Kernel QuantTree

Diego Stucchi [1] **Paolo Rizzo** [1] **Nicoló Folloni** [1] **Giacomo Boracchi** [1]

Abstract

We present Kernel QuantTree (KQT), a non-parametric change detection algorithm that monitors multivariate data through a histogram. KQT constructs a nonlinear partition of the input space that matches pre-defined target probabilities and specifically promotes compact bins adhering to the data distribution, resulting in a powerful detection algorithm. We prove two key theoretical advantages of KQT: *i*) statistics defined over the KQT histogram do not depend on the stationary data distribution ϕ_0, so detection thresholds can be set a priori to control false positive rate, and *ii*) thanks to the kernel functions adopted, the KQT monitoring scheme is invariant to the roto-translation of the input data. Consequently, KQT does not require any preprocessing step like PCA. Our experiments show that KQT achieves superior detection power than non-parametric state-of-the-art change detection methods, and can reliably control the false positive rate.

1. Introduction

Change Detection (CD) is the problem of detecting distribution changes $\phi_0 \rightarrow \phi_1$ in a datastream, namely detecting when the data-generating process drifts from a stationary distribution ϕ_0 towards an unknown post-change distribution ϕ_1. Here, we address the problem of *batch-wise CD*, where data are analyzed in fixed-size batches that, under normal conditions, contain samples drawn from ϕ_0. The timely detection of distribution changes and the control over the false alarm rate are fundamental problems that have been widely explored in both the Machine Learning (Gama et al., 2014) and Statistical Process Control (Basseville et al., 1993) literature. Among the many applications of change-detection algorithms, we mention fault detection (Tartakovsky et al., 2006), financial monitoring (Ross et al., 2011), cryptographic attacks (Frittoli et al., 2020), and quality control (Hawkins et al., 2003).

Most CD algorithms consist of three major ingredients: *i*) a *model* $\widehat{\phi}_0$ describing stationary data, which is usually learned from a training set, *ii*) a *statistical test*, where a test statistic $\mathcal{T}$ is computed to assess the consistency of incoming data to $\widehat{\phi}_0$, and *iii)* a decision rule on $\mathcal{T}$ to establish whether a change has occurred. In many real-world multivariate scenarios, estimating a density model for the stationary data is often unfeasible. Therefore, non-parametric methods that describe stationary data by flexible models are preferred. Unfortunately, most non-parametric statistics are based on ranking (Ross & Adams, 2012) and can only be applied to univariate data. In Section 3, we overview the few non-parametric solutions to monitor multivariate datastreams. A relevant example is QuantTree (QT) (Boracchi et al., 2018), a change detection algorithm based on a histogram partitioning of the input space, which is supported by sound theoretical results. In particular, QT allows to operate at a controlled false alarm rate without knowing ϕ_0 nor resorting to bootstrap to estimate detection thresholds.

A fundamental limitation of QT is that splits are defined along the axis, as in Figure 1(a), resulting in a partitioning that does not always adhere to the input distribution. To mitigate this problem, a preprocessing stage is typically introduced to align the split directions to the principal components of the training set, as shown in Figure 1(b). While this procedure is often beneficial, we observe (Section 6) that it can worsen the detection performance in some unpredictable cases. Moreover, many bins in Figure 1(a)(b) have non-finite volumes, which can lead to poor estimation of bin probabilities.

In this paper, we introduce Kernel QuantTree (KQT), a non-parametric and multivariate CD algorithm that constructs histogram bins via measurable kernel functions, resulting in a powerful CD test. In contrast with the QT algorithm, which constructs bins by axis-aligned splits, 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 1(c)-(d)-(e) shows that the KQT bins are compact subsets of the domain. Our intuition is that compact bins increase the flexibility when

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy. Correspondence to: Diego Stucchi <diego.stucchi@polimi.it>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

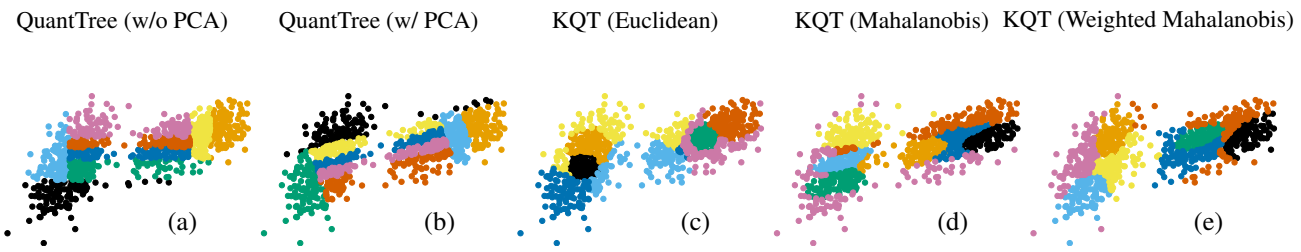

Figure 1. QuantTree generates bins as intersection of hyperplanes, performing cuts along the axis (a). After a preprocessing through PCA, the cuts are oriented along the principal directions (b). Kernel QuantTree generates bins that are subsets of d-dimensional spheres according to the underlying kernel functions, namely the Euclidean (c), Mahalanobis (d) and Weighted Mahalanobis (e) distances.

modeling ϕ_0 by fitting a histogram h to training data. Moreover, estimating the bin probabilities under ϕ_0, which are fundamental to compute the test statistic $\mathcal{T}_h$, is less accurate on non-compact bins.

