors along this selected

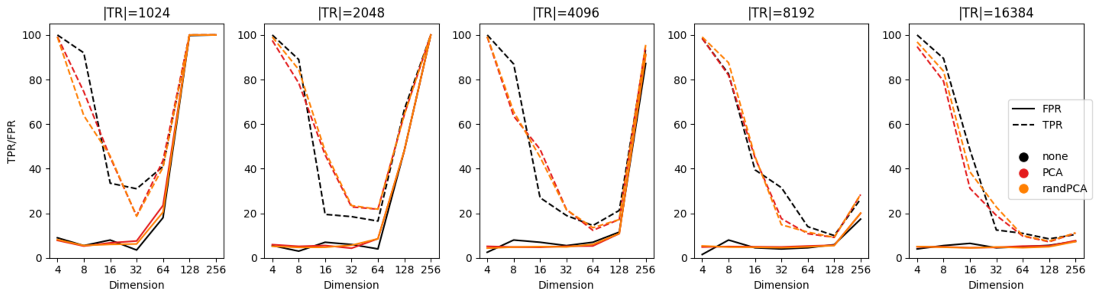


(a) FPR control over data drawn from a monomodal Gaussian distribution in x dimensions, given y training points. The "difficulty" for the CDD systems increases from bottom-left to upper-right. FPR desired value was set to 5%. Data is not preprocessed and it can be seen that QuantTree and Euclidean kQT can effectively control FPR, while kQT with Mahalanobis and Weighted Mahalanobis distances lose this property when trained on "few" points from high dimensional distributions.

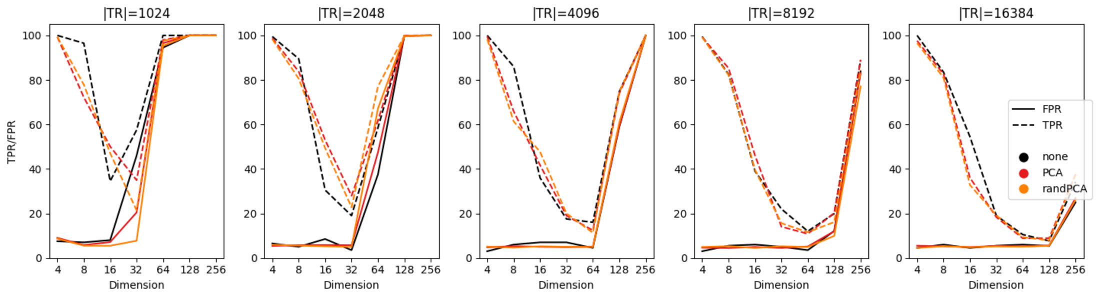


(b) FPR control over data drawn from a monomodal Gaussian distribution in x dimensions, given y training points. Data is preprocessed with PCA and it can be seen that QuantTree cannot control FPR anymore, with respect to Fig. 5.13a. KQT with Euclidean distance is able to maintain FPR control even in unsuitable frames with N/d small, and its performance is invariant to PCA-like rotation. We will focus on the usage of this distance and other distances derived from l_p norms to address concept drift detection in high dimensional dataframes.

Figure 5.13



(a) PCA: kQT with Mahalanobis distance



(b) PCA: kQT with weighted-Mahalanobis distance

Figure 5.14: TPR and FPR measured with kernel-QuantTree (based on Mahalanobis (a) and weighted-Mahalanobis (b) distances) on data from two d-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is preprocessed with PCA and its randomized version. Kernel-QT performance with these distances adopted does not depend on any roto-translation of the data.

low variance direction. This would trigger a false alarm, since the distribution is not changed. But what about the "real" PCs?

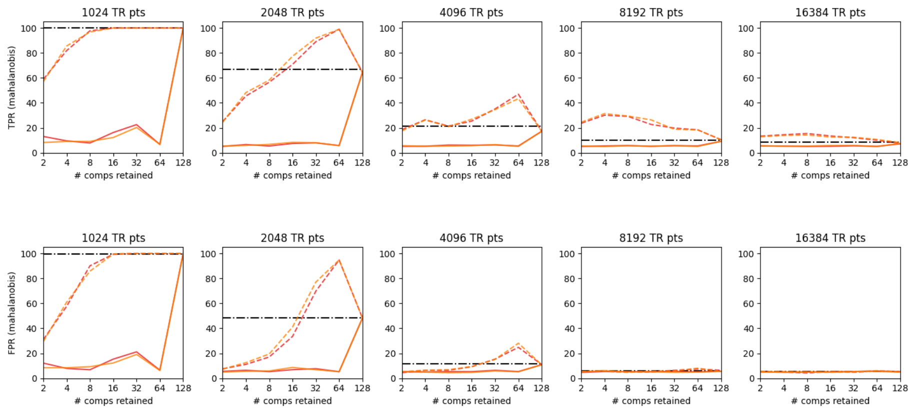
In Figure 5.15 are shown TPR (top) and FPR (bottom) measured with Kernel-QuantTree based on Mahalanobis (a) and Weighted-Mahalanobis (b) distances. Data batches are drawn from two 128-dimensional monomodal Gaussian distributions with $sKL(\phi_0 \rightarrow \phi_1) = 1$ preprocessed with PCA and randomizedPCA, given a different number x of "principal" components retained. Preprocessing helps in keeping the FPR consistently lower with respect to the reference (see KQT on the same 128-dimensional dataset with no preprocessing in a dash-dot black line); while low-variance components achieve an higher TPR, we always have to consider the FPR/TPR trade-off. Even if KQT performance are proved to be invariant under rototranslations, here PCA acts to reduce dimensionality, and helps in decreasing final average FPR; moreover, keeping a low number of components gives the best results in terms of the net detection power $\Delta(TPR - FPR)$.

5.6 RobustPCA

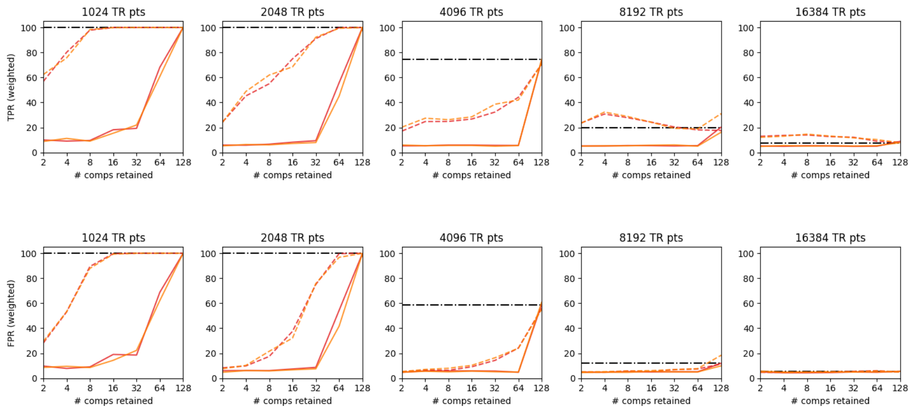
PCA preprocessing is used in infinitely-many applications in data processing, machine learning, etc. Our problem here is that we might have not enough training points to perform it correctly. Indeed, after a simple data projection on the prinipal components, QT loses its control over the FPR. To study the mechanisms that ruin its performance, to know if the problem is given by the presence of outliers in too small training sets, in the following experiment we tried RobustPCA preprocessing for our gaussian pre- and post-change distributions. We refer to Eq. 4.4. Here, the value for the minimization problem was set to $\lambda = 0.1$.

Preprocessed d-dimensional dataframes where given to QT, KQT and EIKM. The results are shown in Fig. 5.16 with no dimensionality reduction (only the PCA-like rotation). Experiments were repeated 5 times and FPR and TPR are averaged over (512+512) test batches. We see that in this settings, the PCA-like rotation alone ruins the models' performance, mostly bringing both the FPR and TPR up to 100%. We report results averaged over 100 experiments for (512+512) test batches each for QuantTree only in Fig. 5.17.

If we use this preprocessing as it is designed for, keeping only a subset of the given components, we find the results averaged over 100*(512+512) confrontations between PCA and robustPCA in Fig. 5.18. When keeping a small subset of high variance components computed over small training sets, FPR control seems to be better with robustPCA. But in these settings TPR always equals FPR. When projecting over low variance components, standard PCA seems to provide more robust coordinates.

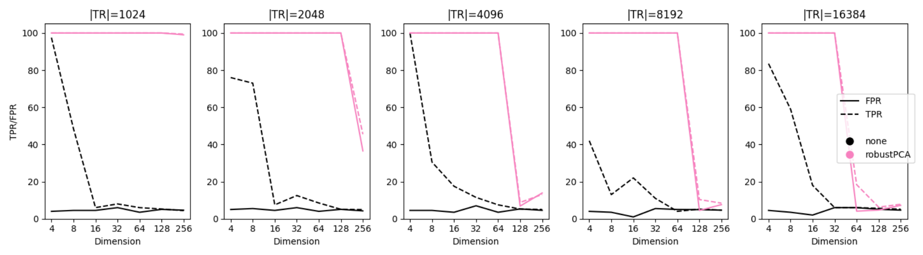


(a) PCA: kQT with Mahalanobis distance

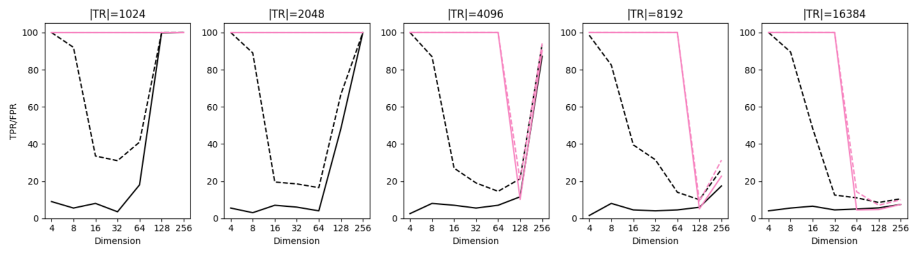


(b) PCA: kQT with weighted-Mahalanobis distance

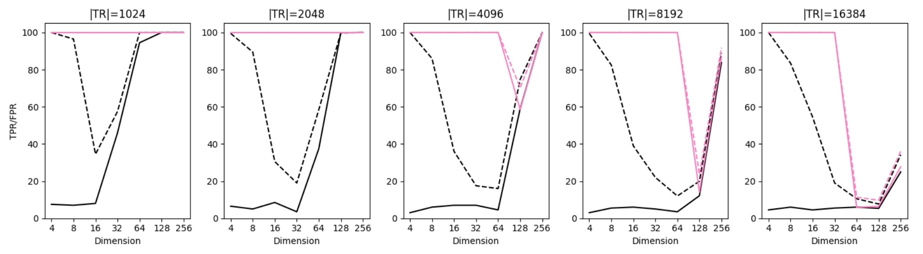
Figure 5.15: TPR and FPR measured with kernel-QuantTree (based on Mahalanobis (a) and weighted-Mahalanobis (b) distances) on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is preprocessed with PCA (red lines) and randomized-PCA (orange lines) and the first $d' \in \{2, 4, 8, 16, 32, 64, 128 = d\}$ components are retained both from the top (solid lines) and the bottom (dashed lines) in order of variance explained. We can observe that preprocessing helps in keeping the FPR lower than the reference (kQT performance on the same 128-dimensional dataset, with no preprocessing, in a dash-dot black line); while low-variance components achieve an higher TPR, we always have to consider the FPR/TPR trade-off. The data was generated 20 times and each time TPR and FPR were computed over 1000+1000 test batches.



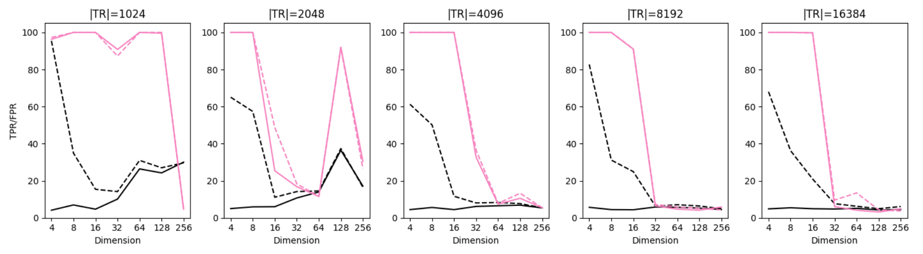
(a) QuantTree



(b) kQT with Mahalanobis distance



(c) kQT with weighted-Mahalanobis distance



(d) EIKM

Figure 5.16: TPR and FPR measured with QuantTree (a), kernel-QuantTree (based on Mahalanobis (b) and weighted-Mahalanobis (c) distances) and EIKM (d) on data from two d-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is preprocessed with robustPCA with $\lambda = 0.1$ (see Eq. 4.4) with no dimensionality reduction. Experiments were repeated 5 times and FPR and TPR are averaged over (512+512) test batches. It's only 5 experiments but we can obviously see that in this settings, the simple PCA-like rotation ruins the detection models performance, bringing both the FPR and TPR up to 100%.

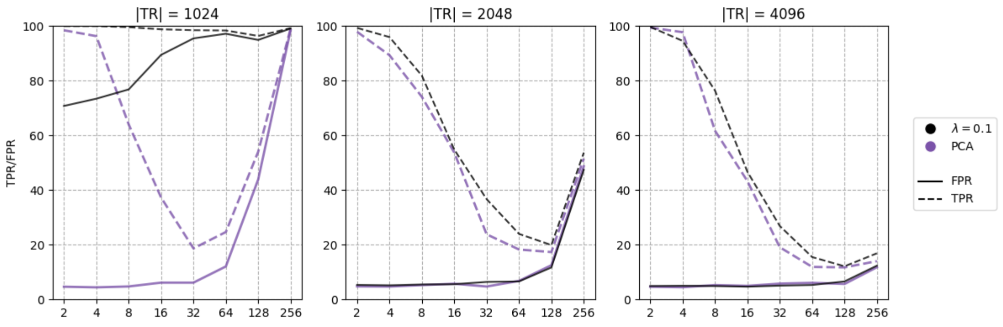
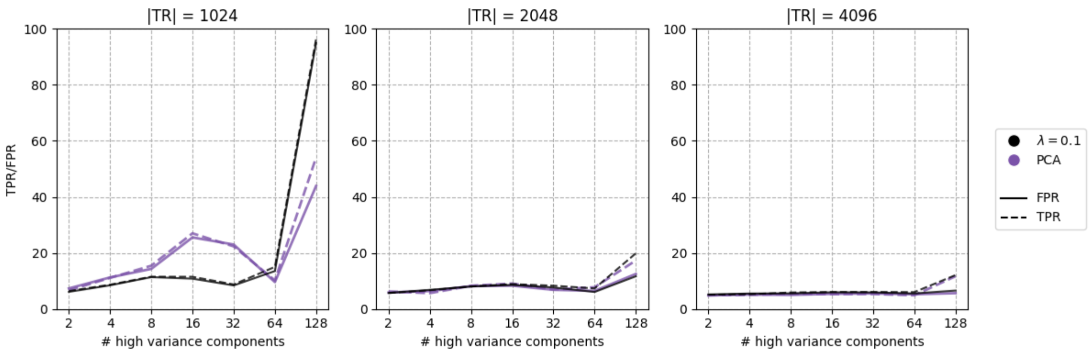


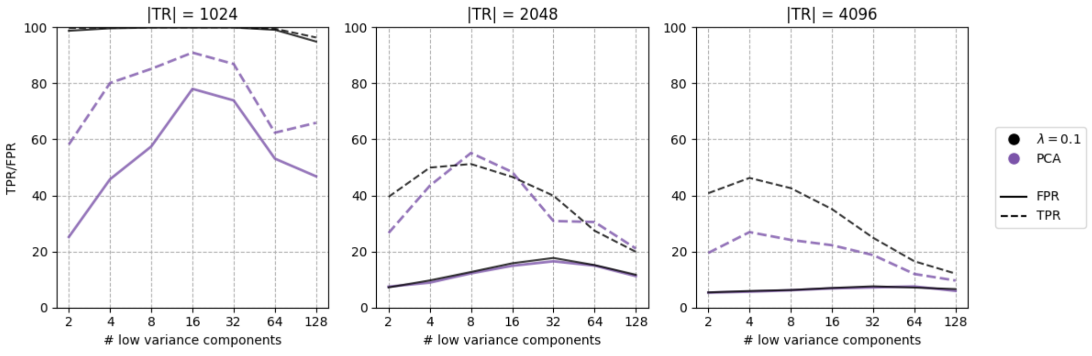
Figure 5.17: TPR and FPR achieved by QT on an x-dimensional Gaussian dataframe preprocessed with robustPCA (see Eq. 4.4) using $\lambda = 0.1$. A standard preprocessing with PCA is always better, considering the control over FPR. Experiments were repeated 100 times over (512+512) test batches.

5.7 Theoretical PCA

We discussed Principal Component Analysis (PCA) as a pivotal tool for dimensionality reduction and feature extraction. However, in our experiments for concept drift detection, we saw that traditional sample-based PCA approaches can exhibit vulnerabilities when applied to datasets characterized by outliers or a paucity of training data. To address this limitation, we introduced a "Theoretical" version of PCA which does not rely on training data but on the known distribution from which our dataframe is sampled. Instead of relying on the empirical training set sampled, we directly compute the principal components from the known parameters of the data's underlying Gaussian distribution. Indeed, known the covariance matrix, we derive the true principal components as its eigenvectors. Our aim was to answer the fundamental question: Is PCA genuinely failing, or are the limitations a byproduct of the training data? The results are presented in Fig. 5.19, and compare the performance of this "Theoretical PCA" against the conventional sample-based PCA method. It is worth noting that in real-life scenarios, we do not have complete knowledge of the true underlying distribution ϕ_0. Therefore, our approach here cannot be a practical solution for applications. This experiment serves as an essential step in discerning the root causes of PCA's performance issues with FPR control. It appears that while a PCA rotation may cause the loss of the FPR control, the covariance matrix-based projections in this approach have the capability to enhance the detection power of QT method in terms of True Positive Rate (TPR). Notably, this improvement in TPR is achieved without causing the FPR to exceed the predefined threshold of 5%. These results suggest that the underlying issue contributing to the observed challenges in using sample-based PCA is likely linked to a suboptimal estimation of the original data distribution. Specifically, the presence of outliers appears to be a key factor affecting the performance of the conventional PCA approach. These conclusions align with the evidence that a substantial increase



(a)



(b)

Figure 5.18: (a) FPR achieved by QT on a 128-dimensional dataframe preprocessed with robustPCA, keeping first x principal components (high variance components). If a sufficient number of training points are provided, FPR achieved is comparable with the one obtained after PCA and does not exceed the $\alpha = 5\%$ threshold. FPR control seems to be better with robustPCA when N is small, as expected. (b) Keeping the last x components to project data points. A robustPCA preprocessing achieves higher TPR values for $N = 4096$, but seems to be less robust with respect to simple PCA with small training sets, in terms of FPR control. Experiments were repeated 100 times and FPR and TPR are averaged over (512+512) test batches.

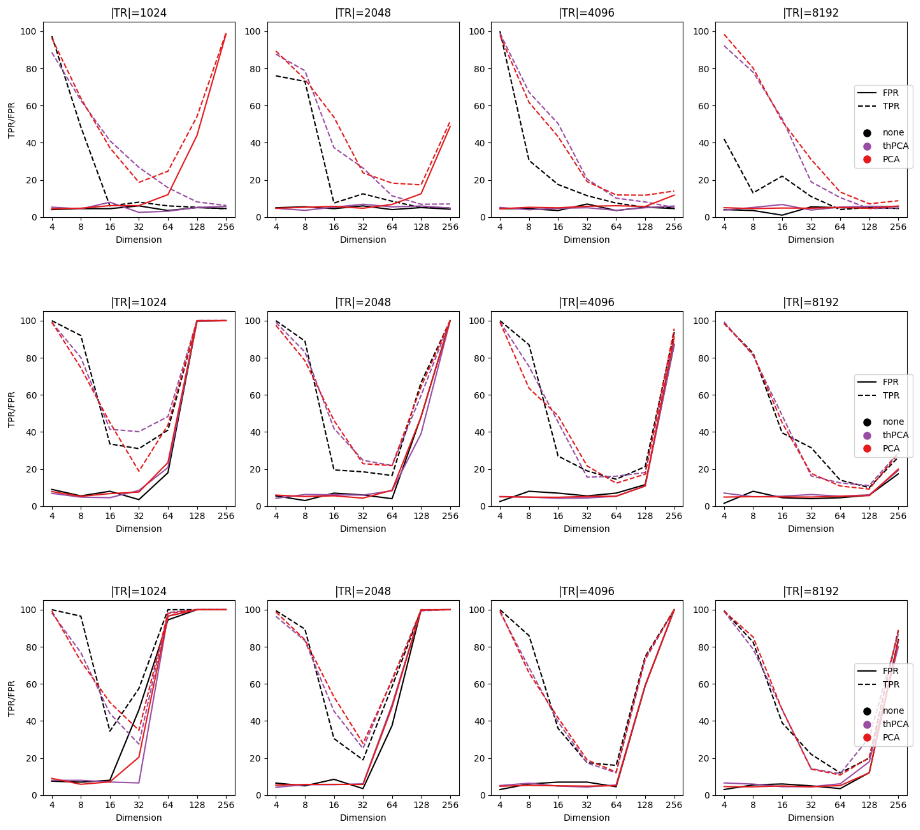


Figure 5.19: TPR (top) and FPR (bottom) measured with QuantTree and kernelQT on data from two x-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is projected on the eigenvectors of the covariance matrix of the original distribution ϕ_0 and concept drift detection performance are presented in violet, compared with the results after a typical sample-based PCA transformation trained on TR (in red). We can see that, as PCA, thPCA does not change the performance of kernel-QT with Mahalanobis and weighted-Mahalanobis distances. On the other side, while a PCA rotation prevents QT to control the FPR, this covariance matrix-based projections increases QT detection power (TPR) while not increasing FPR above the set threshold $\alpha = 5\%$. These experiments were repeated 5 times and FPR and TPR are averaged each time over (128+128) test batches.