In Section 5, we prove that KQT features two theoretical properties that have significant implications in change detection. First, 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 as in QT, without knowing ϕ_0. Second, 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 Section 6. In particular, KQT achieves better detection performance than the alternatives independently of preprocessing steps based on roto-translations.

In summary, these are our main contributions:

i) We present Kernel QuantTree, a non-parametric CD method based on a histogram where bins are defined by kernel functions. KQT achieves state-of-the-art detection performance on multivariate datastreams;

ii) We prove that statistics defined over the KQT histograms do not depend on ϕ_0, but only on few KQT parameters. This enables control of the FPR by thresholds τ set a priori, via Monte Carlo simulations;

iii) We prove that the monitoring performed by KQT is independent of any preprocessing by roto-translations.

2. Problem Formulation

We address the problem of change detection in batch-wise monitoring settings, where stationary data are realizations of a random vector $\mathbf{X}$ with unknown probability density function ϕ_0. We assume that a training set of stationary samples $\mathrm{TR} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^d$ is provided, and that incoming data are processed in batches W of $\nu \in \mathbb{N}$ samples each. We denote as $W \sim \phi_0$ when all the samples in the batch W are drawn from ϕ_0.

Our goal is to design a CD algorithm that: *i)* detects distribution changes in incoming batches, and *ii)* controls the False Positive Rate (FPR), namely the probability of mistakenly detecting a change in stationary data. We formulate this CD problem as a Hypothesis Test to establish whether $W \sim \phi_0$ (null hypothesis) or $W \sim \phi_1 \neq \phi_0$, where ϕ_1 is the unknown post-change distribution. We pursue the mainstream approach of computing a test statistic $\mathcal{T}$ on each batch W and detecting a change when

$$\mathcal{T}(W) > \tau, \tag{1}$$

where $\tau \in \mathbb{R}$ is the threshold that we set to control the FPR.

For the sake of simplicity, we assume that a batch W is either drawn from ϕ_0 or from a different unknown distribution $\phi_1 \neq \phi_0$. However, CD algorithms can in principle detect batches drawn from a mixture of ϕ_0 and ϕ_1, even though the detection power is expected to be lower in this case.

3. Related Work

Change detection in multivariate datastreams is a challenging problem, which can be significantly simplified when ϕ_0 belongs to a known parametric family since the model $\widehat{\phi}_0$ is obtained by estimating its parameters. The most popular solutions pursuing this approach consist in monitoring the likelihood of incoming data with respect to $\widehat{\phi}_0$ fitted on TR. Viable options for $\widehat{\phi}_0$ are Gaussian process (Saatçi et al., 2010), Gaussian Mixtures (Kuncheva, 2011) or kernel density estimators (Krempl, 2011). In (Kuncheva, 2011) and (Kuncheva & Faithfull, 2013), the Semiparametric Log-Likelihood (SPLL) algorithm fits a Gaussian Mixture Model (GMM) to TR and compares incoming batches with batches from TR by a likelihood test. Moreover, in SPLL, it is not possible to set a priori the detection threshold to control the FPR, as the distribution of the test statistic depends on ϕ_0. Moreover, adopting a GMM to approximate ϕ_0 might not

always fit real-world data, as demonstrated by our experiments on high-dimensional datasets.

There are only a few recent multivariate methods that perform non-parametric change detection, namely, that assume that ϕ_0 and ϕ_1 are unknown. Among these, we focus on histogram-based algorithms, since these are non-parametric by design and can efficiently process datastreams in batches. As such, histogram-based algorithms represent very practical solutions to monitor multivariate datastreams. Density Tree (Criminisi et al., 2012) constructs a space partitioning by iteratively splitting regions to maximize an information-gain metric. In this case, the distribution of the test statistic depends on ϕ_0, thus detection thresholds need to be set by bootstrap on TR. Equal Intensity K-means (EIKM) (Liu et al., 2020) 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 (Boracchi et al., 2018) defines a partitioning $\mathcal{S}$ of the input space in K 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. The splits are performed such that the probability of a stationary sample to fall in each bin is close to a set of target probabilities $\{\pi_k\}$ provided as input parameters. 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 (Section 6). KQT 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.