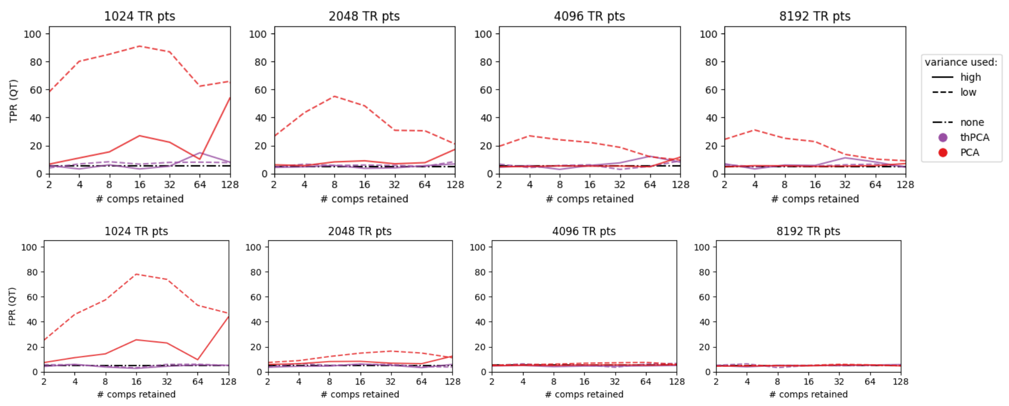


Figure 5.20: TPR (top) and FPR (bottom) measured with QuantTree on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is projected on the eigenvectors of the covariance matrix of the original distribution ϕ_0 and concept drift detection performance are presented in violet, compared with the results after a typical sample-based PCA transformation trained on TR (in red). We can see that, as PCA, thPCA does not change the performance of kernel-QT with Mahalanobis and weighted-Mahalanobis distances. On the other side, while a PCA rotation prevents QT to control the FPR, this covariance matrix-based projections increases QT detection power (TPR) while not increasing FPR above the set threshold $\alpha = 5\%$. These experiments were repeated 5 times and FPR and TPR are averaged each time over (128+128) test batches.

in the number N of training points is promoting effective control over the FPR. In Fig. 5.20 we see that with the covariance matrix-based projections the FPR control is achieved even reducing dimension. However, the detection power stays within the range of the one obtained after no preprocessing. We will investigate the effects of increasing the divergence between ϕ_0 and ϕ_1 when comparing these different dimensionality reduction methods and concept drift detectors all together. These experiments were repeated 5 times and FPR and TPR are averaged each time over (128+128) test batches. To study QT behavior more robustly, instead, we averaged TPR and FPR results over 100 experiments given (512+512) test batches each time, as shown in Fig. 5.21.

5.8 Sample-based PCA dissection

In this experimental investigation, the objective was to dissect the various components of Principal Component Analysis (PCA) and discern their individual impacts on the efficacy of a concept drift detection algorithm, particularly focusing on False Positive Rate (FPR) control. PCA generally comprises two fundamental steps: centering the data by subtracting the mean of each feature

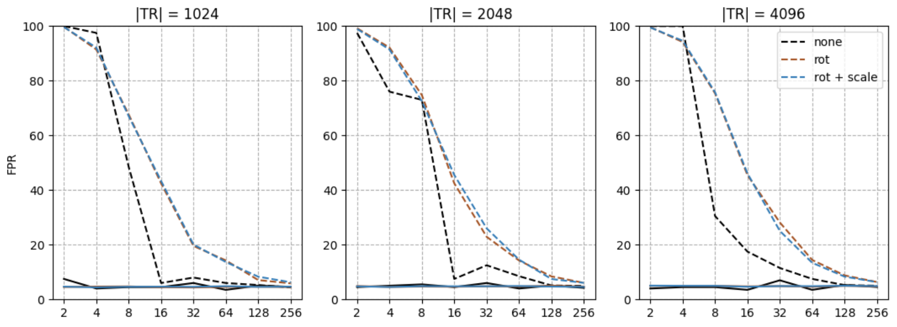


Figure 5.21: TPR and FPR achieved by QT on an x-dimensional Gaussian dataframe preprocessed with theoretical-PCA (as we refer to the projection over the eigenvectors of the true covariance matrices the dataframes belong to). Data is preprocessed twice: projected on the eigenvectors of the covariance matrix, or projected and then scaled by the eigenvalues. Results are given by the mean over 100 experiments where TPR and FPR are averaged on (512+512) test batches each time.

and projecting the centered data onto eigenvectors derived from the covariance matrix. The squared root of the corresponding eigenvalues represents the explained variance of each component. When dealing with high-dimensional data and limited training data points, a simple projection onto all principal components, as we have seen, is sufficient to disrupt the control of concept drift detection algorithms like QuantTree. Data projection over the principal components, which can be seen as a rotation when no dimensionality reduction is employed, enhances the detection power of QuantTree by increasing True Positive Rate (TPR). This enhancement can be attributed to the ability of PCA to capture and accentuate the principal directions of variation in the data, making it more sensitive to changes indicative of concept drift. However, in cases involving high-dimensional datasets and an insufficient number of training points, the application of PCA, even without dimensionality reduction, compromises the control of FPR.

There is a closely related preprocessing step called *whitening* (or sphering) which is needed for some algorithms. The goal of whitening is to make the input less redundant; more formally, our desiderata are that our learning algorithms sees a training input where (i) the features are less correlated with each other, and (ii) the features all have the same variance. This can be achieved simply rescaling each feature already projected on the principal direction with the square root of the eigenvalue corresponding to that direction, i.e.:

$$x_i^{(r)} = \mathbf{x} \cdot \mathbf{v}_i$$

$$x_i^{(w)} = \frac{x_i^{(r)}}{\sqrt{\lambda_i}}$$

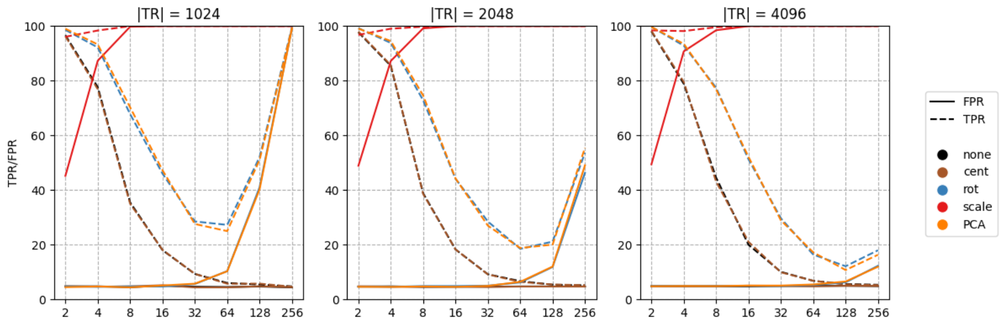


Figure 5.22: TPR and FPR achieved by QT on an x-dimensional Gaussian dataframe preprocessed with different steps of sample-based PCA. QuantTree performance are invariant under data centering (any rigid translation of the dataframe). Data is centered (mean of each feature on the training set is removed from test batches), then is projected on the eigenvectors of the covariance matrix. Our data projection correspond to the PCA implementation in scikit-learn Python library (labelled as PCA). If we scale the data with the square root of the eigenvalues of the sample covariance matrix, FPR control is lost and both FPR and TPR goes to 100%. The data was generated 100 times and each time TPR and FPR were computed over 512+512 test batches.

From 5.22 we can see that whitening the data with the eigenvalues of the sample covariance matrix makes FPR control lost even in low-dimensional frames.

In 5.23 we show the effect of dimensionality reduction in the same cases, i.e. QT's performance over data preprocessed by centering the data, projecting it on principal components, and scaling it by the corresponding eigenvalues, considering a 128-dimensional dataframe. We can see that the best performance are obtained given a high N/d ratio of course, and projecting the data on a small number of components (between 4 and 8).

5.9 Exploring sensitivity to PCA with convex combinations

In our exploration of the impact of Principal Component Analysis (PCA) on concept drift detection with QuantTree, we introduced several experiments using the concept of convex combinations. That is, data matrix is projected by some matrix M such that $M = cA + (1 - c)B$ with c ranging from 0 to 1.

5.9.1 From the original data to Principal Components

In the first try, combinations are represented by the equation $\lambda * M_{PCA} + (1 - \lambda) * \mathbf{1}$, where M_{PCA} represents the matrix transformation associated with PCA while $\mathbf{1}$ is the identity matrix. By varying the value of λ in the range

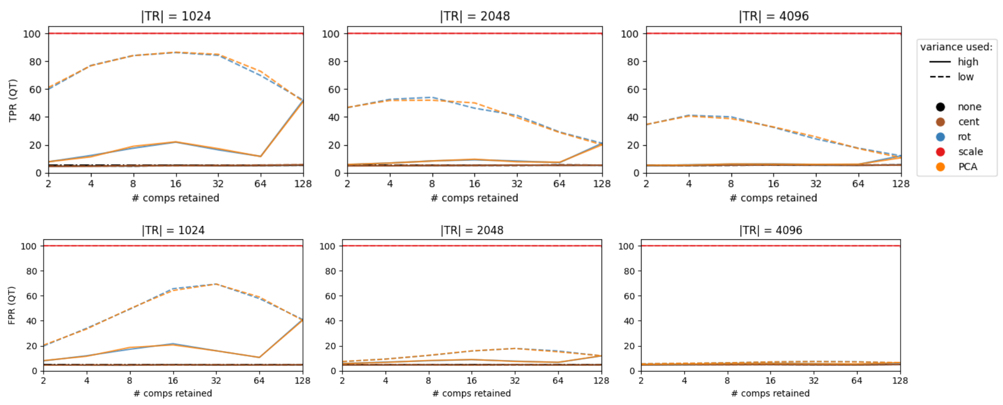


Figure 5.23: TPR (top) and FPR (bottom) measured with QT on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is centered (mean of each feature on the training set is removed from test batches), then is projected on the eigenvectors of the covariance matrix. Our data projection correspond to the PCA implementation in scikit-learn Python library (labelled as PCA). Dimensionality reduction is performed as we keep x high or low variance components out of the 128-dimensional data batches. The data was generated 100 times and each time TPR and FPR were computed over 512+512 test batches.

$(0, 1)$, we systematically examined the effects transformation applied to the data. The outcomes of this experiment are shown in Figure 5.24; we did not exploit dimensionality reduction but only performed a PCA-like rotation of the dataframe. As λ increases, we observe a gradual shift away from the identity matrix (ID) (no preprocessing given $\lambda = 0$) towards PCA transformation ($\lambda = 1$). As we expected, this shift is accompanied by a noticeable deterioration in the control over the False Positive Rate (FPR). In other words, higher values of λ emphasize PCA's influence on the data, and this emphasis appears to compromise the ability to maintain effective FPR control. This observation underscores the sensitivity of FPR to dimensionality reduction methods and offers a compelling perspective on the challenges of balancing dimensionality reduction with concept drift detection.

5.9.2 From theoretical to sample-based PCA

Here, combinations are represented by the equation $c * M_{PCA} + (1 - c) * M_{th}$, where M_{PCA} represents the matrix transformation associated with PCA while M_{th} is the matrix associated with theoretical-PCA, i.e. containing the eigenvectors of the covariance matrix of the Gaussian distribution. By varying the value of c in the range $(0, 1)$, we systematically examined the effects transformation applied to the data. The outcomes of this experiment are shown in Figure 5.25;

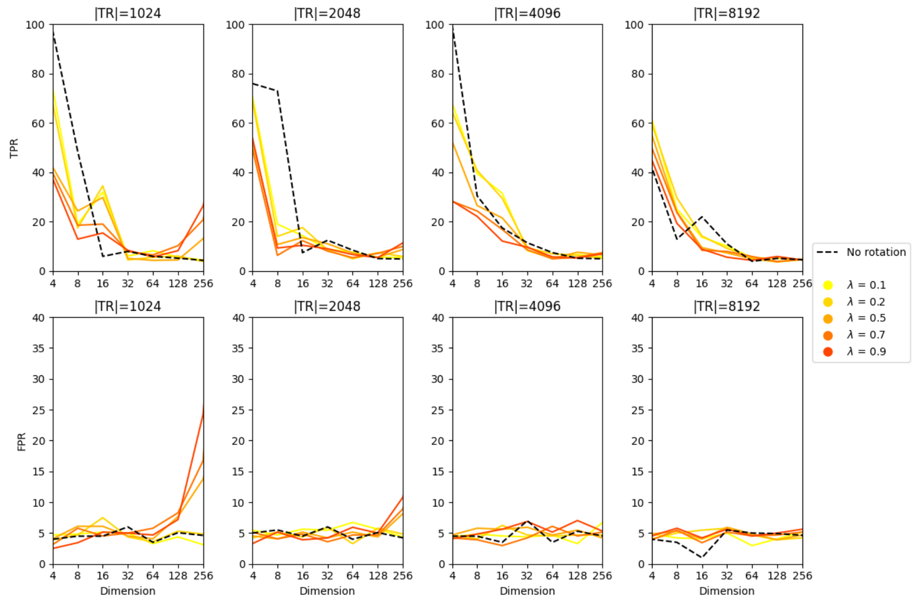


Figure 5.24: TPR and FPR achieved by QT on the usual d-dimensional gaussian dataframe given $sKL(\phi_0 \to \phi_1) = 1$ fixed. Dataframe is preprocessed with a PCA-like rotation matrix given by convex combinations $\lambda * M_{PCA} + (1-\lambda) * \mathbf{1}$, where M_{PCA} represents the matrix transformation associated with PCA, $\mathbf{1}$ is the identity matrix, and $\lambda \in \{0.1, 0.2, 0.5, 0.7, 0.9\}$. Higher values of λ emphasize PCA's influence on the data, and this emphasis appears to compromise the ability to maintain effective FPR control. Experiments were repeated 10 times, TPR and FPR are averaged each time on (128+128) test batches.

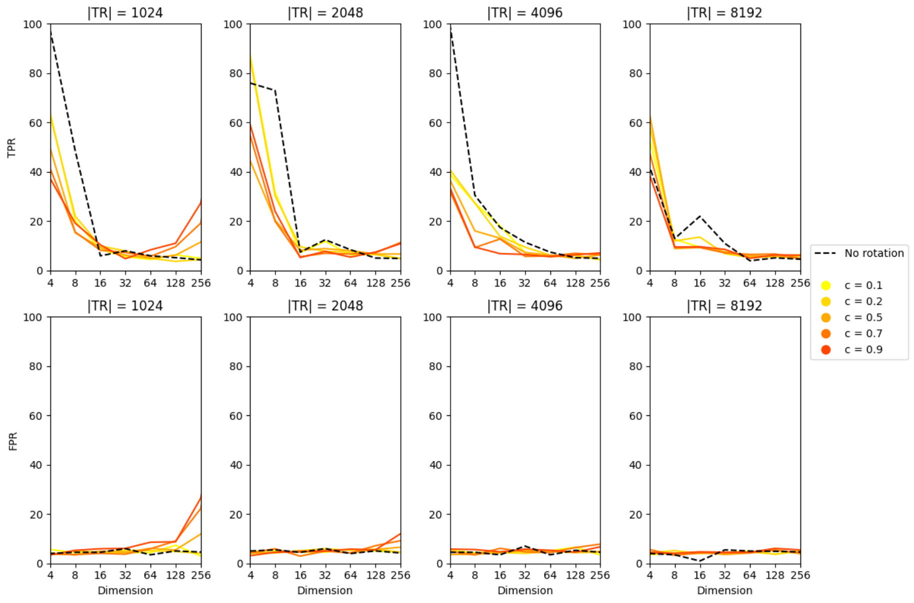


Figure 5.25: TPR and FPR achieved by QT on the usual d-dimensional gaussian dataframe given $sKL(\phi_0 \to \phi_1) = 1$ fixed. Dataframe is preprocessed with a PCA-like rotation matrix given by convex combinations $c * M_{PCA} + (1 - c) * M_{th}$, where M_{PCA} represents the matrix transformation associated with PCA, M_{th} is the identity matrix associated with theoretical-PCA, and $c \in \{0.1, 0.2, 0.5, 0.7, 0.9\}$. Higher values of c emphasize PCA's influence on the data, and this emphasis appears to compromise the ability to maintain effective FPR control as seen in Fig. 5.24. Experiments were repeated 10 times, TPR and FPR are averaged each time on (128+128) test batches.

we did not exploit dimensionality reduction but only performed a PCA-like rotation of the dataframe. As c increases, we observe a gradual shift away from good results provided by theoretical PCA, which require to know the data distribution, towards sample-based PCA transformation. As we expected, this shift is accompanied by a noticeable deterioration in the control over the False Positive Rate (FPR). In other words, higher values of c emphasize PCA's influence on the data, and this emphasis appears to compromise the ability to maintain effective FPR control as we discussed in the last experiment.

5.10 Rotating away from PCs

One hypothesis behind the effects of PCA over the control of the FPR, even with no dimensionality reduction, was around the alignment of our algorithm's histogram construction with the principal components. We thought that this alignment may introduce unintended consequences, especially in scenarios where data points collapse onto particular components (projection on low variance components). This alignment potentially obstructs the intended control over FPR, despite the algorithm's reliability in the original high-dimensional space. To investigate this hypothesis, an experiment is designed to assess the impact of data rotation on FPR control. Specifically, it involves computing the principal components, projecting the data onto them, and subsequently applying a d-dimensional rotation to these projected data points, systematically varying the rotation angle. This variation aims to observe how the control of FPR evolves as the direction of the histogram bins drifts away from the principal components. If this experiment proves successful, it may offer an elegant solution. Utilizing PCA for dimensionality reduction, which has intrinsic benefits, while subsequently applying a rotation to disalign the histogram bins from the principal components, could potentially preserve FPR control and enhance the algorithm's robustness in high-dimensional concept drift detection. The outcomes of this experiment are shown in Figure 5.26; we did not exploit dimensionality reduction but only performed a PCA-like rotation of the dataframe. Regrettably, it seems that the data rotation after PCA preprocessing, does not preserve the control of the False Positive Rate (FPR); indeed, there is no obvious trend of the results with varying α.

5.11 Changing the number of bins

An important parameter of our experiment is the number of bins used to construct the histograms. In each experiment we described, it was fixed to $K = 64$, but how can this be enough to fill up d-dimensional spaces with $d > 64$? QuantTree models ϕ_0 by a histogram made of K bins $\{S_j\}_{j=1}^{k}$ constructed by splitting $\mathbb{R}^d$ along random directions. K is a fundamental parameter that influences the change-detection performance of QT, and if $K < d$, we're anyway performing some random projections along the basis which defines our space.

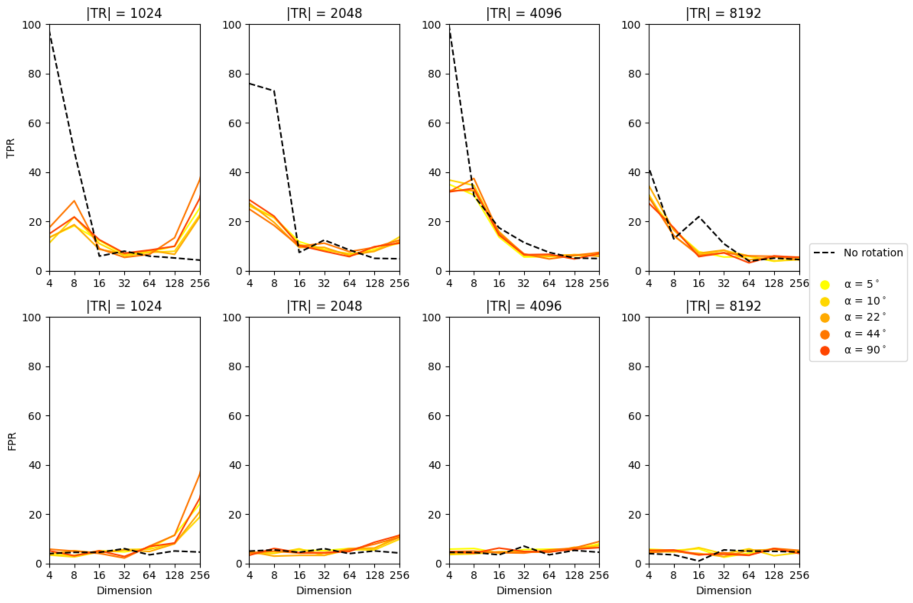


Figure 5.26: TPR and FPR achieved by QT on the usual d-dimensional gaussian dataframe given $sKL(\phi_0 \to \phi_1) = 1$ fixed. Dataframe is preprocessed with PCA and then rotated under each of the d axes by an increasing angle $\alpha \in \{5°, 10°, 22°, 44°, 90°\}$. Higher values of α should reduce PCA's influence on the data, which appears to compromise the ability to maintain effective FPR control as seen in Fig. 5.24. Experiments were repeated 10 times, TPR and FPR are averaged each time on (128+128) test batches.