4. Kernel QuantTree

We present Kernel QuantTree (KQT)[1], a CD algorithm that solves a major limitation of QuantTree (QT) while generalizing and extending its theoretical guarantees. 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.

[1]Code available at github.com/diegocarrera89/quantTree.

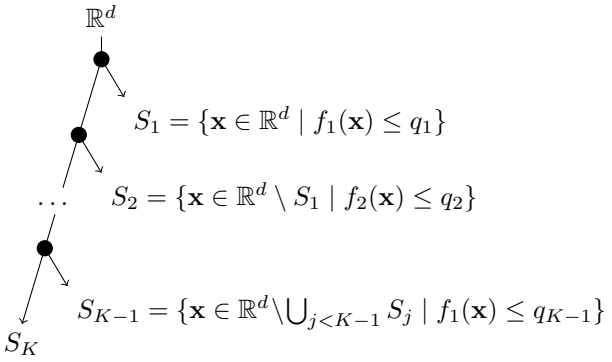

Figure 2. The Generalized QuantTree histogram is a binary splitting tree where splits isolate leaves, i.e. bins of the histogram.

As illustrated in Figure 2, a GQT partitioning corresponds to an extremely imbalanced binary tree, where each split isolates a leaf, corresponding to a bin S_k. In what follows (Section 4.1 and 4.2), we illustrate in detail the GQT partitioning scheme, providing a few examples of kernel functions. In Section 5, we demonstrate 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 in (1) a priori by Monte Carlo simulations.

The monitoring 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 (Lehmann et al., 2005):

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

where $\{\pi_k\}$ are the target bin probabilities. In Section 5.3, we also prove 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. Finally, in Section 4.4, we analyze the computational complexity of KQT both at the training and monitoring stages.

4.1. Generalized QuantTree (GQT) Partitioning

The two elements defining each bin in a GQT are: *i)* a *measurable function* $f_k : \mathbb{R}^d \to \mathbb{R}$ mapping multivariate data to a single dimension and *ii)* a *split value* $q_k \in \mathbb{R}$, chosen to match the target probability π_k. Algorithm 1 illustrates the GQT histogram construction, which requires

Algorithm 1 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 $\widetilde{\pi}_k = \pi_k(1 - \sum_{j<k} \pi_j)^{-1}$
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: $S_k = \{\mathbf{x} \in \bigcap_{j<k} \overline{S}_j \mid f_k(\mathbf{x}) \leq q_k\}$
9: $\mathcal{X}_k = \{\mathbf{x} \in \mathcal{X}_{k-1} \mid f_k(\mathbf{x}) > q_k\}$
10: **end for**
11: $S_K = \mathbb{R}^d \setminus \bigcup_{j<K} S_j$

as input a training set TR and the target probabilities $\{\pi_k\}$. The rationale underpinning the space partitioning of KQT is to iteratively construct bins by selecting sublevel sets of the kernel functions f_k. In particular, S_1 is defined as the sublevel set of f_1 with respect to the split value q_1:

$$S_1 = \{\mathbf{x} \in \mathbb{R}^d \mid f_1(\mathbf{x}) \leq q_1\}. \tag{3}$$

The other bins $S_k, 1 < k < K$ are obtained by isolating sublevel sets from the space that is not yet assigned to a bin

$$S_k = \{\mathbf{x} \in \bigcap_{j<k} \overline{S_j} \mid f_k(\mathbf{x}) \leq q_k\} \text{ for } k < K, \tag{4}$$

where $\overline{S_j}$ denotes the complement of S_j in $\mathbb{R}^d$. The last bin, $S_K = \mathbb{R}^d \setminus \bigcup_{j<K} S_j$, is the *residual* bin, which contains all the points that do not fall in the previous bins.

The split values q_k are defined at each iteration by first identifying the set $\mathcal{X}_k \subset \text{TR}$ which contains the training points that do not fall in any bin $\{S_j \mid j < k\}$. In the beginning, no training samples have been assigned to a bin, thus, we set $\mathcal{X}_0 = \text{TR}$ and $N_0 = |\text{TR}|$ (line 3). At the k-th step, we first the percentage of points of $\mathcal{X}_{k-1}$ that must fall in S_k to meet the target probability π_k (line 5), which we denote as:

$$\widetilde{\pi}_k = \pi_k \Big(1 - \sum_{j<k} \pi_j\Big)^{-1}, \tag{5}$$

such that S_k contains $\pi_k N = \widetilde{\pi}_k |\mathcal{X}_{k-1}|$ points. Then, we evaluate f_k on all the samples in $\mathcal{X}_{k-1}$ (line 6), and compute the split value q_k as the $\widetilde{\pi}_k$-quantile of the projected samples $\{f_k(\mathbf{x}), \mathbf{x} \in \mathcal{X}_{k-1}\}$ (line 7). Finally, we define S_k as in (4) (line 8), and we update the set of points $\mathcal{X}_k$ that will be used to construct the next bin (line 9). The process results in the GQT histogram $h = \{(S_k, \widehat{\pi}_k)\}_{k=1}^{K}$, where $\widehat{\pi}_k$ is the percentage of training points that fall in S_k and approximates the probability of $\mathbf{x} \sim \phi_0$ to fall in S_k.