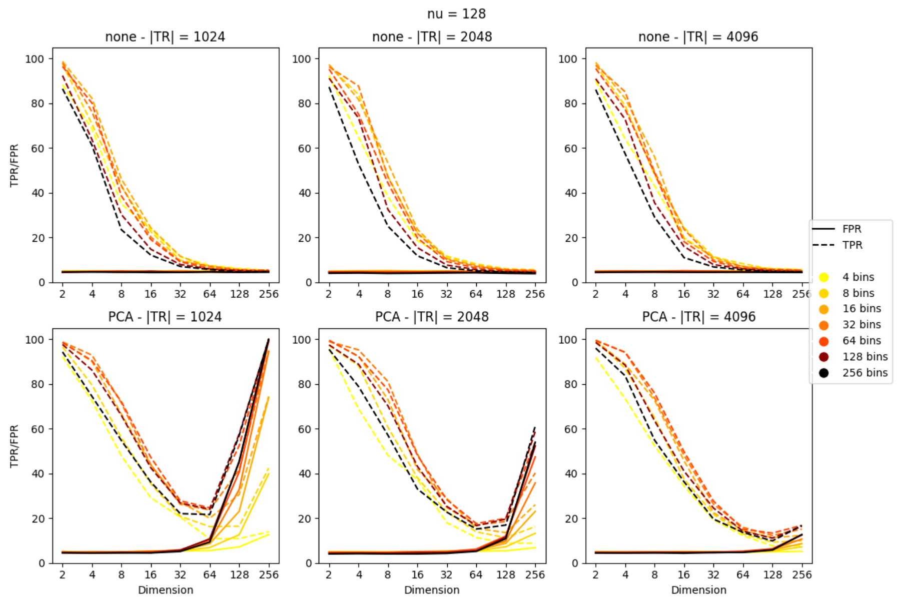
To analyze the impact of this choice, we test QT with $K \in 4, 8, 16, 32, 64, 128, 256$, computing the resulting TPR and FPR over $(512 + 512)$ pre- and post-change batches averaged over 100 experiments, both with and without PCA dimensionality reduction.

In contrast with histograms based on regular grids, the number of bins K in QT is a priori defined, and does not need to grow exponentially with d. From 5.27 we can see that: **i**) without any PCA preprocessing, the highest TPR is given, on average by histograms constructed over 32 bins; anyway, we must consider the value of K together with the number of points in the training set N and also with the cardinality of each test batch ν; **ii**) when applying a PCA rotation without dimensionality reduction, FPR control holds, on average, at higher dimensions with a smaller number of bins.

From 5.28 and 5.29 we can see that: **i**) again, the highest TPR is given, on average by histograms constructed over 32 bins; usually the best way to control FPR after a PCA-like rotation estimated from the training set is to keep a small number of components; **ii**) best detection power in terms of $\Delta(TPR - FPR)$ is always given when keeping the low variance components, although they are associated to higher FP rates. It's worth noting that, on average, with fewer bins it is less probable to build a bin on a low variance component computed over the training set.

5.12 Between the explained variance and FPR: bad components

To understand the intricate dynamics of False Positive Rate (FPR) control in the presence of preprocessing techniques like Principal Component Analysis (PCA), we propose also a punctual analysis. Through 100 experiments conducted across datasets of varying dimensions, we can see that as the dimensionality of the datasets increases, not only does FPR surge to 100%, but the eigenvalues associated with the last principal components steadily approach zero. This pattern is shown in Figures 5.30 and 5.31. Could the presence of principal components with vanishing eigenvalues be a contributing factor to the high FPR, rendering them susceptible to false alarms in the concept drift detection process? This question echoes a discourse in the literature where contrasting perspectives have been presented. In 2014, Kuncheva asserted that components with the lowest variance should be retained, positing that they are more likely to be sensitive to changes [17]. However, in 2015, Qahtan offered an alternative viewpoint, emphasizing the benefits of discarding principal components with negligible eigenvalues [22]. Qahtan argued that these components, often representing variances of minuscule magnitude, introduce challenges in density estimation and comparison due to their sensitivity to sample size and model parameters. The result, as Qahtan suggested, could be an increased likelihood of false alarms during change detection. Several key questions should be explored. Are these vanishing eigenvalues indicative of components that are indeed sensitive



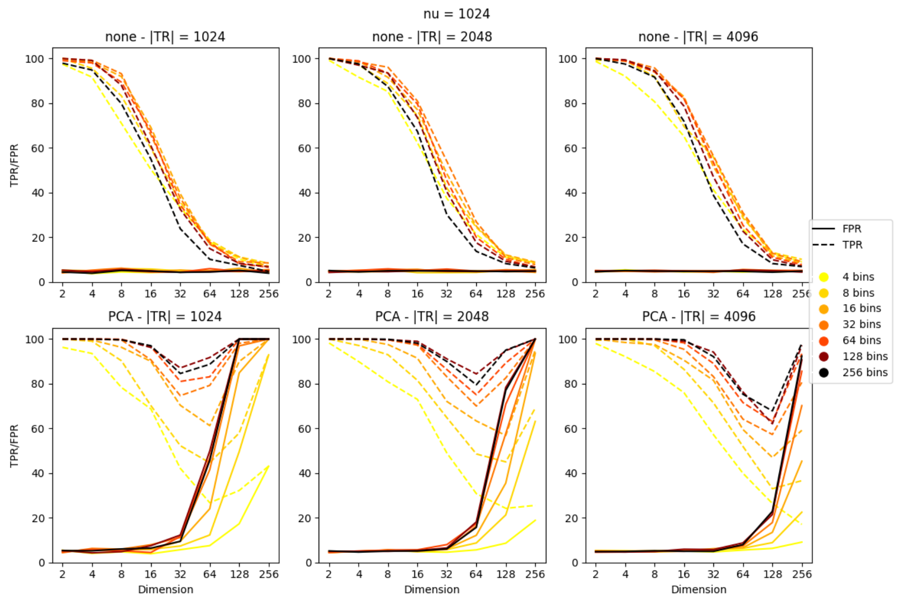
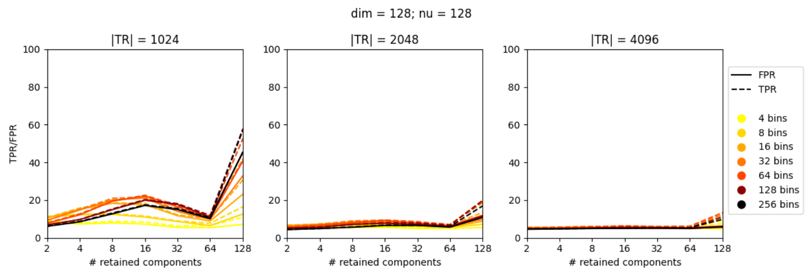
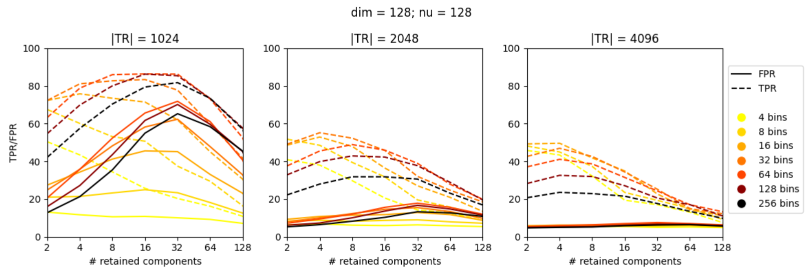


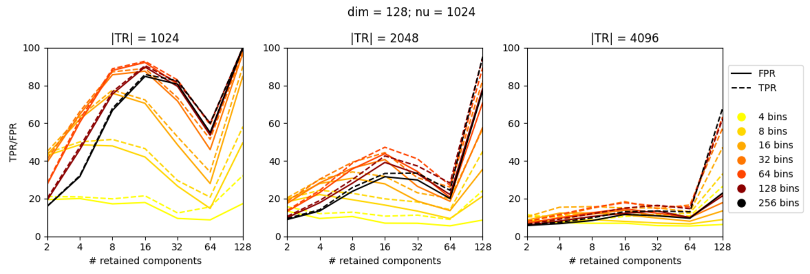
Figure 5.27: TPR and FPR measured with QT on data from two x-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is not preprocessed or preprocessed with PCA with no dimensionality reduction. The number of bins $K \in 4, 8, 16, 32, 64, 128, 256$ is represented by a color while the number of points per batch is fixed to $\nu = 128$ (a) or $\nu = 1024$ (b). The data was generated 100 times and each time TPR and FPR were computed over 512+512 test batches.



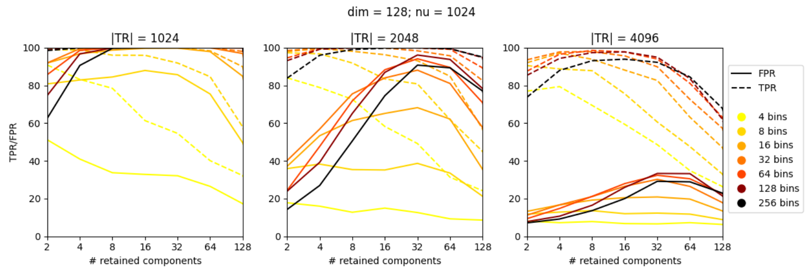
(a) $\nu = 128$, high variance components kept



(b) $\nu = 128$, low variance components kept

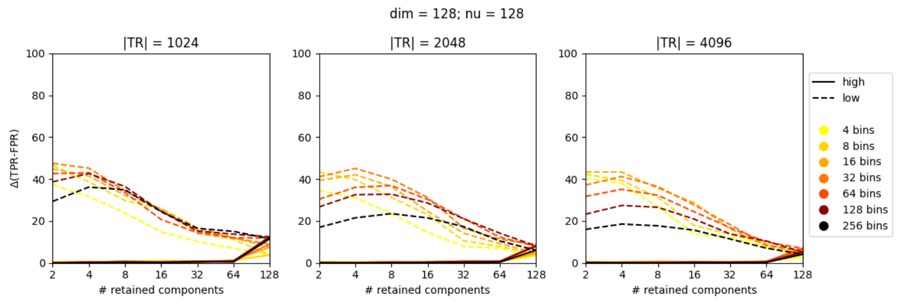


(c) $\nu = 1024$, high variance components kept

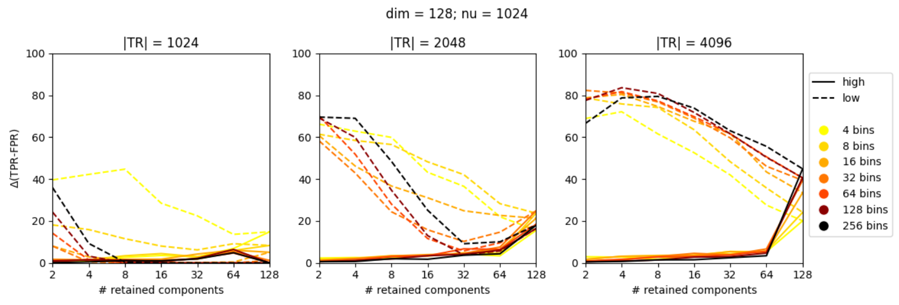


(d) $\nu = 1024$, low variance components kept

Figure 5.28: TPR and FPR measured with QT on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is preprocessed with PCA with dimensionality reduction $128 = d \to d' = x$. The number of bins $K \in \{4, 8, 16, 32, 64, 128, 256\}$ is represented by a color while the number of points per batch is fixed to $\nu = 128$ (a,b) or $\nu = 1024$ (c,d). Respectively high variance (a,c) and low variance (b,d) components are kept and given to QuantTree. The data was generated 100 times and each time TPR and FPR were computed over 512+512 test batches.



(a)



(b)

Figure 5.29: Δ(TPR-FPR) measured with QT on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \rightarrow \phi_1) = 1$ fixed. Data is preprocessed with PCA with dimensionality reduction $128 = d \rightarrow d' = x$. The number of bins $K \in \{4, 8, 16, 32, 64, 128, 256\}$ is represented by a color while the number of points per batch is fixed to $\nu = 128$ (a) or $\nu = 1024$ (b). Respectively high variance (solid lines) and low variance (dashed lines) components are kept and given to QuantTree. The data was generated 100 times and each time TPR and FPR were computed over 512+512 test batches.

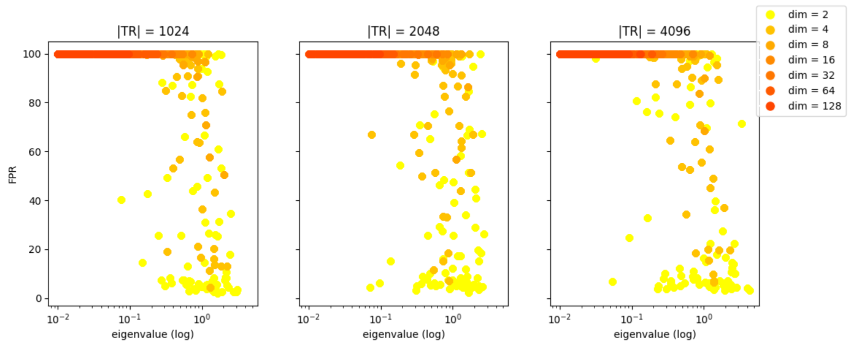
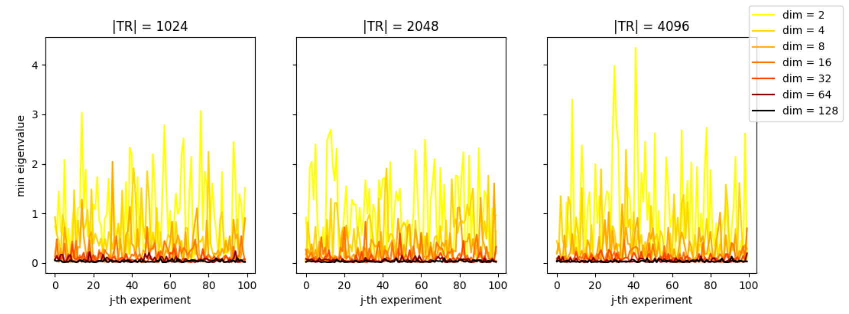


Figure 5.30: Exploring the Eigenvalue-FPR Relationship: we present the results of 100 experiments conducted across datasets of varying dimensions. Each data point in the scatterplot represents the lowest eigenvalue (in logarithmic scale) of the experiment and the associated False Positive Rate (FPR) observed in the concept drift detection process. As dimensionality increases, eigenvalues approach zero and FPR escalates; a part from this expected behavior, there seems to be no other correlation between the two quantities.

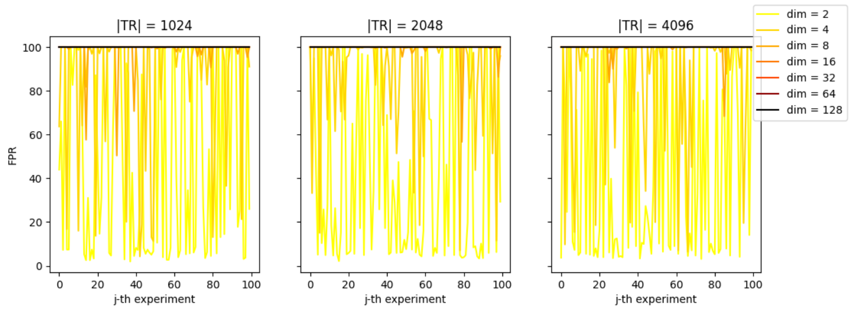
to changes? How does the choice of principal components affect the overall robustness of the concept drift detection system? Proposed experiments and calculations may involve assessing the impact of retaining or discarding principal components with vanishing eigenvalues on FPR control and the algorithm's detection capabilities. It may also be insightful to examine the density estimation process and its sensitivity to changes in the presence of such components.

5.13 Random Projections

While PCA serves as a widely used technique for dimensionality reduction, its application in concept drift detection warrants a closer examination. PCA operates by maximizing the variance captured by each principal component, which can inadvertently alter the data distribution characteristics. Given that we practically lack knowledge of the original data distributions, PCA can lead to unintended consequences, particularly, as we have discussed, when dealing with high-dimensional data and a limited number of training points. In such scenarios, the control over the False Positive Rate (FPR) may be lost. That is why we turned to random projections as an alternative approach for dimensionality reduction: offering the advantage of reducing dimensionality without imposing any specific ordering of 'high' or 'low' variance components, they are a valuable candidate for concept drift detection as they avoid the potential variance-related issues associated and provide a robust solution when dealing with limited training data points. Reducing data dimension d to $d' < d$ by projecting the original input space on a randomly generated matrix where components are drawn from the distribution $N(0, \frac{1}{d'})$, RPs offer a remarkably simplified solution. When



(a)



(b)

Figure 5.31: Exploring the Eigenvalue-FPR Relationship: In this fiery visualization, we present the results of 100 experiments conducted across datasets of varying dimensions. Each data point on the graph (a) represents the lowest eigenvalue and the associated False Positive Rate (FPR) observed in the concept drift detection process is reported in (b). As dimensionality increases, eigenvalues approach zero, and FPR escalates.

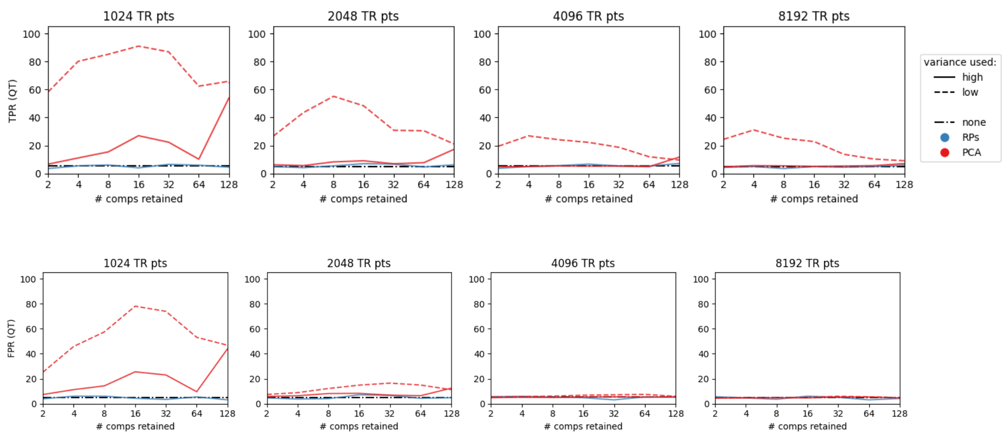
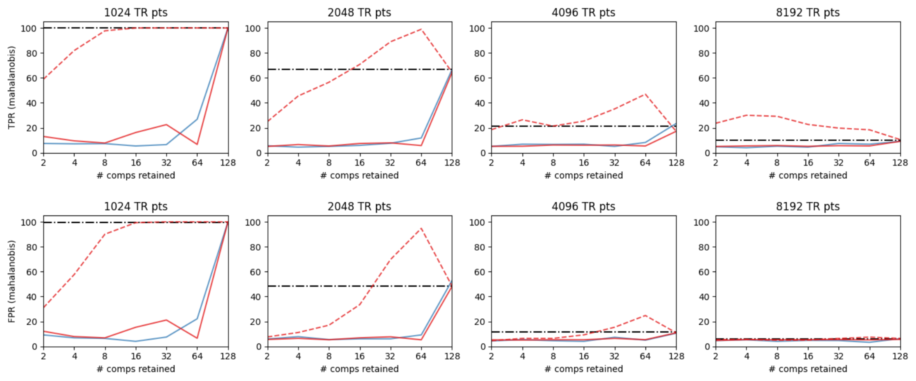


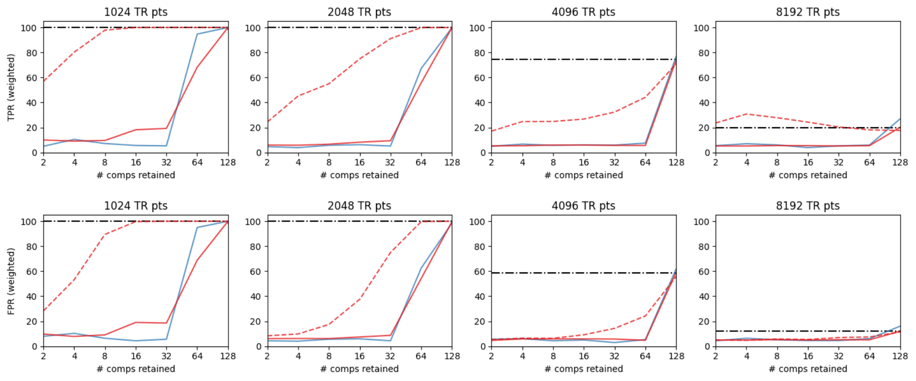
Figure 5.32: TPR (top) and FPR (bottom) measured with QuantTree on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \rightarrow \phi_1) = 1$ fixed. Data dimension is reduced from $d = 128$ to $d' \in \{2, 4, 8, 16, 32, 64, 128 = d\}$ by projecting the original input space on a randomly generated matrix where components are drawn from the distribution $N(0, \frac{1}{d'})$. We can see the obtained scores compared with the ones obtained with PCA dimensionality reduction and with the reference dash-dot black line (QT performance on the same 128-dimensional dataset, with no preprocessing). We notice that the FPR control is achieved, but with this distance between pre- and post-change distribution, QT algorithm as no power in detecting the drift (both TPR and FPR goes to 5%. These experiments were repeated 5 times and FPR and TPR are averaged each time over (128+128) test batches.

employing PCA, the process typically includes the computation of principal components, variance maximization, and consideration of ranking, all of which can be computationally intensive, particularly in high-dimensional datasets. In contrast, RPs require no such computational overhead. They operate by mapping the original data onto a randomly generated matrix, where the components are drawn from a specified distribution: the process essentially boils down to computing dot products in the lower-dimensional space significantly reducing the computational burden.