The GQT extends the partitioning scheme underpinning QT, which corresponds to using linear split functions:

$$f_k(\mathbf{x}) = \pm 1 \cdot P_j \mathbf{x}, \tag{6}$$

where P_j is the projection over a randomly selected component j and ± 1 randomly introduces a sign flip for the projection. In the next section, we present specific measurable functions f_k that we employ in KQT.

4.2. Employed Kernel Functions in KQT

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

$$f_k(\mathbf{x}) = (\mathbf{x} - \mathbf{c}_k)^T A (\mathbf{x} - \mathbf{c}_k), \tag{7}$$

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 $\{S_k\}$ in (4) 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.

Here, we construct KQT using the Euclidean, the Mahalanobis, and the Weighted Mahalanobis (Tipping, 1999) distances, whose bins are illustrated in Figure 1. We obtain the Euclidean distance by setting $A = \mathbb{I}_d$, namely, the d-dimensional identity matrix, resulting in isotropic bins. Figure 1(c) shows that these bins poorly fit the data 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 1(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 multiple bins might span different clusters. To promote bins containing samples from a single cluster, we adopt the Weighted Mahalanobis distance and fit a Gaussian Mixture of M components to TR, and then assign a larger distance to points that belong to different components of the GMM. In KQT, we use $M = 4$ components, and the Weighted Mahalanobis kernel matrix is then defined as:

$$A(\mathbf{x}) = \frac{\sum_{m=1}^{M} \rho_m \cdot i_m(\mathbf{x}, \mathbf{c}) \cdot C_m^{-1}}{\sum_{m=1}^{M} \rho_m \cdot i_m(\mathbf{x}, \mathbf{c})}, \tag{8}$$

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 in (8) represents a weighted average of the inverse covariance matrices of the GMM components. As in (Tipping, 1999), 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 $\mathbf{c}$.

In the following, we discuss the centroid selection strategy employed in KQT.

4.3. Centroid Selection

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. Therefore, we select the centroid in TR by optimizing a partition-quality metric I, namely,

$$\mathbf{c}_k = \underset{\mathbf{c} \in \mathcal{X}_{k-1}}{\operatorname{argmax}} I[\mathbf{c}], \tag{9}$$

where $\mathcal{X}_{k-1}$ are the training samples used to construct S_k, and $I[\mathbf{c}]$ denotes the value of the metric when we select $\mathbf{c}$ as a centroid. In KQT, we consider two centroid selection strategies: *i)* maximizing the *information gain* associated with the split and *ii)* minimizing the *Gini index* of the distances to the centroid. For computational reasons, when $\mathcal{X}_{k-1}$ is large, we restrict the search space to a subset of randomly sampled potential centroids $\{\mathbf{c}\} \subset \mathcal{X}_{k-1}$.

The *information gain* (Mitchell, 1997) measures the decrease in the overall entropy H after a split in the data and is typically used to assess the split quality in a data set, for example by Density Tree (Criminisi et al., 2012). The best split lowers the data entropy, maximizing the information gain. In KQT, we compute the information gain yielded by the split that divides $\mathcal{X}_{k-1}$ into $\mathcal{X}_k$ and $\overline{\mathcal{X}}_k = \mathcal{X}_{k-1} \setminus \mathcal{X}_k$. In particular, we can compute the entropy $H(B)$ of a set of points B using the Gaussian approximation:

$$H(B) = (1/2) \log \Big((2\pi e)^d \det(\operatorname{cov}[B]) \Big). \tag{10}$$

where $\operatorname{cov}[B]$ represents the sample covariance matrix computed over B. The information gain associated with the centroid $\mathbf{c}$ is defined as

$$I[\mathbf{c}] = |\mathcal{X}_{k-1}|\, H(\mathcal{X}_{k-1}) - (|\overline{\mathcal{X}}_k|\, H(\overline{\mathcal{X}}_k) + |\mathcal{X}_k|\, H(\mathcal{X}_k)). \tag{11}$$

In the supplementary material, we discuss the simplifications we introduced to lower the computational burden of assessing (11) for multiple potential centroids, like the Gaussian approximation of H, which does not influence the non-parametric nature of KQT.