QT performances after Random Projections are shown in Fig. 5.32, juxtaposed with those obtained through PCA dimensionality reduction. Additionally, the reference dash-dot black line represents QuantTree's performance on the unaltered 128-dimensional dataset, without any preprocessing. Interestingly, the outcomes suggest that while FPR control is successfully achieved, even if the QuantTree algorithm loses detection power: with sKL = 1 fixed, both TPR and FPR goes to the set threshold $\alpha = 5\%$. In this framework there seems to be no difference in QT performances with or without preprocessing. For what



(a) kernel-QuantTree with Mahalanobis distance



(b) kernel-QuantTree with weighted-Mahalanobis distance

Figure 5.33: TPR (top) and FPR (bottom) measured with kernel-QuantTree (with Mahalanobis (a) and weighted-Mahalanobis (b) distances) on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data dimension is reduced from $d = 128$ to $d' \in \{2, 4, 8, 16, 32, 64, 128 = d\}$ by projecting the original input space on a randomly generated matrix where components are drawn from the distribution $N(0, \frac{1}{d'})$. We can see the obtained scores compared with the ones obtained with PCA dimensionality reduction and with the reference dash-dot black line (QT performance on the same 128-dimensional dataset, with no preprocessing). kernel-QT performances are definitely comparable with the ones obtained after PCA preprocessing keeping d' high-variance components; it's worth noting that the computational burden with RPs is way lower than the one to perform PCA. These experiments were repeated 5 times and FPR and TPR are averaged each time over (128+128) test batches.

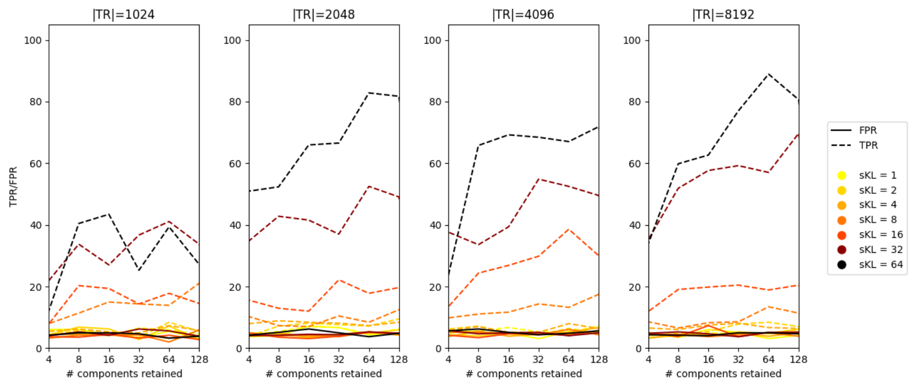


Figure 5.34: TPR and FPR achieved by QT on 128-dimensional gaussian dataframes given $sKL \in \{1, 2, 4, 8, 16, 32, 64\}$. As expected, as we generate post-change distribution ϕ_1 with sKL values greater than 1, TPR or the detection power increases. It is worth noting that FPR does not depend on the post-change distribution ϕ_1 as it measures alarms raised from batches drawn from ϕ_0 by definition. These experiments were repeated 5 times and FPR and TPR are averaged each time over (128+128) test batches.

matters kernel-QT, instead, results after RPs preprocessing seems in line with the ones obtained keeping high-variance components after PCA. If dimensions are effectively reduced from $d = 128$ to $d' < d$, the FPR control is achieved, but concept drift detection power diminishes. It's worth noting that the computational burden with RPs is way lower than the one to perform PCA. We will go through these performance differences and similarities increasing the divergence between pre- and post-change distributions.

5.13.1 Increasing sKL with random projections

QT performances after Random Projections were shown in Fig. 5.32, juxtaposed with those obtained through PCA dimensionality reduction. Additionally, the reference dash-dot black line represents QuantTree's performance on the unaltered 128-dimensional dataset, without any preprocessing. Interestingly, the outcomes suggest that while FPR control is successfully achieved, even if the QuantTree algorithm loses detection power: with sKL = 1 fixed, both TPR and FPR goes to the set threshold $\alpha = 5\%$. In this framework there seems to be no difference in QT performances with or without preprocessing. But if we try to increase the divergence between the two distributions, we can see how random projections successfully work. It is worth noting that FPR does not depend on the post-change distribution ϕ_1 as it measures alarms raised from batches drawn from ϕ_0 by definition. We show the increasing detection power in Fig. 5.34.

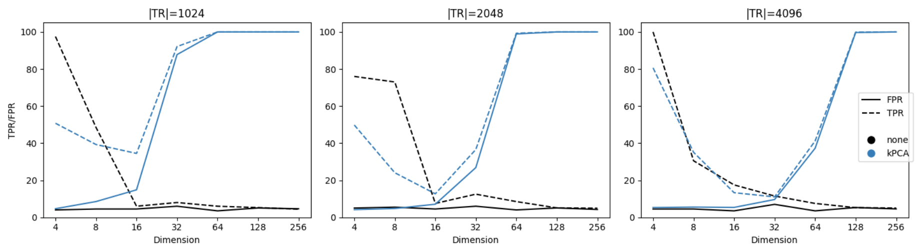
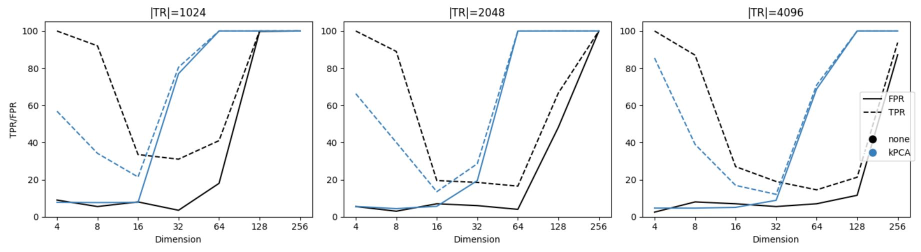


Figure 5.35: QT's TPR and FPR measured on data from monomodal gaussians with increasing dimensionality and $sKL = 1$ fixed, given training sets TR of increasing cardinality from left to right. Even with not dimensionality reduction, also a kernel-PCA rotation is sufficient for QT to lose control over the FPR (reference without preprocessing in black). A greater number of training points can delay this phenomenon. The data was generated 20 times and each time TPR and FPR were computed over 1000+1000 test batches.

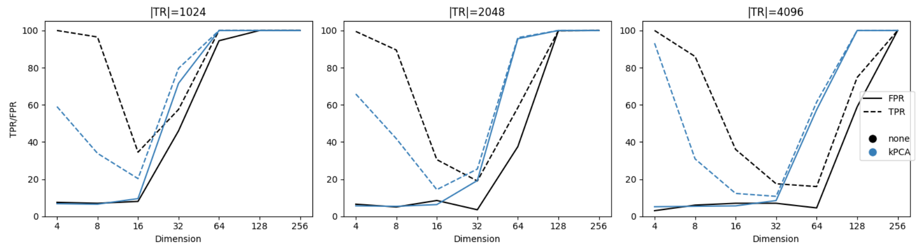
5.14 Kernel-PCA

Next choice in the exploration of dimensionality reduction techniques was kernelPCA with radial basis kernel functions, as an alternative to standard Principal Components Analysis. Again, the dataframe was generated from monomodal gaussian distributions in $\mathbb{R}^d$ with $d \in \{2, 4, 8, 16, 32, 64, 128, 256\}$. Preprocessing was applied to select principal components and our models QuantTree and kQT with Mahalanobis and Weighted Mahalanobis distances were fed those. Data was generated and preprocessed, as we decided to run our CD algorithms on the first $d' \in \{2, 4, 8, 16, 32, 64, 128, 256 = d\}$ principal components and on the last d' selected components. As before, the post change distribution ϕ_1 was built such that the symmetric KL is unitary, and TPR and FPR were obtained from post-change and stationary batches respectively with QuantTree and kernel QT histograms with $K = 64$ bins built, in the latter case, using Mahalanobis and Weighted Mahalanobis distances. Again, the FPR desired value was set to $\alpha = 0.05$ and Pearson statistic was chosen for the hypothesis tests. The data was generated 20 times and each time TPR and FPR were computed over 1000+1000 test batches.

In the pursuit of effective dimensionality reduction techniques for enhancing the performance of concept drift detection algorithms, the exploration turned towards kernelPCA with radial basis kernel functions as a viable alternative to the standard PCA, since it can capture nonlinear structure in the data. As we can see, it might not be the optimal choice here. It is worth noting that PCA generally has lower memory and runtime requirements than kPCA, and can be scaled to massive datasets.



(a) kQT with Mahalanobis distance



(b) kQT with weighted-Mahalanobis distance

Figure 5.36: TPR and FPR measured with kernel-QuantTree (based on Mahalanobis (a) and weighted-Mahalanobis (b) distances) on data from two d-dimensional monomodal gaussian distributions with $sKL(\phi_0 \rightarrow \phi_1) = 1$ fixed. Data is preprocessed with kernelPCA with radial-basis kernel functions. We notice that even with no dimensionality reduction, this preprocessing worsen the kQT control over FPR. The data was generated 20 times and each time TPR and FPR were computed over 1000+1000 test batches.

5.15 Independent Component Analysis

We tried Independent Component Analysis (ICA) as a method to separate our multivariate signal into additive subcomponents (also called factors or latent variables). Whitening ensures that all dimensions are treated equally a priori before the algorithm is run. It is worth noting that our purpose here is not dimensionality reduction: dimensionality of the output is the same as the input's, there is no obvious way to drop components as in PCA and its variants. We are transforming our dataset in a maximally independent set of components. In Figure 5.37 are reported the performances of QuantTree (a) and kernel-QuantTree (based on Mahalanobis (b) and weighted-Mahalanobis (c) distances) in detecting the drift $\phi_0 \to \phi_1$ with the distributions ϕ_0, ϕ_1 monomodal gaussians in x-dimensional spaces, with and without ICA preprocessing. From Fig. (a) we can see that the detection algorithm's power is increased by ICA, as the difference Δ(TPR-FPR) increases for each of the x-dimensional dataframes. However, QT loses the control over the FPR after the ICA preprocessing if an insufficient number of training points are provided. For what we can see in (b), (c), this preprocessing leaves kQT performances unchanged. Experiments were repeated 10 times and FPR and TPR are averaged over (500+500) test batches.

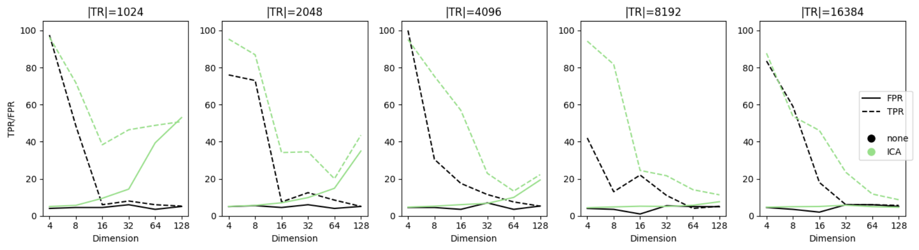
5.16 Euclidean KQT

In [25] it is proven that KQT is invariant under rototranslations when the employed kernel function is either the Mahalanobis or Weighted-Mahalanobis distance. This holds also when the Euclidean distance is employed. The usage of the Euclidean distance results in isotropic bins, which poorly fit the data distribution with respect to the Mahalanobis and Weighted Mahalanobis distances. On the other side, the computation of the histogram is much faster.

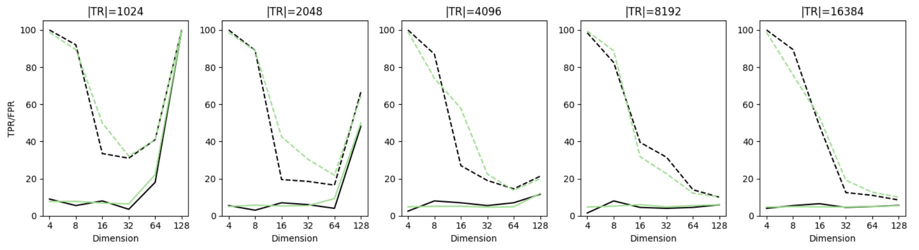
We explore the behavior of KQT with the Euclidean distance, and examine its performance under various conditions, including dimensionality changes and PCA rotations. We think and show that this implies that there will be no issue with the FPR control, even if the ratio N/d is not convenient, i.e. with few training points from high dimensional spaces.

We can see how PCA rotation not affects the behavior of Euclidean kQT: the algorithm's FPR control remains intact when employing the Euclidean distance (see Fig. 5.39 and 5.40), emphasizing its invariance under roto-translations. However, if the only PCA rotation ensures kQT's control over the FPR (invariance under rototranslations) when employing the Euclidean distance, a dimensionality reduction from d to $d' < d$ can ruin this property if an insufficient number of training points is given. In particular, FPR control is lost when we decide to keep low-variance components from high dimensional dataframes, if these are trained on too few training points, as we show in Fig. 5.41.

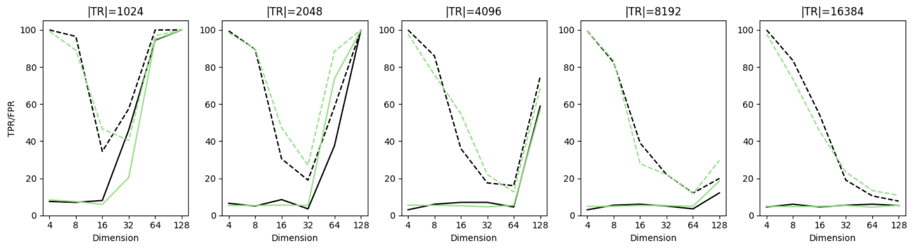
As we discussed at the beginning of the chapter, the Euclidean distance (l_p norm with $p = 2$) might not be the best choice when dealing with high-dimensional dataframes because of the curse of dimensionality. We are going to



(a) QuantTree



(b) kQT with Mahalanobis distance



(c) kQT with weighted-Mahalanobis distance

Figure 5.37: TPR and FPR measured with QuantTree (a) and kernel-QuantTree (based on Mahalanobis (b) and weighted-Mahalanobis (c) distances) on data from two d-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is preprocessed with Independent Component Analysis (ICA). We notice that while this preprocessing does not change kQT performances, it increases QT's TPR at the expenses of worsening its control over FPR. Experiments were repeated 10 times and FPR and TPR are averaged over (500+500) test batches.

Figure 5.38: TPR and FPR achieved by QT and kQT with Euclidean and Mahalanobis distances on x-dimensional gaussian dataframes given $sKL = 1$. We can notice that, as it is with QT, FPR control holds when employing the Euclidean distance, independently from the number of training points and the data dimension. Experiments were repeated 10 times, TPR and FPR are averaged each time on (256+256) test batches.

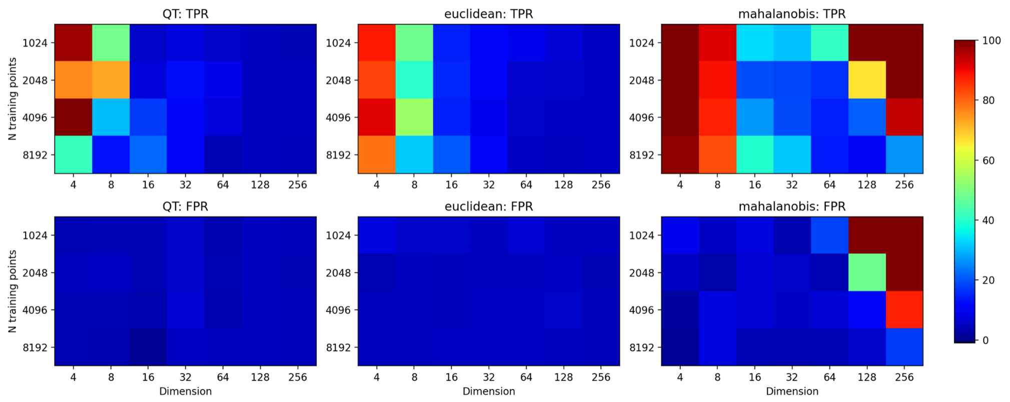


Figure 5.39: TPR and FPR achieved by QT and kQT with Euclidean and Mahalanobis distances on x-dimensional gaussian dataframes given a number y of training points and $sKL = 1$ fixed. We can notice that FPR control holds when employing the Euclidean distance, independently from the number of training points and the data dimension. This property, which comes from QuantTree space partitioning, is not true employing Mahalanobis distance or weighted-Mahalanobis distance, but holds with the Euclidean choice. Experiments were repeated 10 times, TPR and FPR are averaged each time on (256+256) test batches.

try with different choices like the Manhattan distance (l_p norm with $p = 1$) and also fractional distance metrics where ($p < 1$) in the following experiments.

5.17 l_p distances with $p \leq 1$

In *On the Surprising Behavior of Distance Metrics in High Dimensional Space* [1] it is examined the behavior of l_p norms and shown that the problem of meaningfulness in high dimensionality is sensitive to the value of p. The authors explain that, under some assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target in high dimensional space is almost unitary for a wide variety of data distributions and distance functions: the nearest neighbor problem here is ill defined. Specifically, it is shown that the l_1 distance metric (Manhattan Distance metric) is the most preferable for high dimensional applications, followed by the euclidean distance (l_2 metric), the l_3 metric, ... knowing this, the authors studied fractional distance metrics (where $p < 1$) and showed that indeed these are even more effective at preserving the meaningfulness of proximity measures. It is worth noting that the Euclidean distance (l_2 norm) is invariant under roto-translations. While each norm defines a translation invariant metric, this is not true for rotations; e.g., the Manhattan distance is in general dependent from rotations (if these are different from a $\pi/2$ rotation). That is, KQT with these metrics generally loses its invariance

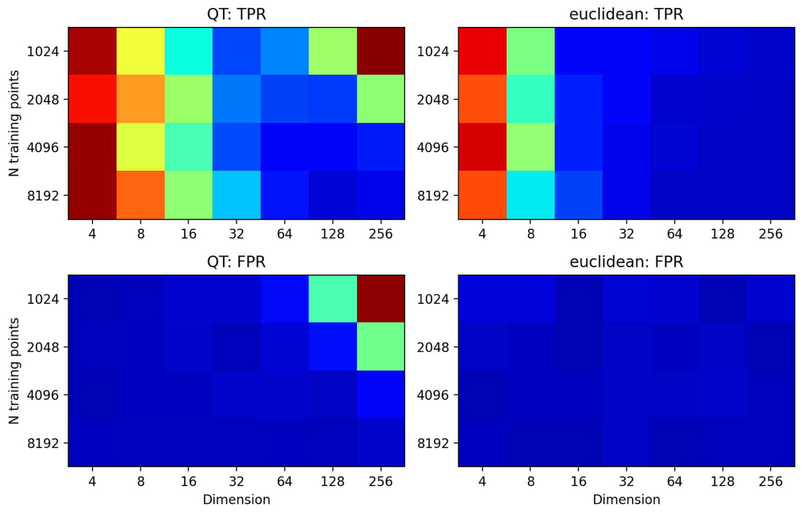


Figure 5.40: TPR and FPR achieved by QT and kQT with Euclidean and Mahalanobis distances on x-dimensional gaussian dataframes given $sKL = 1$. Data is preprocessed by PCA with no dimensionality reduction. We notice that FPR control holds when employing the Euclidean distance, independently from the number of training points and the data dimension, even after a PCA rotation (kQT is proven to be invariant under rototranslations). Instead, as we already noticed, PCA rotation only is enough to make QT's control of the FPR lost. Experiments were repeated 10 times, TPR and FPR are averaged each time on (256+256) test batches.