The *Gini index* (Gini, 1912) measures the level of uniformity in an empirical distribution and takes values between 0 (perfect equality) and 1 (maximum inequality). In KQT, we use the Gini index to prevent the selection of centroids in low-density regions. Specifically, we compute the Gini index of the distances between the training samples and the centroid as

$$I[\mathbf{c}] = \frac{\sum_{i,j} |f_k(\mathbf{x}_i) - f_k(\mathbf{x}_j)|}{2\, |\mathcal{X}_{k-1}| \sum_i f_k(\mathbf{x}_i)}. \tag{12}$$

We select the centroid that minimizes (12) to promote bins that cover densely populated regions of the input space.

4.4. Computational Remarks

In terms of computation cost, the training 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 $O(N)$, and *iii)* the centroid selection. The cost of computing the Euclidean distance is $O(d)$, while the Mahalanobis costs $O(d^2)$ and the Weighted Mahalanobis costs $O(Md^2)$, 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 in (10), which costs $O(d^3)$ 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 by (9). Therefore, an upper bound for the cost of KQT construction is $O(KT(N + MNd^2 + d^3))$ 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 $O(\nu KMd^2)$ in case of the Weighted Mahalanobis distance.

Table 1 reports the complexity of all the methods considered in our experiments, showing that KQT with the Weighted Mahalanobis distance is most computationally demanding, both in terms of training and inference. However, the experiments discussed in Section 6 prove that this cost is balanced by superior detection performance.

Table 1. Comparison of the computational complexity of KQT and the other considered methods, where M is the number of Gaussian components employed by KQT with the Weighted Mahalanobis distance, and R is the number of splits of Density Tree.

Method	Training Cost	Inference Cost
KQT (Weighted Maha.)	$O(KT(N + MNd^2 + d^3))$	$O(\nu KMd^2)$
QuantTree	$O(KN \log N)$	$O(\nu K)$
EIKM	$O(K^2 N \log N)$	$O(\nu K)$
SPLL	$O(Nd^2)$	$O(\nu d^2)$
Density Tree	$O(KRd^3)$	$O(\nu K)$

5. Theoretical Guarantees

This section illustrates the theoretical properties of KQT and is organized as follows. In Section 5.1, we prove that the distribution of the test statistic computed by GQT over stationary data is independent of ϕ_0, hence generalizing the main result of QT from (Boracchi et al., 2018) to a more extensive set of histogram-based monitoring schemes, including KQT. Then, in Section 5.2, we show how to exploit this result to compute detection thresholds by Monte Carlo

simulations such that the empirical FPR matches any target value α. Finally, in Section 5.3, we prove that KQT is invariant to roto-translations of the data when we use the kernel functions in Section 4.2.

5.1. Generalization of the QT Independence Theorem

The following result implies that the distribution of a test statistic like (2) computed over stationary batches by a GQT is independent of ϕ_0, the input dimension d and the employed functions f_k. Thus, such distribution can be empirically computed via Monte Carlo simulations and used in any monitoring scenario as long as $\{\pi_k\}_k$, the training set size N, and the batch size ν are fixed.

Theorem 5.1. *Let $h = \{(S_k, \widehat{\pi}_k)\}_{k=1}^K$ 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 $y_1, \dots, y_K$ 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 $\{\pi_k\}_k$.*

The proof of Theorem 5.1 follows three propositions as the proof of Theorem 1 in (Boracchi et al., 2018), which we generalize to a broader set of partitioning schemes. The first proposition states that the probability of a point drawn from the stationary distribution ϕ_0 to fall in any bin of a GQT histogram follows a Beta distribution.

Proposition 5.2. *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 $Z = f(X)$. 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*

$$Q_{f,L} := \{\mathbf{x} \in \mathcal{D} : f(\mathbf{x}) \leq z_{(L)}\}. \tag{13}$$

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

The proof of Proposition 5.2 is reported in the supplementary material. In the following, we denote the probability of a stationary point $\mathbf{x} \sim \phi_0$ to fall in bin k as $p_k = P_{\phi_0}(\mathbf{x} \in S_k)$. Moreover, we denote as $\widetilde{p}_k = P_{\phi_0}(\mathbf{x} \in S_k \mid \mathbf{x} \notin \bigcup_{j<k} S_j)$ the probability of $\mathbf{x}$ to fall in S_k and not in any of the previous bins.