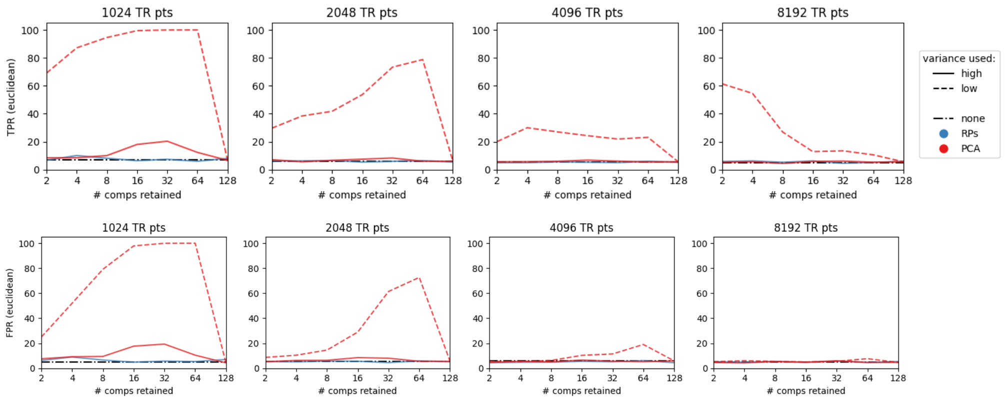


Figure 5.41: TPR (top) and FPR (bottom) measured with kernel-QuantTree with Euclidean distance employed, on data from two 128-dimensional monomodal gaussian distributions with $sKL(\phi_0 \to \phi_1) = 1$ fixed. Data is preprocessed with PCA or RPs. In the first case, we take as input both first and last x principal components to build the histogram and project the test batches. If the only PCA rotation ensures kQT's control over the FPR (invariance under rototranslations), a dimensionality reduction from d to $d' < d$ can ruin this property if an insufficient number of training points is given. In particular, FPR control is lost when we decide to keep low-variance components from high dimensional dataframes, if these are trained on too few training points. Experiments were repeated 10 times, TPR and FPR are averaged each time on (256+256) test batches.

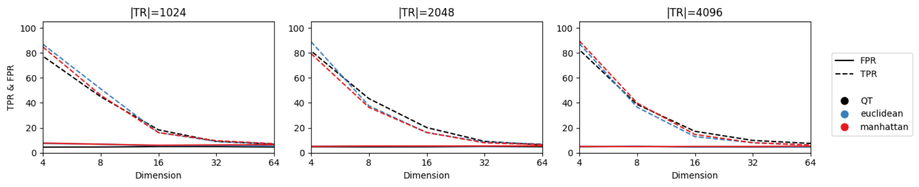


Figure 5.42: TPR and FPR achieved by QT and KQT with Euclidean (l_2 norm) and Manhattan (l_1 norm) distances on x-dimensional gaussian dataframes given $sKL = 1$. Data is not preprocessed. We notice that FPR control holds when employing the Manhattan distance, independently from the number of training points N and the data dimension d. Experiments were repeated 100 times, TPR and FPR are averaged each time on (256+256) test batches.

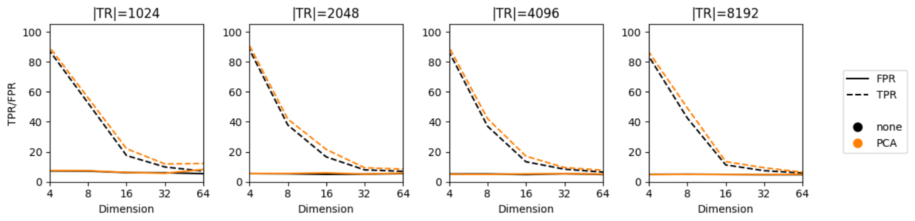


Figure 5.43: Manhattan-KQT performance with and without a PCA-like rotation of the data (no dimensionality reduction). Even if the invariance to rotations property drops when employing the l_1 norm, KQT performance seems to hold still. Experiments were repeated 100 times, TPR and FPR are averaged each time on (256+256) test batches.

properties.

5.17.1 Manhattan distance ($p = 1$)

Similarly to the Euclidean distance and the standard QT, the Manhattan distance exhibited excellent control over the False Positive Rate (FPR) independently from the number of training points and the data dimension, as we show in Fig.5.42. In terms of power (TPR), the performance seems to be similar to the one of the Euclidean KQT in these configurations of N and d. While the Manhattan distance, as an l_1 norm, is not generally invariant to rotations that deviate from a $\frac{\pi}{2}$ rotation, even in the presence of PCA-like transformations, kQT's performance with the Manhattan distance remains remarkably consistent if a sufficient number of training points is given.

The consistency of KQT's performance when using the Manhattan distance in the presence of PCA-like rotations without dimensionality reduction underscores its potential as a robust tool for concept drift detection in high-dimensional spaces. Our experiments (see Fig. 5.43) unveiled that when employing PCA

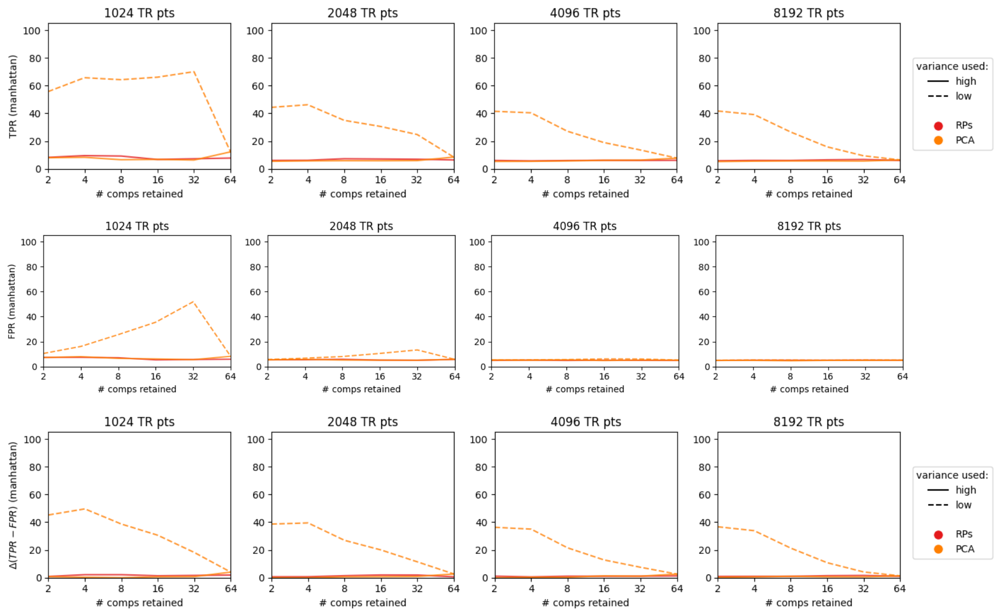


Figure 5.44: 128-dimensional dataset preprocessed with PCA and RPs and analyzed by Manhattan-kQT keeping only x components; for PCA we distinguish the two cases in which we keep high variance (i.e. first x principal components) and low variance components. FPR control that kQT exhibits when using the Manhattan distance may not hold if we choose to keep only a subset of the projections. This is particularly evident when low variance components (in dashed lines) are retained. This can be mitigated by increasing the number of training points (N); it is worth noting that this choice of low-var components can significantly boost the algorithm's detection power (as shown by TPR and $\Delta(TPR - FPR)$. Experiments were repeated 100 times, TPR and FPR are averaged each time on (256+256) test batches.

for dimensionality reduction, particularly in retaining only a subset of the principal components, some challenges arise. Control over the False Positive Rate (FPR) that KQT exhibits when using the Manhattan distance may not hold if we choose to keep only a subset of the projections. This is particularly evident when low variance components are retained, as they often represent noise or less significant information in the data; however, this effect can be mitigated by increasing the number of training points (N). The more training points available, the algorithm becomes more resilient to the noise introduced by low variance components, as we have seen previously with other methods. While maintaining strict FPR control is a critical requirement in concept drift detection, it's equally important to consider the trade-off between control and detection power: retaining low variance components, despite the potential challenges they pose to FPR control, can significantly boost the algorithm's detection

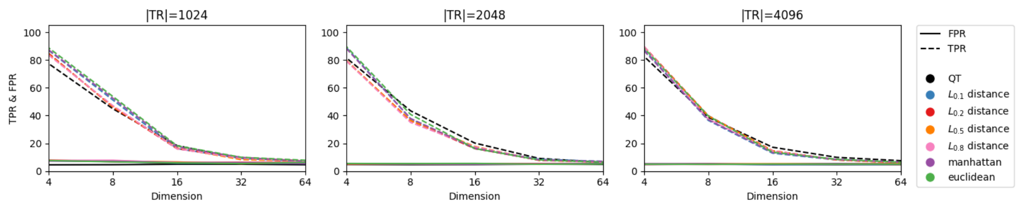


Figure 5.45: TPR and FPR achieved by QT and kQT with Euclidean (l_2 norm), Manhattan (l_1 norm) distance and the l_p fractional distances for $p \in 0.1, 0.2, 0.5, 0.8$ on x-dimensional gaussian dataframes given $sKL = 1$. Data is not preprocessed. FPR control holds when employing also the fractional distance, independently from these numbers of training points N data dimensions d. Experiments were repeated 100 times, TPR and FPR are averaged each time on (256+256) test batches.

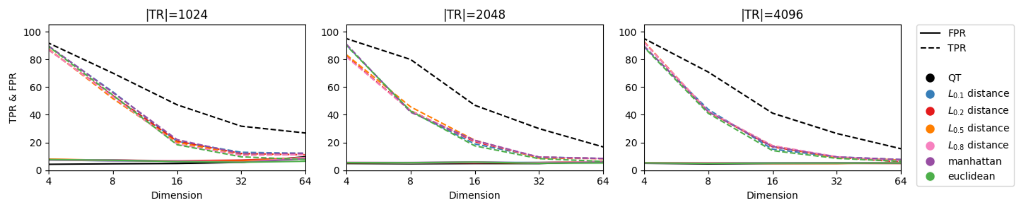


Figure 5.46: PCA preprocessing: l_p-KQT performances after PCA (projection on principal components with no dimensionality reduction). Even if the invariance to rotations property drops, KQT performance seems to hold still. However Experiments were repeated 100 times, TPR and FPR are averaged each time on (256+256) test batches.

power. The increase in the difference between TPR and FPR is shown in the last figure of Fig. 5.44.

5.17.2 Fractional distances ($p < 0.1$)

Similarly to the standard QT, kQT with the Euclidean distance and the Manhattan distance, also kQT based on a fractional distance (derived from the l_p norm with $p \in \{0.1, 0.2, 0.5, 0.8\}$) exhibited excellent control over the False Positive Rate (FPR) given a sufficient number of training points and the data dimension, as we show in Fig.5.45. In terms of detection power (TPR), the performance seems to be similar to the one of the Euclidean KQT in these configurations of N and d. While the fractional distances are not generally invariant to rotations, as we have seen with the Manhattan distance, even in the presence of rotations like the ones introduced by PCA-like transformations, kQT's performance remained remarkably consistent.

We can see from Fig. 5.47 and 5.48 that even if with a sufficient number

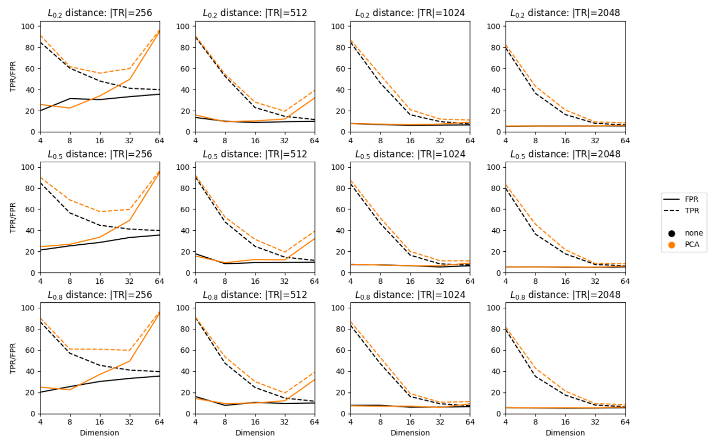


Figure 5.47: Effects of sample-based PCA with small training sets: x-dimensional dataset preprocessed with PCA and analyzed by KQT given the three metrics $l_{0.2}$, $l_{0.5}$ and $l_{0.8}$ keeping only x components. A poor training set is enough to lose FPR control even without any rotation, even in small dimensional spaces. KQT performances with l_p distances become invariant to PCA with a sufficient N. Experiments were repeated 30 times, TPR and FPR are averaged each time on (512+512) test batches.

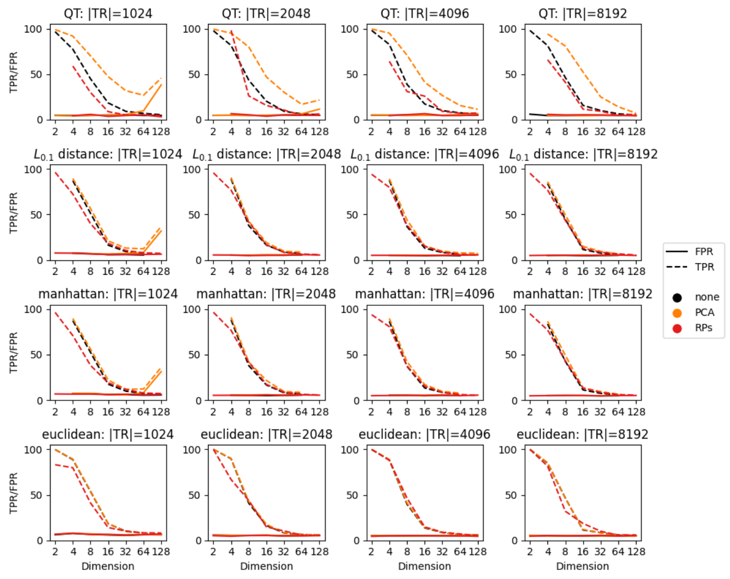


Figure 5.48: Overall effect of PCA and Random Projections with no dimensionality reduction over QT and KQT with l_p distances. We can see that only in the case of $p = 2$, i.e. Euclidean distance, the performances of the algorithm are independent of PCA-like rotations. Experiments were repeated 30 times, TPR and FPR are averaged each time on (512+512) test batches.

N of training points the FPR stays below the fixed threshold $\alpha = 5\%$, KQT performances with l_p norms are never really better than QT's, considering that a QT histogram is way faster to compute. The same can be shown when reducing dimensionality (Fig. 5.49). KQT clusters are more robust when we project data on the Principal Components (high variance components), maintaing the FPR at the threshold $\alpha = 5\%$ when the number of dimensions is substantially reduced. On the other hand, QT is more stable when keeping low variance components, which may be much important than usual in a Concept Drift Detection framework - in this subspace we expect the change to happen.

5.18 The Swarm Behavior Dataset

We perform our Concept Drift Detection algorithms by randomly selecting a training set of cardinality N from "flocking" datapoints and split the remaining $12000 - N$ instances in batches of size K. Post-change batches are made out of points drawn from the "non-flocking" distribution, as if at some point the animals had stopped moving in swarms and had begun to move independently, each on its own. We do not standardize the values of the $12 * 200 = 2400$ features.

5.18.1 Random Projections

As discussed in Chapter 4, the Johnson-Lindenstrauss (JL) lemma states that any high dimensional dataset can be randomly projected into a lower dimensional Euclidean space while controlling the distortion in the pairwise distances. The distortion ϵ introduced by a random projection is asserted by the fact that the projection is defining an eps-embedding with good probability as defined by Eq. 4.3. There is a minimum number of projection components to guarantee this eps-embedding, or: the increase of the admissible distortion ϵ allows to reduce drastically the minimal number of dimensions for a given number of samples.

We use the high-dimensional dataframe Swarm Behavior to validate the bounds given by the JL lemma: we want to see that for low values of the post-processing dimensionality (number of projections), the distribution is wide with many distorted pairs and a skewed distribution, due to the hard limit of zero ratios between distances since distances are always positive, while for larger subsets of components, the distortion is controlled and the distances are well preserved by the random projections. The dataframe is a multivariate set of more than 24000 intances in a 2400-dimensional space. We reduce dimensionality with Gaussian random projections, i.e. drawing the n components from $\mathcal{N}(0, 1/n)$, and plot the 1D histogram of the ratio of pairwise distances in original and projected spaces ($\frac{\text{projected}}{\text{original}}$). Results are shown in Fig. 5.50.

According to the JL lemma, projecting 300 samples without too much distortion will require at least several thousands dimensions, irrespectively of the number of features of the original dataset. Hence using random projections e.g. on a 128-

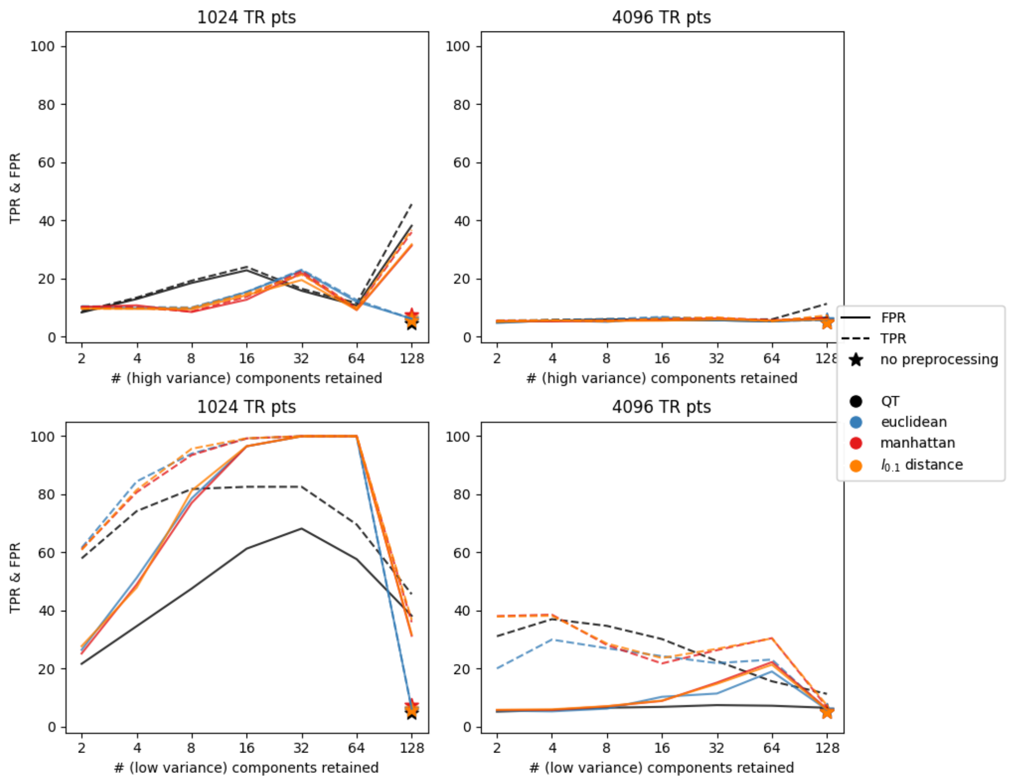


Figure 5.49: Effects of sample-based PCA: 128-dimensional dataset preprocessed with PCA and analyzed by KQT ($N \in \{1024, 4096\}$) given the three metrics $l_{0.1}$, l_1 and l_2 and keeping only x components, from left to right the first (high variance) and the last components respectively. KQT clusters are more robust when we project data on the Principal Components (high variance components), maintaing the FPR at the threshold $\alpha = 5\%$ when the number of dimensions is substantially reduced. On the other hand, QT is more stable when keeping low variance components, which may be much important than usual in a Concept Drift Detection framework - in this subspace we expect the change to happen. Experiments were repeated 30 times, TPR and FPR are averaged each time on (512+512) test batches.

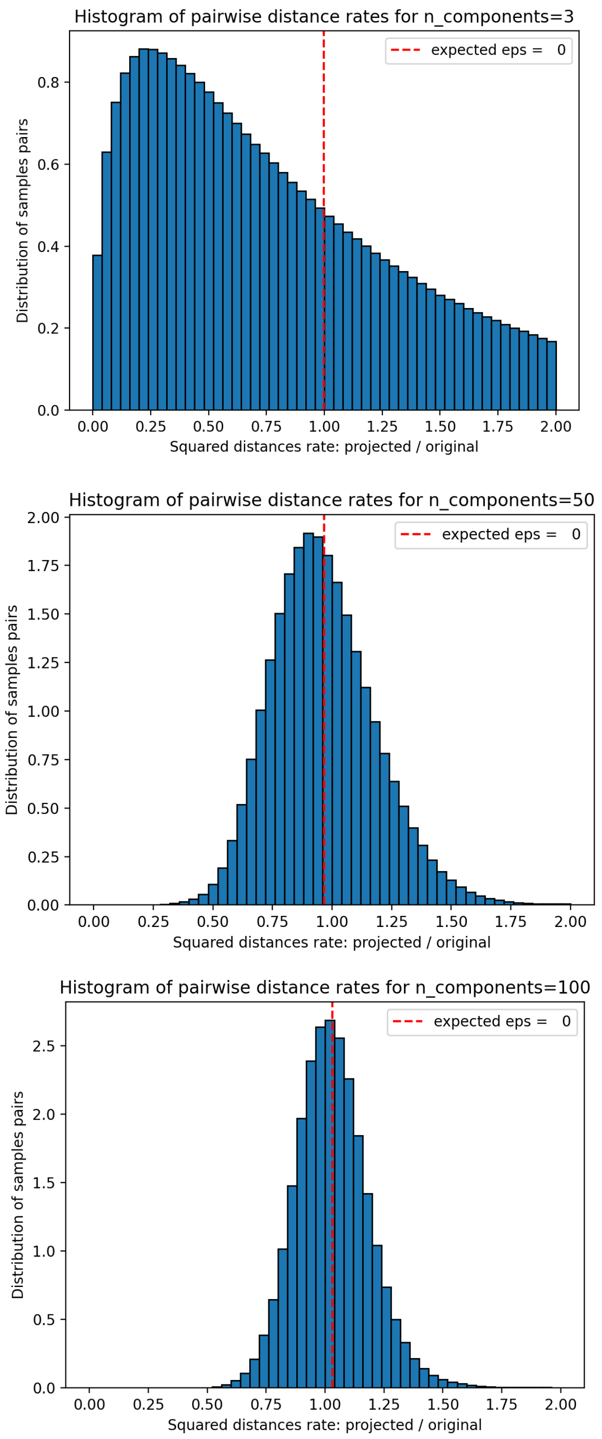


Figure 5.50: Projection of 300 samples from the Swarm Behavior dataset ($d = 2400$) using Gaussian RPs. We can see that for low values of n_components the distribution is wide with many distorted pairs and a skewed distribution, while for larger values of n_components the distortion is controlled and the distances are well preserved by the RP.

dimensional dataframe, does not make much sense. With this huge dimensional dataframe, we hope that RPs will able our detection methods to learn something from the relatively few datapoints provided.

5.18.2 PCA preprocessing

We reduce the dimensionality with PCA to test the performances of QT and KQT adopting various distances. Evaluations are made on both the high variance components and the low variance ones, keeping subspaces of increasing dimension $d' \in \{2, 4, ..., 512, 1024\}$. In Figure 5.51 we show the results we obtain from small training sets of $N = 512$ points used both to find the projected space and to build the histograms. Keeping high variance components, KQT with Weighted Mahalanobis distance is the first to lose FPR control when the projected space dimensionality scales, followed by KQT with Mahalanobis distance, QT, and KQT with l_p distances with $p \in \{0.1, 1\}$. Euclidean-KQT is able to maintain near the α threshold for any number of projection components. On the other hand, keeping low variance components is always bringing FPR value to 100% (each new sample raises an alarm).; the only exception given by KQT if we keep at least $d' = d/2$ dimensions for the projected space. The average TPR computed always equals 100%, i.e. the models are always signaling an alarm when receiving a post-change batch: each batch drawn from the non-flocking dataframe is recognized as one. In these conditions, it seems that QT after reducing dimensionality to $d' \in \{2, 4\}$ can maintain the FPR below the 5% threshold and achieve a 100% TPR.

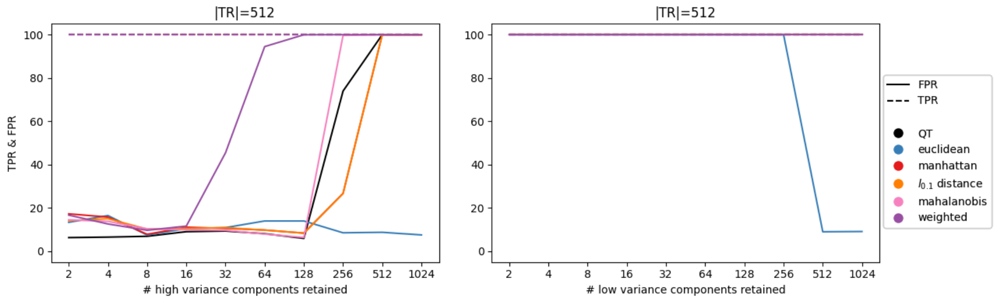


Figure 5.51: Dimensionality reduction by PCA: 1200-dimensional dataset Swarm Behavior preprocessed with PCA and analyzed by QT and KQT ($N = 512$) keeping only x components, from left to right the first (high variance) and the last components respectively. With a threshold $\alpha = 5\%$, all the models are able to detect the change of the distribution (from "flocking" distribution ϕ_0 and non-flocking) Experiments were repeated 30 times over the whole dataframe randomly sampled. KQT with Weighted Mahalanobis distance is the first to lose FPR control when the projected space dimensionality scales, followed by KQT with Mahalanobis distance , QT, and KQT with l_p distances with $p \in \{0.1, 1\}$. Euclidean KQT is able to maintain near the α threshold for any number of high variance components. FPR control is completely lost when projecting data on low variance components estimated from the training set.

Chapter 6

Online experiments

In this chapter we present the experimental evaluation of the proposed solution KQT-EWMA. Our aim is to show that the algorithm controls the false alarms comparably or better than competing methods, while achieving lower detection delays. In QT-EWMA and KQT-EWMA we set $K = 32$ - unless otherwise specified - and uniform target probabilities $\pi_j = 1/K$, since uniform histograms have been shown to be very effective for change detection purposes [7]. For SPLL we set $\nu = 32$.

6.1 Datasets

6.1.1 Synthetically generated data

As we discussed in the Batch-wise experiments chapter, to enable a fair comparison among change-detection algorithms, we rely on the "Controlling Change Magnitude" framework [2] to generate post-change distributions given the symmetric Kullback-Leibler divergence with respect to the stationary distribution. We generate synthetic datastreams in different dimensions d by choosing an initial distribution ϕ_0 with random covariance matrix, and as alternative post-change distribution a random roto-translation of ϕ_0 computed as $\phi_1 = \phi_0(Q, v)$, with parameters Q and v computed using the CCM framework.

Monomodal Gaussian

We consider a stationary distribution ϕ_0 which is a null-mean Gaussian with a random covariance matrix. The post-change distribution ϕ_1 is obtained by roto-translation using the CCM framework, such that the symmetric Kullback-Leibler distance (sKL) between ϕ_0 and ϕ_1 is fixed. If not specified, in our experiments the target sKL is set to 1. N points are sampled from ϕ_0 to generate the training set; we build sequence of a fixed length ($l_{seq} = 10000$ if not specified otherwise) to be monitored. The sequences can be characterized by a change at some point cp, or not. In the first case, we concatenate a sequence drawn from ϕ_0 which has length τ with a sequence of length $l_{seq} - \tau$ of samples drawn from the post-change distribution ϕ_1; to test the average running length ARL_0 or the false alarm rate, we build sequences with samples drawn from the stationary distribution ϕ_0. When a target ARL_0 is given, the sequence is $L = 6{*}ARL_0$ long. We perform our experiments in different configurations given by the changing cardinality of the training set $|TR| = N$ and the dimensionality d of the dataframe the sequences are sampled from. The symmetric divergence is fixed to $sKL = 1$ unless otherwise specified. A visual insight we can refer to, together with a discussion on the *curse of dimensionality*, is given in the previous chapter (see Figures 5.1, 5.2)

6.1.2 Real-world dataframes

In *Challenges in Benchmarking Stream Learning Algorithms with Real-world Data* [24], the authors review the difficulties related to the *comparison and evaluation of streaming algorithms due to the lack of publicly available non-stationary real-world datasets.* To test QT-EWMA, in [10], the authors employ real-world datasets from the UCI Machine Learning Repository [14] with dimensions ranging from $d = 5$ to $d = 50$. We also test our change-detection method on these traditional multivariate classification datasets: Credit Card Fraud Detection ("credit", $d = 28$), Sensorless Drive Diagnosis ("sensorless", $d = 48$), MiniBooNE particle identi- fication ("particle", $d = 50$), Physicochemical Properties of Protein Ternary Structure ("protein", $d = 9$), El Niño Southern Oscillation ("niño", $d = 5$), and two of the Forest Covertype datasets ("spruce" and "lodgepole", $d = 10$). As in [10], we standardize the datasets and sum to each component of "sensorless", "particle", "spruce" and "lodgepole" imperceptible Gaussian noise to avoid repeated values, which harm the construction of QuantTree histograms.

The problem with these dataframes is that they are not meant to be used to test concept drift detection algorithms. The distributions are typically considered stationary, so we can randomly sample the datastreams, but we introduce changes thanks to the usual CCM framework [2], differently from [10], where changes are obtained shifting the post-change samples by a random vector drawn from a d-dimensional Gaussian scaled by the total variance of the dataset. These artificially introduced changes should label these experiments as something in the middle between a synthetic testbed and one from the real world... let's be cautious. However, each statistic derived from the pre-change distribution only, and thus the whole histograms, will be genuinely independent from these changes. That is why in [24] it is introduced a benchmark dataset based on the use of optical sensors to recognize flying insect species in real-time, which was also used in [10]. They cannot assume a stationary stochastic process because of the behavior of the insects: for example, temperature influences their metabolism, air pressure and humidity can change their flying behavior. For these reasons, the measures suffer from concept drifts over time. It works like this: the sensor has two parallel mirrors face-to-face, an infrared LED uses the mirrors to create a light window that ends in a phototransistor; when a flying insect crosses the light, its wings and body partially occlude it, causing small variations that are captured by the phototransistor. To classify insect species, the wing-beat frequency is one of the most relevant information that can be extracted from the signals. Beyond the fundamental frequency, the spectrum of a signal also has harmonic components which position also constitute important information. For three months, the authors of [24] varied temperature and humidity in these traps and recovered around one million instances for 17 different species including mosquitoes, houseflies, bees, and wasps, to build a 33-dimensional dataset. They wrote that their*signal, although optical, is very similar to audio* [as they consider it in the frequency space, *ndr*] *and consequently high-dimensional.* It is worth noting that these 33 dimensions are all derived from signal processing techniques to extract additional discriminative features for data obtained from a sensor, i.e. 1D signals, to be transformed in a feature vector. The authors extracted the wing-beat frequency, complexity measures of the signal spectrum, statistics from temporal representation, among others; I believe it is a risky choice to call it a multi-dimensional frame, especially in the context of machine learning, which tries to come over our human-defined parameters often burying some of the important aspects.

Considering this ubiquitous artificiality we used these dataframes way less than synthetics. In fact, we needed data points sampled from an enormous meshwork of different configurations, specifically considering values for d and N, and a lot of instances to ran thousands of experiments which results are often characterized by high variance.

6.2 Figures of Merit

In the context of batch-wise concept drift detection, which classify entire data batches as originating from either the stationary distribution or a new post-change distribution, we focus on metrics like True Positive Rate (TPR) and False Positive Rate (FPR). As we already discussed, TPR quantifies the ability to correctly classify batches from the post-change distribution, while FPR measures the rate of misclassifying stationary batches as drifted. On the other side, online methods typically employ metrics such as Average Run Length (ARL) and Detection Delay, prioritizing timely detection. We better define these quantities before beginning with the description of the experimental framework.

Empirical ARL_0

To assess whether kQT-EWMA and the other considered methods maintain the target ARL_0, we compute the empitical ARL_0 as the average time before raising a false alarm. To this purpose, we run the considered methods on n datastreams drawn from ϕ_0, setting the target $ARL_0 \in \{500, 1000, 2000, 5000\}$ as in [10]. We consider datastreams of length $L = 6 * ARL_0$ to have a detection in each datastream. Since, by construction, the detection time t^* of our method under ϕ_0 is a geometric random variable with parameter $\alpha = 1/ARL_0$, the

Equation 2.9 indicates that the probability of having a false alarm before L is $\mathbb{P}(t^* \leq L) \approx 0.9975$.

Detection delay

We evaluate the detection performance of kQT-EWMA and the other considered methods by their detection delay, i.e. $ARL_1 = \mathbb{E}[t^* - \tau]$, where the expectation is taken assuming that a change point τ is present. We run the methods configured with target $ARL_0 \in \{500, 1000, 2000, 5000\}$ on n datastreams of length $l_{seq} = 10000$, each containing a changepoint at $\tau = 300$. We estimate the ARL_1 as the average difference $t^* - \tau$, excluding false alarms.

False alarm rate

To assess whether the considered methods maintain the target false alarm probability, we compute the percentage of false alarms obtained on the datastreams used to evaluate the detection delay, i.e., those in which a detection occurs at some $t^* < \tau$. Also in this case, we set the target $ARL_0 \in \{500, 1000, 2000, 5000\}$, which, according to Eq. 2.9, yield a false alarm in 45%, 26%, 14% and 6% of the datastreams, respectively.

6.3 KQT-EWMA

Our first "demo" experiment to test the performance of kernel-QT-EWMA was performed on datastreams drawn from a relatively small dimensional space $\mathbb{R}^d$ with $d \in \{2, 4, 8, 16, 32\}$. QT and KQT histograms were built on training sets TR of cardinality $N \in \{1024, 4096\}$, containing sequences of samples from the stationary distribution ϕ_0. The post-change samples were drawn from a roto-translation of ϕ_0, ϕ_1 built such that $sKL(\phi_0 \to \phi_1) = 1$. The parameter λ for the computation of the EWMA statistic (see Eq. 2.5) was set to $\lambda = 0.03$. Our purpose was to show that *i*) the empirical ARL_0 of KQT-EWMA approaches the target and *ii*) that changes can be detected with smaller delays with respect to QT-EWMA. We report the outcomes of 1000 experiments in Fig. 6.1. The empirical ARL_0 of KQT-EWMA approach the target in small dimensional frames ($d \leq 8$) or with a sufficient number of training points. While in any case QT-EWMA and Euclidean KQT-EWMA can control the ARL_0 and SPLL cannot, KQT-EWMA with Mahalanobis and Weighted Mahalanobis lose this property with increasing dimensionality, when the number of training points N is fixed. We can compare these results with what was shown in the offline framework in Fig. 5.4, where we can see that FPR control is lost by KQT when built on Weighted Mahalanobis (first) and Mahalanobis (later) distances when dimensionality is increasing. We can compare FPR with ARL_0 control. Also in this case, an increased number of training points N can alleviate this phenomenon. As QuantTree and Euclidean KQT (see Fig. 6.1) could control FPR no matter the data dimension, QT-EWMA and Euclidean KQT-EWMA seems to approach the target ARL_0, no matter the data dimensionality or the

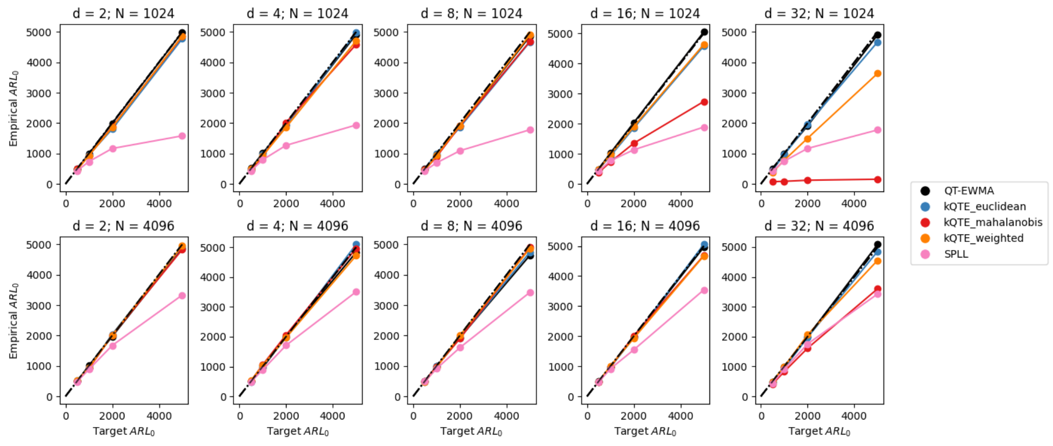


Figure 6.1: Experimental results over Gaussian datastreams with $d \in \{2, 4, 8, 16, 32\}$ and $N \in \{1024, 4096\}$ show that the empirical ARL_0 of KQT-EWMA approach the target in small dimensional frames ($d \leq 8$) or with a sufficient number of training points. While in any case QT-EWMA and Euclidean KQT-EWMA can control the ARL_0 and SPLL cannot, kQT-EWMA with Mahalanobis and Weighted Mahalanobis lose this property with increasing dimensionality, when the number of training points N is fixed. Values were averaged over 1000 experiments.

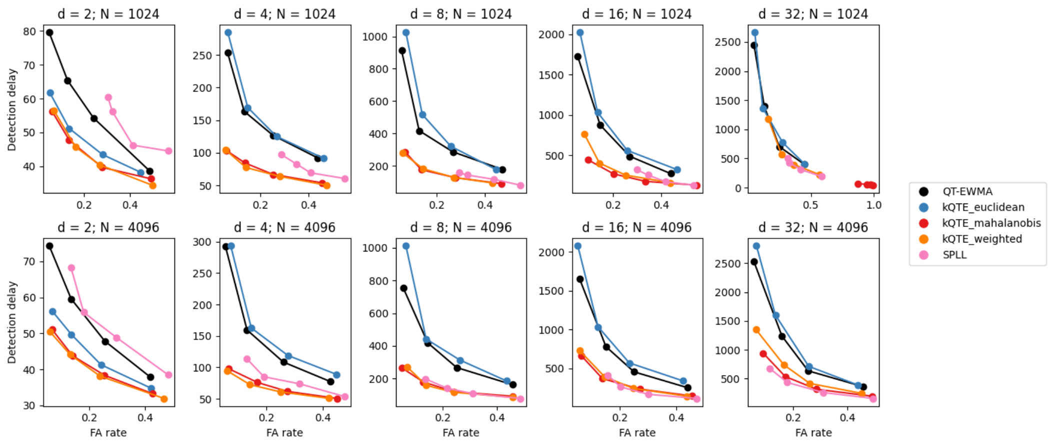


Figure 6.2: Experimental results over Gaussian datastreams with $d \in \{2, 4, 8, 16, 32\}$ and $N \in \{1024, 4096\}$ show that KQT-EWMA achieves the best (smallest) detection delays given the same false alarm rates. While QT-EWMA and Euclidean KQT-EWMA can control the false alarm rates, this is not true with Mahalanobis and Weighted Mahalanobis distances employed. Values were averaged over 1000 experiments.

number of training points provided. In Fig. 6.2 are compared the detection delays and the false alarm (FA) rates. KQT-EWMA achieves the best (smallest) detection delays given the same false alarm rates. While QT-EWMA and Euclidean KQT-EWMA can control the false alarm rates, this is not true with Mahalanobis and Weighted Mahalanobis distances employed. Also in this case, an increased number of training points N can alleviate this phenomenon. As QuantTree and Euclidean KQT (see Fig. 5.38) could control FPR no matter the data dimension, QT-EWMA and Euclidean KQT-EWMA seems to approach the target ARL_0 and maintain a low FA rate, no matter the data dimensionality or the number of training points provided. When there is control over the false positives, i.e. with a sufficient number of training points, Mahalanobis and Weighted Mahalanobis KQT-EWMA achieves the lowest detection delays.