Proposition 5.3. *For a Generalized QuantTree histogram, the following relation holds:*

$$p_k = \widetilde{p}_k \cdot \Big(1 - \sum_{j<k} p_j\Big) = \widetilde{p}_k \prod_{j<k} \Big(1 - \widetilde{p}_j\Big). \tag{14}$$

Proposition 5.4. *For a Generalized QuantTree histogram, the random variables $\{\widetilde{p}_k\}$ are independent.*

The proofs of Propositions 5.3 and 5.4 are equivalent to the proofs of the Proposition 2 and 3 for QT (Boracchi et al., 2018). Finally, the proof of Theorem 5.1 follows from Propositions 5.2-5.3-5.4 and from the following facts: *i)* the employed statistic (2) only depends on the number of samples falling in each bin $\{y_k\}$, and *ii)* when the batch is drawn from ϕ_0, the vector $[y_1, \dots, y_K]$ is a realization of a Multinomial distribution of parameters $(\nu, p_1, \dots, p_K)$.

5.2. Threshold Computation for FPR Control

From Theorem 5.1, it follows that in GQT we can compute a detection threshold $\tau = \tau(\alpha)$ yielding an FPR α when used as in (1) for any test statistic $\mathcal{T}_h$ that only depends on $\{y_k\}$. For this purpose, we estimate by Monte Carlo simulations the empirical distribution of $\mathcal{T}_h$ on stationary batches. Interestingly, (Frittoli et al., 2022) prove that the empirical distribution of N samples drawn from ϕ_0 in a QT histogram follows a Dirichlet distribution of parameters $\{\pi_1 N, \dots, \pi_{K-1} N, \pi_K N + 1\}$, where $\{\pi_k\}$ are the target probabilities used for constructing the histogram. Since the projection function does not influence the proof in (Frittoli et al., 2022), the same result holds for GQT. Moreover, the distribution of a batch $W \sim \phi_0$ in the histogram bins follows a Multinomial distribution. Thus, we can efficiently simulate the construction of a GQT and compute the values $\mathcal{T}_h$ over $W \sim \phi_0$ by Monte Carlo simulations, such that τ is the $(1 - \alpha)$-quantile of the resulting distribution.

Remarkably, the distribution of the test statistic does not depend on the employed kernel functions $\{f_k\}$. Therefore, we can use the same detection thresholds for any GQT that uses any measurable function f_k. Moreover, these detection thresholds also work for QT, which is a special case of GQT that uses (6).

5.3. Roto-Translation Invariance of KQT

In this section, we prove that KQT is invariant under roto-translations when the employed kernel function is either the Euclidean, Mahalanobis or Weighted Mahalanobis distance. We denote a roto-translation as $\Phi : \mathbb{R}^d \to \mathbb{R}^d$, and the image of a set $B \subset \mathbb{R}^d$ as $\Phi(B) = \{\Phi(\mathbf{x}) \mid \mathbf{x} \in B\}$. The following theorem states that the two KQT histograms $h = \{(S_k, \widehat{\pi}_k)\}$ and $h' = \{(S'_k, \widehat{\pi}'_k)\}$, constructed with and without preprocessing by Φ, respectively, are equivalent.

Theorem 5.5. *Let $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ be a roto-translation. Let $h = \{(S_k, \widehat{\pi}_k)\}$ and $h' = \{(S'_k, \widehat{\pi}'_k)\}$ be the KQT histograms constructed from the training sets $\mathrm{TR} \subset \mathbb{R}^d$ and $\mathrm{TR}' = \Phi(\mathrm{TR})$, where the kernel function is either the Euclidean, Mahalanobis or Weighted Mahalanobis distance. Then, we have that $S'_k = \Phi(S_k)$ and $\widehat{\pi}'_k = \widehat{\pi}_k$ for $k = 1 \dots, K$. In particular, for any batch W, if we compute $W' = \Phi(W)$, we have that $\mathcal{T}_h(W) = \mathcal{T}_{h'}(W')$.*

Theorem 5.5 proves that, for the considered kernel functions, the value of the test statistic computed over a batch W of data does not change when we employ a roto-translation Φ for preprocessing, including PCA, that is sometimes required for QT to achieve good detection performance. The proof of Theorem 5.5 is reported in the supplementary material and relies on the fact that the construction of the two histograms uses the same points (up to Φ), namely $\mathcal{X}'_k = \Phi(\mathcal{X}_k)$, to define each bin. In particular, we need to prove that our centroid selection strategy results in the same centroid (up to Φ) for S_k and S'_k:

Lemma 5.6. *Let $\mathcal{X}_{k-1}$ and $\mathcal{X}'_{k-1} = \Phi(\mathcal{X}_{k-1})$ be the set of points used to construct the KQT histogram bins S_k and S'_k, respectively. Then, the centroid selection by (9) results in centroids $\mathbf{c}_k$ and $\mathbf{c}'_k = \Phi(\mathbf{c}_k)$.*

In the supplementary material, we prove this lemma, and we use it in the proof of Theorem 5.5 to show by induction that $S_k = S'_k$ for every k. Consequently, every batch W will result in the same value of test statistic because if a point $\mathbf{x} \in W$ falls in S_k, then $\Phi(\mathbf{x}) \in W'$ will fall in S'_k.