6.4 Euclidean KQT-EWMA in High-Dimensional dataframes

While in any case QT-EWMA and Euclidean KQT-EWMA can control the ARL_0 and SPLL cannot, KQT-EWMA with Mahalanobis and Weighted Mahalanobis lose this property with increasing dimensionality, N fixed (same discussion as for the FPR control batch-wise). When there is control over the false positives, i.e. with a sufficient number of training points, Mahalanobis and Weighted Mahalanobis KQT-EWMA achieves the lowest (the best) detection delays and

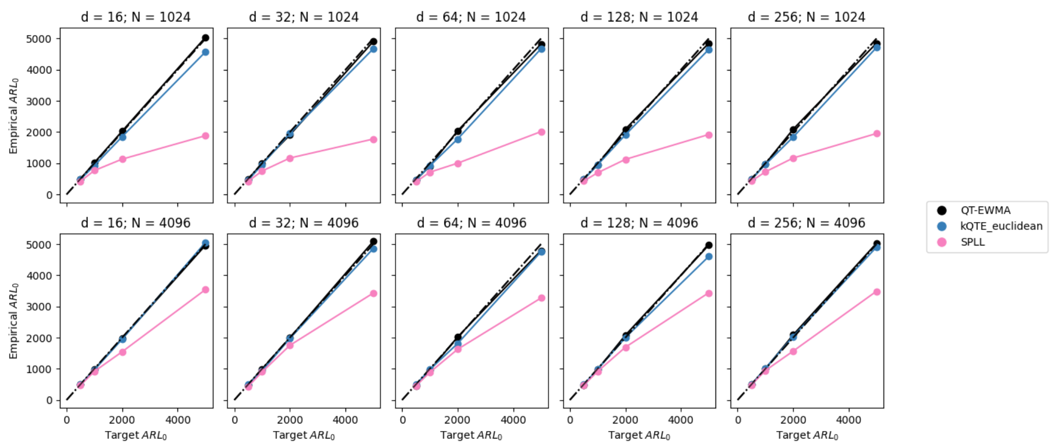


Figure 6.3: Experimental results over streams sampled from Gaussian distributions with $d \in \{2, 4, 8, 16, 32, 64, 128, 256\}$, $N \in \{1024, 4096\}$, and $sKL = 1$, show that QT-EWMA and Euclidean KQT-EWMA are able to control RL_0 even at small N/d ratios, when SPLL is not. Target ARL_0 values where chosen between 500, 1000, 2000, and 5000. Data was not preprocessed. Values are averaged over 1000 experiments.

outshines both QT-EWMA and the "oracle" SPLL, both in terms of controlling ARL_0 and in achieving impressively low detection delays when N/d ratio is in favor. As we did offline, for concept drift detection higher-dimensional spaces we go on with QT and Euclidean KQT, these being able to control the false alarms independently from the dimension.

In Fig. 6.3 experimental results conducted on these Gaussian datastreams with no preprocessing show the effectiveness of QT-EWMA and Euclidean KQT-EWMA in controlling ARL_0 at small N/d ratios. Target values were chosen between 500, 1000, 2000, and 5000. We remark that SPLL, which assumes a Gaussian distribution, enjoys an inherent advantage in fitting the data; despite this, both QT-EWMA and Euclidean KQT-EWMA, employing non-parametric histogram-based approaches, demonstrate comparable or superior performances in maintaining control over ARL_0, especially under challenging conditions. Figure 6.4 emphasize the robust performance of QT-EWMA and Euclidean KQT-EWMA in achieving the chosen False Alarm (FA) rate, particularly at small N/d ratios, when SPLL faces challenges and QT-EWMA and Euclidean KQT-EWMA exhibit the ability to achieve lower detection delays. Another representation of our results is in Fig. 6.5. As dimensionality scales, QT-EWMA and KQT-EWMA delays approach the target ARL_0 as TPR approaches FPR batch-wise; we referred to this phenomenon as *detectability loss*. SPLL's detection delay has the same trend, but empirical ARL_0 drops away from the target when N/d decreases.

In situations where the sample size (N) is small (e.g., 1024) and dimensionality

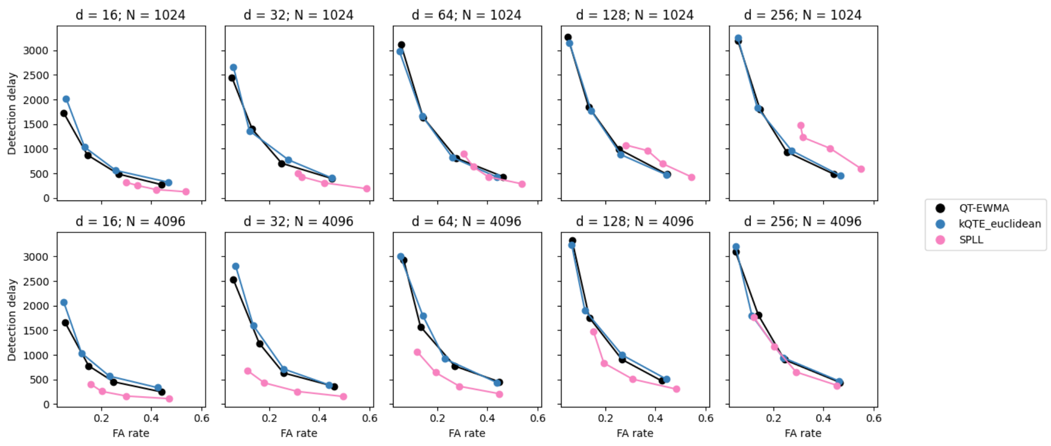


Figure 6.4: Experimental results over datastreams sampled from Gaussians with $d \in \{2, 4, 8, 16, 32, 64, 128, 256\}$, $N \in \{1024, 4096\}$, and $sKL = 1$, show that QT-EWMA and Euclidean KQT-EWMA are able to achieve the chosen False Alarm (FA) rate even at small N/d ratios, when SPLL is not. In these inconvenient frames, QT-EWMA and KQT-EWMA achieve lower detection delays with respect to the "oracle" SPLL given the same FA rates. Target ARL_0 values where chosen between 500, 1000, 2000, and 5000. Values are averaged over 1000 experiments.

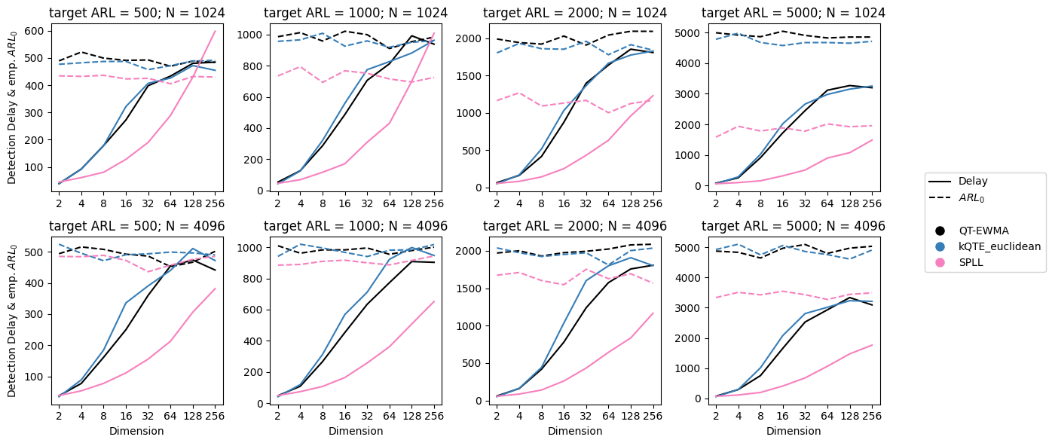


Figure 6.5: Experimental detection delay and FA rate over datastreams sampled from Gaussians distribution with $d \in \{2, 4, 8, 16, 32, 64, 128, 256\}$, $N \in \{1024, 4096\}$, and $sKL = 1$; QT-EWMA and Euclidean KQT-EWMA are compared with SPLL given different target ARL_0 values in {500, 1000, 2000, 5000}. Values are averaged over 1000 experiments.

scales, QT-EWMA and Euclidean KQT-EWMA exhibit the lowest (best) detection delays, with comparable performance (but with QT being way faster to compute from TR). In more favorable environments with larger training sets, SPLL's detection delays decrease under GQT's; however, the model remains impractical due to its inability to manage false positives. We remark one last time the significance of the False Alarm rate as a critical metric in evaluating a model's performance, transcending the mere reduction of detection delays, as higher FA rates indicate a constant stream of alarms, not only when genuinely needed, undermining the model's reliability.

6.5 UCI datasets and PCA

We have seen the effects of PCA on the outcomes of our algorithms over data sampled from Gaussian distributions. We have seen what happens when we keep a small number of high or small variance components to build the projected space. We can now guess that when performing PCA with a sufficient number of training points and then analyze each component *separately* we obtain something like Fig. 6.6: given control over ARL_0, the detection delay is lower for low variance components, which are more probable to be affected by the change significatively.

This section marks the conclusion of the thesis but also starts from my initial exploration of these algorithms. The works from G.Boracchi et al.([6], [10], [25]) reported that PCA preprocessing is generally beneficial for QT, but only in certain settings. We wanted to systematically explore PCA effects over these dataset, which were referred by that *generally, but only in certain settings.* We usually do not rely on one single component since we don't want to lose much information while the FPR control holds, still we performed this kind of experiment for the UCI dataframes to see if this success in using low variance components is suitable only with Gaussian distributions or if it holds on these real-world datastreams, too.

In Figures 6.6 and 6.7, we observe that the last components, in terms of explained variance, often correspond to lower detection delays. It is always important that the number of training points is sufficient to maintain a the target ARL_0. In these experiments, the mean FA rate remains constant and fixed, and no significant variations or orderings are measured among different components. This is not true for "particle" ($d = 50$) and "insects" ($d = 33$) dataframes. In both cases we can't distinguish any trend with respect to the variance-based ranking of the components used to project datasets. We can see the results of this experiment over the Insects dataframe in Fig. 6.8, where the "last" component is the one achieving the worst detection delay, while on the direction associated with the lowest detection delay we also lose ARL_0 control, and the FA rate doubles the average.

In Figure 6.9 we show the results obtained building our models over subsets of first (or last) x principal components, as we always did. We show for example the"credit" dataframe. There is an obvious trend in the detection delay which

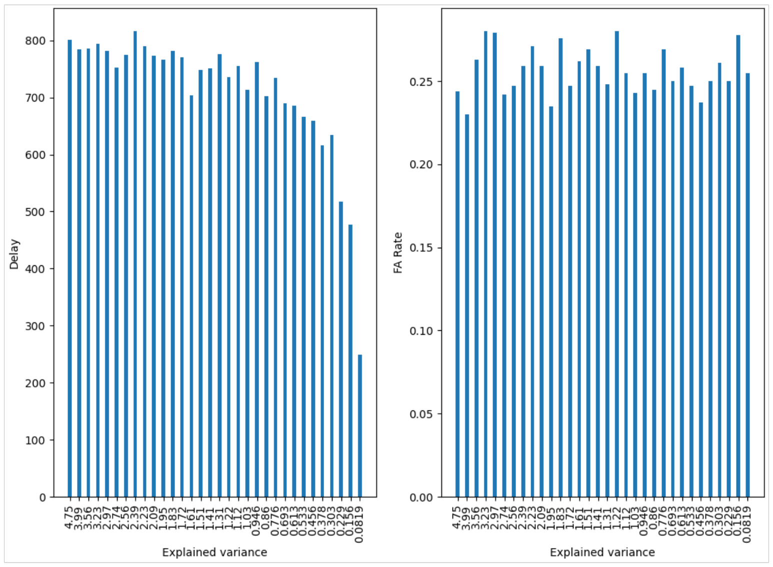
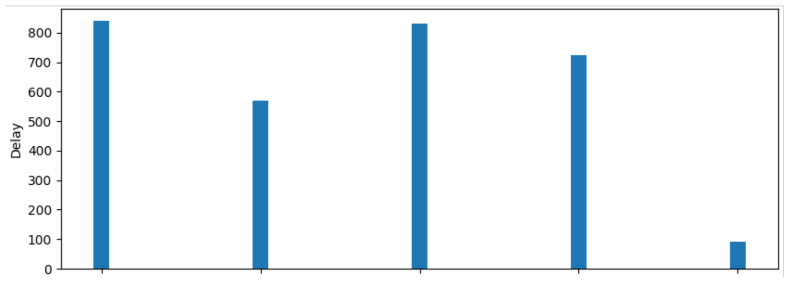
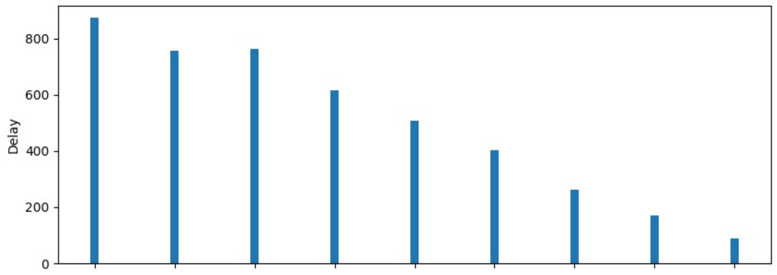


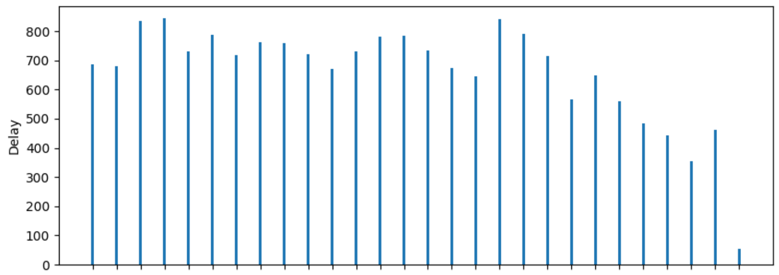
Figure 6.6: Experimental detection delay and FA rate over datastreams sampled from Gaussians distribution with $d = 32$, $N = 1024$, $sKL = 1$. Datastreams are preprocessed with PCA, then projected on one single component and given to QT-EWMA as 1D signals, one component each time till all are used to be compared. We saved the mean Explained variance of the n-th component over 1000 experiments and display it on the x axis ordered by its rank.



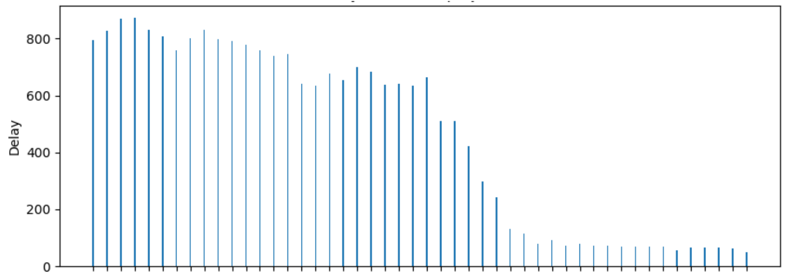
(a) “niño” dataframe ($d = 5$)



(b) “protein” dataframe ($d = 9$)



(c) “credit” dataframe ($d = 28$)



(d) “sensorless” dataframe ($d = 48$)

Figure 6.7: Experimental results over datastreams sampled from UCI datasets with increasing d and $N = 1024$. Datastreams are preprocessed with PCA, then projected on one single component and given to QT-EWMA as 1D signals, one component each time till all are used to be compared. We saved the mean Explained variance of the n-th component over 1000 experiments and display it on the x axis ordered by its rank.

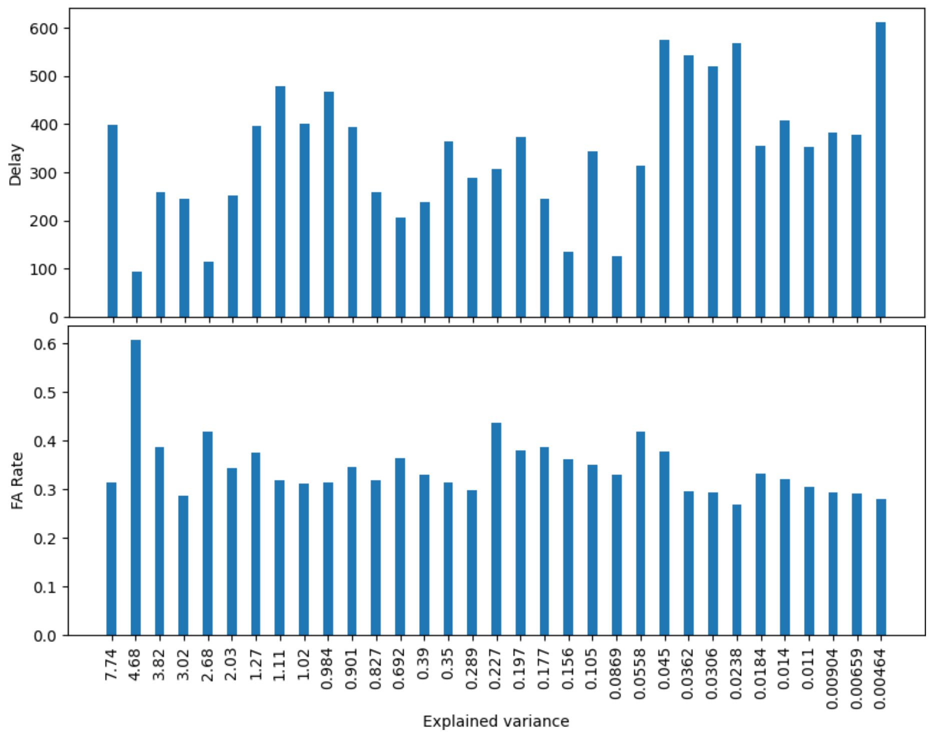
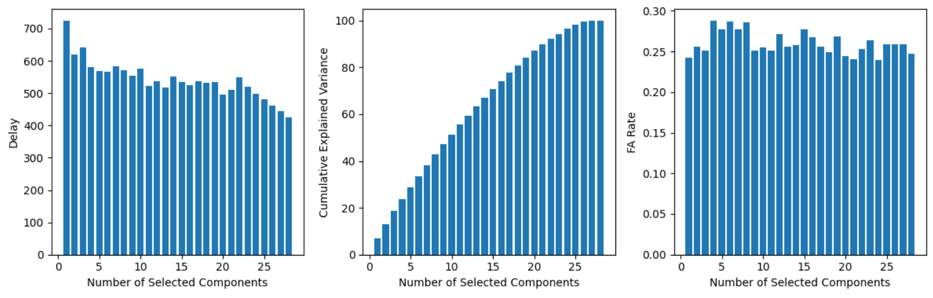
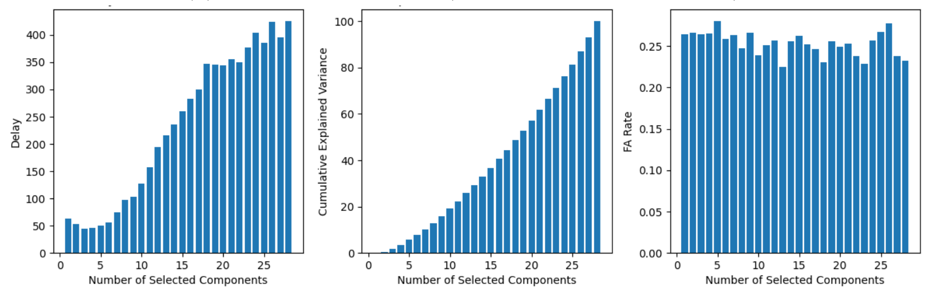


Figure 6.8: Experimental results over datastreams from UCI Insect dataframe ($d = 33$) with $N = 1024$. Datastreams are preprocessed with PCA, then projected on one single component and given to QT-EWMA as 1D signals, one component each time till all are used to be compared. We saved the mean Explained variance of the n-th component over 1000 experiments and display it on the x axis ordered by its rank. We can see no trend in the detection delay $t^* - \tau$ as the explained variance goes to 0; the second principal component achieves the lowest $t^* - \tau$ but is also associated with a False Alarm rate which doubles the average. The direction explaining the lowest variance is the one giving the worst delay.



(a) Keeping first x PCs (high variance components)



(b) Keeping last x PCs (low variance components)

Figure 6.9: "credit" dataframe ($d = 28$), $N = 1024$. Datastreams are preprocessed with PCA, then projected on groups of x components chosen in order and given to QT-EWMA as x-dimensional multivariate streams, till all components are used. We saved the mean Cumulative Explained variance of the components over 1000 experiments and display it between the detection delay (on the left column) and the FA rate (right column). We can see no trend in the FA rate, but the lowest detection delay is achieved when keeping the last 3-4 PCs to project the test points.

reaches its minimum when keeping the last 3-4 principal components to project the test points.

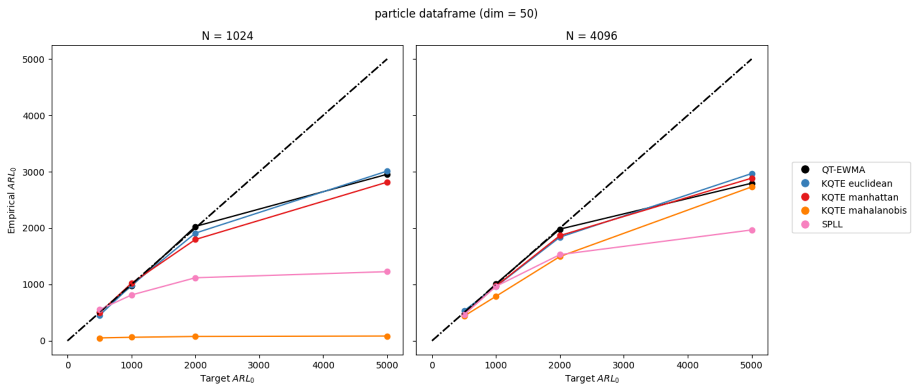
KQT-EWMA

The problem with these real world datasets, with respect to our usual synthetic frames generated from Gaussian distributions, might be the control over false positives. We show for example the results obtained over "particle" ($d = 50$, our biggest UCI datasets within the ones used in [10]), "credit" ($d = 28$), and "protein" ($d = 9$) dataframes. We set the cardinality of the training sets to $N \in \{1024, 4096\}$, and describe the results obtained with our new implementation KQT-EWMA adopting the three Manhattan, Euclidean (l_p with $p = 1$, $p = 2$ respectively) and Mahalanobis distances, including a comparison with QT-EWMA and SPLL.