6. Experiments

In this section, we validate KQT through several experiments, proving that KQT *i)* reaches detection performance that are statistically superior than state-of-the-art algorithms, and *ii)* can accurately control the FPR.

6.1. Datasets

We present the synthetic and real-world datasets that we employ in our experiments.

In each experiment, we consider TR made of $N = 4096$ points sampled without replacement by ϕ_0. During testing, we randomly sample 5000 batches of ν samples from ϕ_0 and 5000 batches of ν samples from ϕ_1 to robustly assess detection performance.

Synthetic. We consider two synthetic settings with $d = 4$, the *unimodal* and the *bimodal*. In the unimodal setting, the stationary distribution ϕ_0 is a 0-mean Gaussian with a random covariance matrix. The post-change distribution ϕ_1 is obtained by roto-translation using the CCM framework (Alippi et al., 2017), such that the Kullback-Leibler distance between ϕ_0 and ϕ_1 is 1. In the bimodal setting, ϕ_0 is a Gaussian mixture of two slightly-overlapping components, and ϕ_1 is again generated by a roto-translation of each component of ϕ_0 computed using CCM. In the supplementary material, we discuss the same experiment performed with $d \in \{4, 8, 16, 32, 64, 128\}$.

INSECTS. The INSECTS dataset (Souza et al., 2020) contains feature vectors ($d = 33$) extracted from sensor measurements describing the wing-beat frequency of six (annotated) species of flying insects. This dataset contains real changes caused by temperature modifications that affect the insects' flying behavior. We set up the change detection experiment such that ϕ_0 describes measurements acquired at a temperature, and the change $\phi_0 \to \phi_1$ corresponds to a temperature change. We denote as $i \to i+1$ the considered temperature changes, with $i \in \{1, 2, 3, 4, 5\}$.

UCI. We employ real-world datasets from the UCI Machine Learning Repository (Dua & Graff, 2017) and from (Dal Pozzolo et al., 2017), with dimensions ranging from $d = 5$ to $d = 50$, reported in Table 2. We standardize these datasets and add a negligible amount of noise $\eta \sim N(0, \sigma)$ to each component to prevent the many repeated values from harming the histogram construction. The values of σ for each dataset are reported in the supplementary material. These datasets contain no distribution changes, thus stationary samples are drawn by sampling the dataset. We generate a post-change distribution ϕ_1 by shifting stationary data in a random direction with a magnitude proportional to the variance of each component.

Swarm. The Swarm Behavior classification dataset from the UCI Machine Learning Repository (Dua & Graff, 2017) comprises high-dimensional data ($d = 2400$) describing the motion of large groups of animals, which are labeled as flocking or not-flocking. We define the stationary distribution ϕ_0 as the distribution of data describing flocking groups of animals. In contrast, the post-change distribution ϕ_1 is defined by data corresponding to non-flocking animals.

High-dimensional datasets represent a challenging scenario for change detection algorithms, especially when they require estimating a density model. Therefore, these algorithms typically employ dimensionality-reduction techniques to map data to lower dimensions (Thudumu et al., 2020). To show that high-dimensional problems can be tackled by KQT upon employing such techniques, in our experiments we apply a PCA-based preprocessing step, retaining the 32 components explaining the most variance in the data.

6.2. Figures of Merit

We assess the performance of CD algorithms with two standard figures of merit, FPR and AUC. 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 AUC, and we report their average rank (Demšar, 2006) over all the datasets and over 500 runs of each experiment. Moreover, we report the p-values of the Nemenyi post-hoc test (Nemenyi, 1963), comparing the AUCs of each method against the best-performing one. In Table 2, we mark in bold the largest AUC achieved over each dataset. We also underline values when the Nemenyi test confirms that the difference with the second best-performing

Table 2. FPR/AUC achieved by the considered methods with $K = 16$ bins and batches of $\nu = 128$ points. We report the average ranking with respect to the AUC and the p-value of the Nemenyi test. For each dataset, we mark in bold the AUCs of the best-performing method, and underline them when found to be significantly different from the second best-performing.