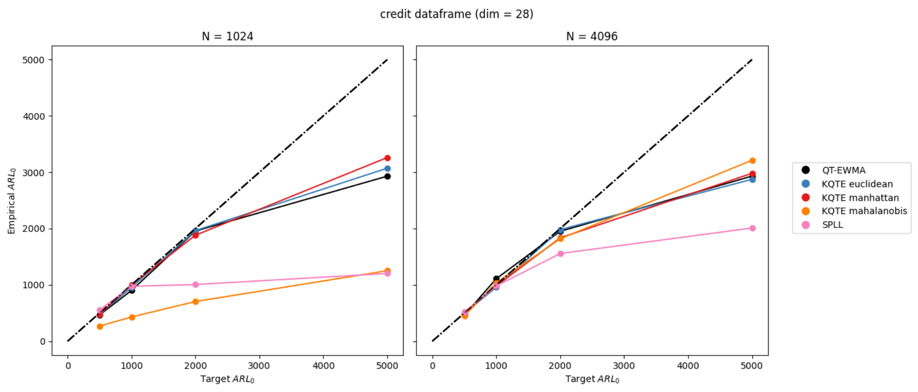
We confront control over ARL_0, given the fixed target, in Fig. 6.10. All these models struggle to maintain empirical ARL_0 values near the target when this target increases. This is true also for the "protein" - our smaller - dataset, given $N = 4096$ training points. SPLL - which, we remark, is based on a Gaussianity hypothesis which is not satisfied here, is never able to control the false positives - same as it ever was; KQT-EWMA with Mahalanobis distances gives the worst results when the N/d ratio is small (e.g. on "particle" and "credit" dataframes with $N = 1024$). KQT-EWMA with Euclidean and Manhattan distances generally achieves the same control over ARL_0 as QT-EWMA did, which is the best achieved overall. KQT-EWMA with Mahalanobis distance obtains the best advantages from a bigger training set in terms of decreasing the FA rate.

QT-EWMA and each of its variants always outperforms SPLL over all these dataframes(see Fig. 6.11). When ARL_0 - thus, FA rate - is controlled, Mahalanobis KQT-EWMA is the best model in terms of detection delay obtained, achieving considerably lower detection delays with respect to SPLL, QT-EWMA and KQT-EWMA with l_p norms (Manhattan distance corresponding to $p = 1$ and Euclidean to $p = 2$). Mahalanobis distance, considering global relationships between datapoints - it is indeed based on the inverse sample covariance matrix computed over training points - scores a half of the detection delay of QT-EWMA on the "particle" dataframe when $N = 4096$, which is was already way lower than SPLL's. That is, once more, for what we have tried and we know, KQT-EWMA can outperform the state of the art, but lacks of robustness.

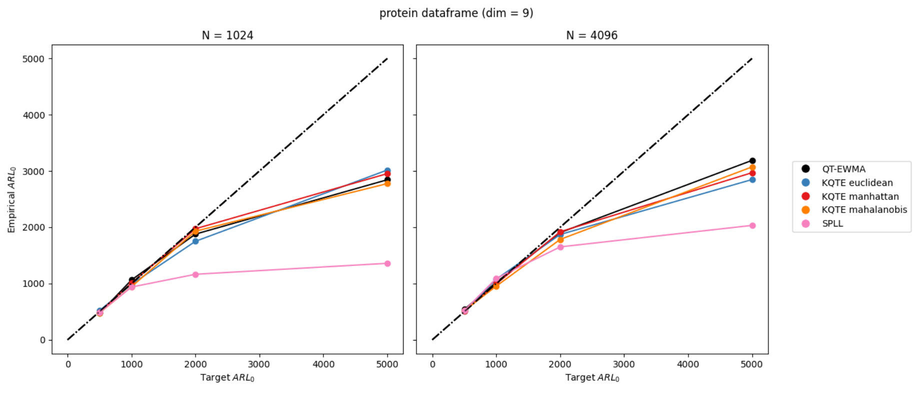
We reduce dimensionality with PCA using as examples the "particle" and "credit" dataframes. Results (Fig. 6.12 shows once again that low variance components are more likely to be useful to detect drifts. Indeed, to achieve lower detection delays keeping PCs computed by sample-based PCA, it is convenient to hold a greater number of components, as it is evident in the results obtained from the "particle" dataframe. Instead, using our models to analyze low variance components gives way lower detection delays for each of our models. Also, in the frames we tested and show, low variance components are always robust enough to achieve the same FA rate that PCs achieve, with the only exception of SPLL detection performance over the "particle" dataframe.



(a)

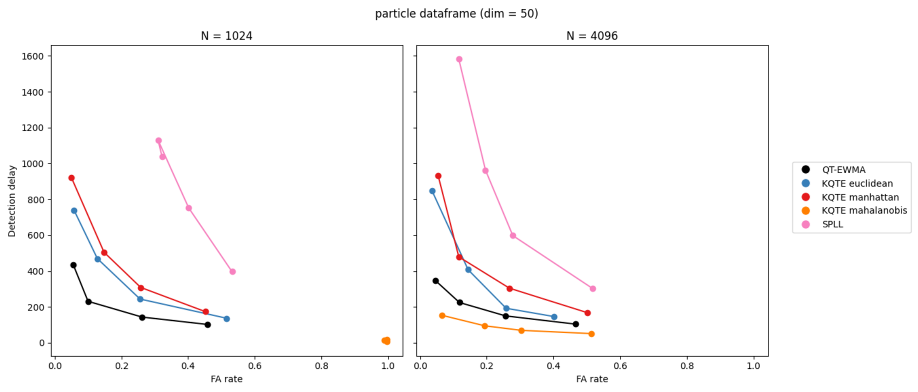


(b)

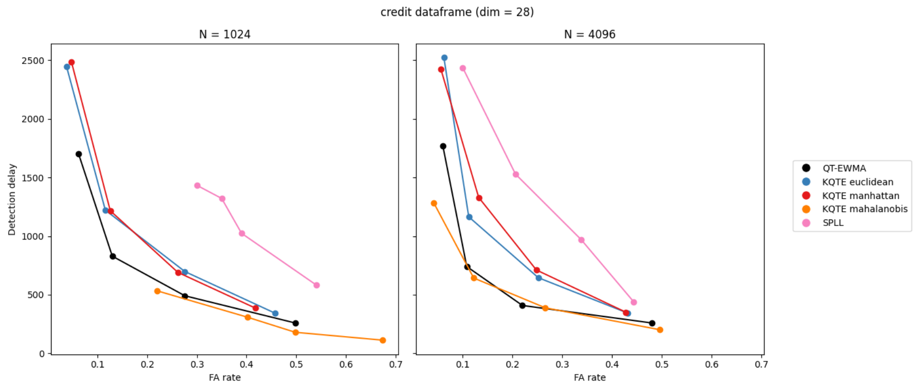


(c)

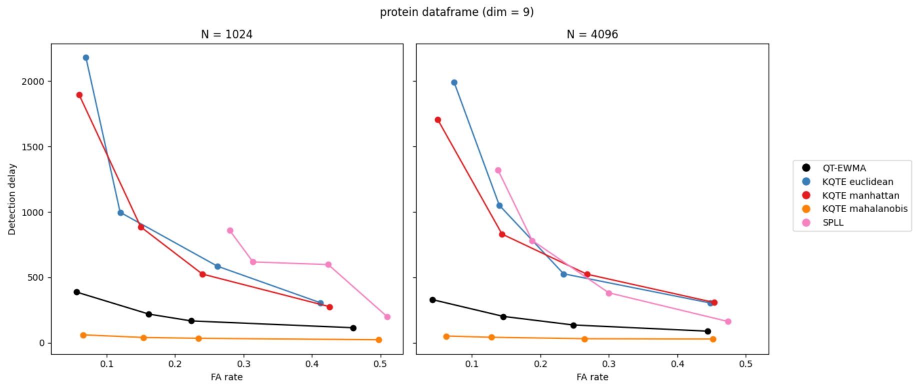
Figure 6.10: False Positives control over UCI (a:"particle" ($d = 50$), b:"credit" ($d = 28$), and c:"protein" ($d = 10$)) dataframes between SPLL, QT-EWMA and KQT-EWMA variants. The empirical ARL_0 is confronted with its target values, set by the user - here between {500,1000,2000,5000}. All these models struggle to maintain empirical ARL_0 values near the target when this target increases, given both values of $N \in \{1024, 4096\}$. Results are averaged over 500 experiments.



(a)

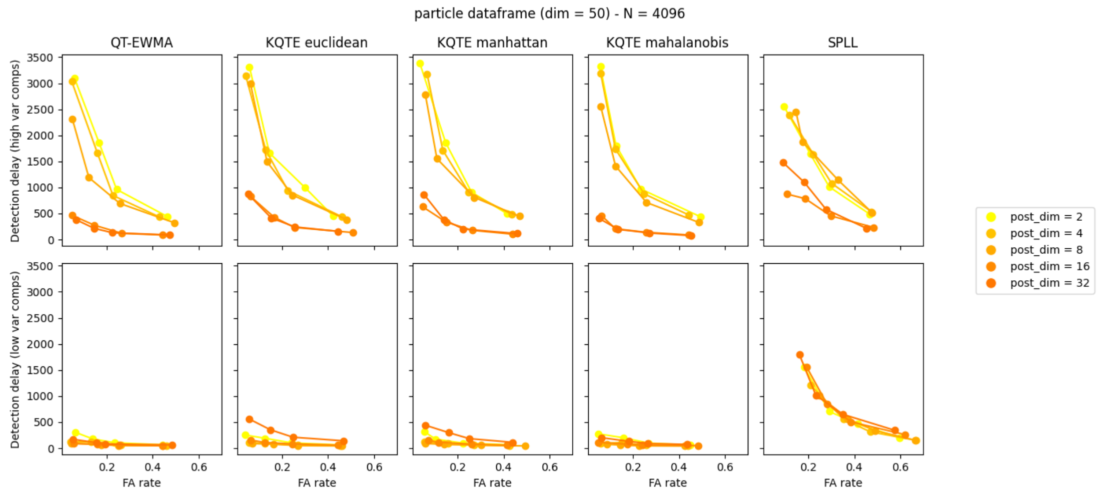


(b)

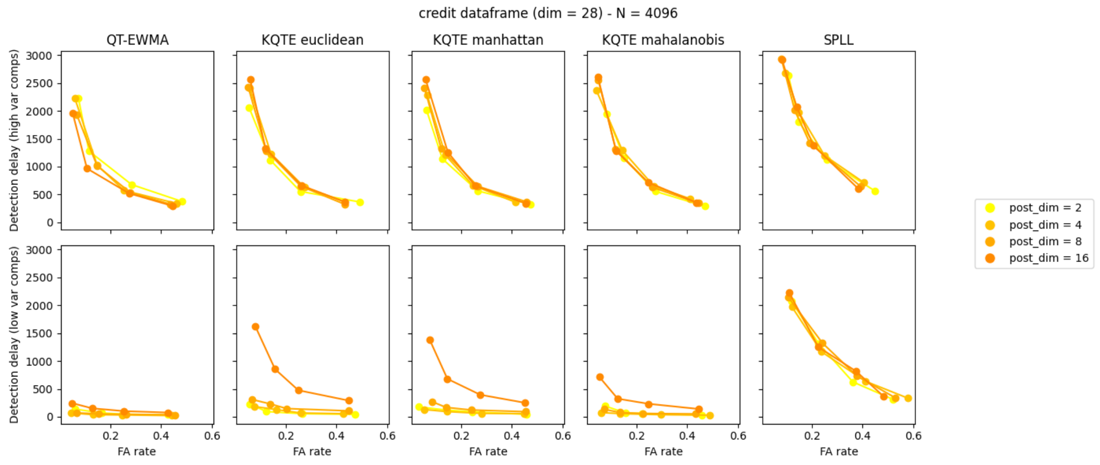


(c)

Figure 6.11: Detection Delay and False Alarms (FA) over UCI (a:"particle" ($d = 50$), b:"credit" ($d = 28$), and c:"protein" ($d = 10$)) dataframes between SPLL, QT-EWMA and KQT-EWMA variants. All these models struggle to maintain FA rates near the targets when the target increases, given both values of $N \in \{1024, 4096\}$. In these conditions, QT-EWMA achieves lower detection delays with respect to its generalized version when l_p norms are used. If a sufficient number of training points is provided, KQT-EWMA with Mahalanobis distances is our best try. Results are averaged over 500 experiments.



(a)



(b)

Figure 6.12: **PCA** - Detection Delay and False Alarms (FA) over UCI "particle" ($d = 50$) dataframe when preprocessing with PCA and projecting data on a subset of high (a) and low (b) variance components. The projected space dimensionality takes values in $\{2, 4, 8, 16, 32\}$. Results shows once again that low variance components are more likely to contain the subspace where the change happens. Indeed, to achieve the lower detection delays keeping PC selected by a sample based PCA, it is convenient to hold a greater number of components. Results are averaged over 500 experiments.

Chapter 7

Conclusions

Concept Drift Detection frameworks give a comprehensive view of the challenges and strategies inherent in monitoring predictive models under dynamic data conditions. The thesis work confronted the complexity of high-dimensional multivariate data streams, studying the performances of the QT algorithm, its generalized version Kernel-QT (KQT), and its online variant, QT-EWMA. This exploration finally included the proposal of a novel online algorithm, KQT-EWMA, which combines a generalized QT histogram with an exponentially weighted moving average statistic and outshines both QT-EWMA and the "oracle" SPLL, both in terms of controlling ARL_0 and in achieving impressively low detection delays when N/d ratio is in favor. We considered the interplay between dimensionality, training data availability, and the choice of distance metrics, together with conventional data processing methods such as PCA; in particular we considered intricacies in FPR control with respect to the significance of the choice of components for building the projected space.

The conclusions of this study show paths for future explorations and real-world applications: we tried to set a controlled but comprehensive framework for establishing the limits of these algorithms, but in doing so, we just discovered new questions, and the music is just starting. Between all these N/d inconvenient ratios we remark that simplicity is often the best answer: QT and QT-EWMA consistently excel, both with and without data preprocessing, competing favorably against significantly heavier algorithms in terms of TPR/detection delay and FPR/ARL_0. Their simplicity and elegance, coupled with a foundation in robust theoretical results, make them stand out. I believe that a potentially important exploration should be done on the behavior of ensembles of QuantTree models. Constructing multiple QT histograms on a training set derived from randomized projections of a high-dimensional space. First of all, randomized feature selection (e.g. Gaussian RPs we used) induces an implicit regularization; moreover, this ensemble would satisfy the requirements of diversity, independence, and superiority of each member to random choices, potentially resulting in a highly efficient system.

Chapter 8

Conclusioni e ringraziamenti

Un quadro sul rilevamento di derive del concetto: scrivere in italiano di "Concept Drift Detection" mi è complicato tanto quanto provare a raccontare questi risultati della statistica e dell'analisi dati - anche solo "big data"; come suona male "dati grandi".

Abbiamo visto questioni e strategie per il monitoraggio di modelli predittivi in condizioni dinamiche, passando per la complessità dei flussi di dati multivariati ad alta dimensionalità, studiando gli ottimi risultati dell'algoritmo QT che rimane un semplice istogramma, della sua versione generalizzata Kernel-QT (KQT), e della sua variante online, QT-EWMA. Abbiamo proposto un nuovo algoritmo per l'analisi online, KQT-EWMA, che combina un istogramma QT generalizzato con la statistica EWMA e si distingue sia nel controllo di ARL_0 che nel rilevare cambiamenti nei pattern in tempi incredibilmente bassi.

Abbiamo considerato l'interazione tra dimensionalità, disponibilità di dati, e la scelta di metriche di distanza, insieme a metodi di elaborazione dati e riduzione della dimensionalità convenzionali come PCA; in particolare, abbiamo esaminato le complessità nel controllo del tasso di falsi positivi (FPR) rispetto all'importanza della scelta dei componenti per la costruzione dello spazio proiettato.

Le conclusioni sperimentali indicano nuove vie e applicazioni a problemi reali: abbiamo cercato di stabilire un quadro complessivo ma controllato per definire i limiti e i benefici di questi algoritmi, ma nel farlo abbiamo solo trovato nuove domande, e la musica è appena iniziata.

Grazie a Bianca, senza di te, le tue idee, le tue parole, sarei poco poco. A te, tutte le simulazioni Monte Carlo.

Grazie ad Alice, ai cinemini alle letture alle mucche armene alla musica che abbiamo visto insieme; quanto ci sei tu in queste pagine: chissà se lo vedi, anche se lo nego.

Un grande grazie a mia madre che mi ha permesso di fare questi tanti e tanto belli anni di università, a tutta la famiglia allargata; non ho e non basterebbero altre parole.

Un grande grazie a Bea, Ceci, Giacomo, Michele, Mone, Taguhi, Tommi, ai fisici tossici, e tutt* l* altr* student* del Patio per avermi passato la fisica, la matematica, l'ostetricia, la lingua araba e le altre cosette che mi hanno permesso di superare gli esami e scrivere questa tesi.

Soprattutto un grande grazie a Filippo che mi ha guidato, tra tutti questi esperimenti, attraverso un oceano - letteralmente due, nelle nostre call settimanali Italia-Nuova Zelanda tra notti estive, mattine invernali e altri incroci; questa tesi dovrebbe portare anche la sua firma.

Nota finale: Ho fatto girare, approssimativamente, giorno e notte per quattro mesi, 40 core CPU. Sono 2880 ore in cui, mediamente stimiamo, sono richiesti 100 W. Considerando il costo medio dell'energia in kWH in Italia l'anno scorso, uguale a 38.9 g di CO_2 equivalente, il lavoro dietro questa tesi include oltre un quintale ($\sim$ 112 kg) di anidride carbonica.

Bibliography

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[2] Cesare Alippi, Giacomo Boracchi, and Diego Carrera. Ccm: Controlling the change magnitude in high dimensional data. In Plamen Angelov, Yannis Manolopoulos, Lazaros Iliadis, Asim Roy, and Marley Vellasco, editors, *Advances in Big Data*, pages 216–225, Cham, 2017. Springer International Publishing.

[3] Cesare Alippi, Giacomo Boracchi, Diego Carrera, and Manuel Roveri. Change detection in multivariate datastreams: Likelihood and detectability loss. 10 2015.

[4] Maroua Bahri, Albert Bifet, João Gama, Heitor Murilo Gomes, and Silviu Maniu. Data stream analysis: Foundations, major tasks and tools. *WIREs Data Mining and Knowledge Discovery*, 11, 03 2021.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975.

[6] Giacomo Boracchi, Diego Carrera, Cristiano Cervellera, and Danilo Macciò. QuantTree: Histograms for change detection in multivariate data streams. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 639–648. PMLR, 10–15 Jul 2018.

[7] Giacomo Boracchi, Cristiano Cervellera, and Danilo Macciò. Uniform histograms for change detection in multivariate data. pages 1732–1739, 05 2017.

[8] Emmanuel J. Candes, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis?, 2009.

[9] Andrew Béla Frigyik, Amol Kapila, and Maya R. Gupta. Introduction to the dirichlet distribution and related processes. 2010.

[10] Luca Frittoli, Diego Carrera, and Giacomo Boracchi. Change detection in multivariate datastreams controlling false alarms. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 421–436, Cham, 2021. Springer International Publishing.

[11] Minos N. Garofalakis and Johannes Gehrke. Querying and mining data streams: you only get one look a tutorial. In *ACM SIGMOD Conference*, 2002.

[12] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.

[13] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000.

[14] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository.

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

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

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

[18] Michael Lexa. Useful facts about the kullback-leibler discrimination distance. 01 2004.

[19] Anjin Liu, Jie Lu, and Guangquan Zhang. Concept drift detection via equal intensity k-means space partitioning. *IEEE Transactions on Cybernetics*, 51(6):3198–3211, jun 2021.

[20] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2018.

[21] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space, November 1901.

[22] Abdulhakim A. Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In

Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, page 935–944, New York, NY, USA, 2015. Association for Computing Machinery.

[23] Bernhard Schölkopf, Alex Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 07 1998.

[24] Vinicius M. A. Souza, Denis M. dos Reis, André Gustavo Maletzke, and Gustavo E. A. P. A. Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34:1805 – 1858, 2020.

[25] Diego Stucchi, Paolo Rizzo, Nicolò Folloni, and Giacomo Boracchi. Kernel quanttree. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[26] M.E. Tipping. Deriving cluster analytic distance functions from gaussian mixture models. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 815–820 vol.2, 1999.