	d	QT (w/o PCA)	QT (w/ PCA)	KQT (Euclidean)	KQT (Mahalanobis)	KQT (Weighted Maha.)	EIkM	SPLL (C=3)	PCA-SPLL (C=3)	DT (w/o PCA)	DT (w/ PCA)
unimodal	4	4.83%/ 0.96	4.81%/ 0.98	4.86%/ 0.95	4.82%/ 0.99	4.83%/ 0.99	4.82%/ 0.87	5.46%/ **1.00**	5.92%/ 0.99	7.84%/ 0.79	7.75%/ 0.81
bimodal	4	4.80%/ 0.90	4.81%/ 0.93	4.80%/ 0.90	4.81%/ 0.95	4.80%/ **0.97**	4.82%/ 0.82	5.53%/ 0.92	6.02%/ 0.90	7.65%/ 0.75	7.62%/ 0.77
nino	5	5.04%/ 0.84	4.99%/ 0.91	5.00%/ 0.61	5.02%/ 0.90	5.01%/ **0.92**	4.83%/ 0.53	6.14%/ 0.82	7.69%/ 0.84	7.55%/ 0.73	7.57%/ 0.58
protein	9	4.97%/ 0.90	4.98%/ 0.98	4.97%/ 0.62	4.98%/ 0.99	5.03%/ **0.99**	4.88%/ 0.51	13.15%/ 0.92	8.42%/ 0.95	7.65%/ 0.70	7.64%/ 0.59
spruce	10	4.81%/ 1.00	4.83%/ 1.00	4.82%/ 0.60	4.84%/ **1.00**	4.90%/ **1.00**	4.86%/ 0.51	11.43%/ 1.00	11.56%/ 1.00	7.56%/ 1.00	7.57%/ 1.00
lodgepole	10	4.83%/ 1.00	4.82%/ 1.00	4.85%/ 0.65	4.80%/ **1.00**	4.90%/ **1.00**	4.92%/ 0.51	10.78%/ 1.00	10.89%/ 1.00	7.60%/ 1.00	7.58%/ 1.00
credit	28	4.83%/ 0.70	4.96%/ 0.87	4.89%/ 0.60	4.85%/ 0.78	5.06%/ **1.00**	4.96%/ 0.51	8.67%/ 0.60	16.06%/ 0.66	7.63%/ 0.69	7.59%/ 0.82
insects ($1 \to 2$)	33	4.92%/ 1.00	4.93%/ 0.96	4.91%/ 0.96	4.93%/ 0.97	5.19%/ 0.99	4.93%/ 0.84	5.90%/ 0.81	6.48%/ 0.87	7.57%/ **1.00**	7.60%/ 1.00
insects ($2 \to 3$)	33	4.93%/ 0.99	4.91%/ 1.00	4.92%/ 1.00	4.96%/ **1.00**	5.25%/ **1.00**	4.96%/ 0.96	5.54%/ **1.00**	6.16%/ 1.00	7.60%/ 1.00	7.59%/ 1.00
insects ($3 \to 4$)	33	4.92%/ 0.98	4.89%/ 0.90	4.90%/ 0.90	4.88%/ 0.94	5.22%/ 0.99	4.89%/ 0.83	6.09%/ 0.75	6.69%/ 0.74	7.59%/ **1.00**	7.54%/ 1.00
insects ($4 \to 5$)	33	4.92%/ 1.00	4.95%/ 1.00	4.91%/ 1.00	4.92%/ **1.00**	5.25%/ **1.00**	4.91%/ 0.95	5.48%/ **1.00**	6.01%/ 1.00	7.63%/ 1.00	7.56%/ 1.00
insects ($5 \to 6$)	33	4.91%/ 1.00	4.90%/ 0.97	4.90%/ 0.98	4.92%/ 0.99	5.26%/ 1.00	4.90%/ 0.96	5.86%/ 0.98	6.19%/ 0.98	7.61%/ **1.00**	7.63%/ 1.00
sensorless	48	4.84%/ 0.86	5.01%/ 1.00	4.82%/ 0.54	5.01%/ **1.00**	7.42%/ **1.00**	4.93%/ 0.50	4.33%/ **1.00**	4.83%/ **1.00**	7.55%/ 0.74	7.58%/ 0.60
particle	50	4.85%/ 0.89	4.87%/ 0.93	4.81%/ 0.55	4.94%/ 0.98	5.80%/ **0.99**	4.84%/ 0.51	5.93%/ 0.84	6.07%/ 0.90	7.52%/ 0.80	7.60%/ 0.54
Average Ranking		5.24	4.93	7.08	3.82	**2.98**	9.37	5.57	5.34	5.11	5.56
Nemenyi p-value		$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	-	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$

method is statistically significant. Confidence intervals are reported in the supplementary material.

6.3. Methods

We configure all the histogram-based methods to partition the space in K bins with uniform target probabilities $\pi_k = \frac{1}{K}$, as advised by (Boracchi et al., 2018). The number of bins K and the batch size ν must be chosen to guarantee that batches contain enough samples for a stable measure of these target probabilities. In particular, since histograms approximate the probability of a point falling in a bin by the number of training samples per batch that falls in each bin, we expect that a larger number of points per bin (i.e., the ratio ν/K) yield better detection performance. We confirm this by considering two settings: the high-ratio setting ($K = 16$, $\nu = 128$) and the low-ratio one ($K = 32$, $\nu = 64$).

QuantTree. QuantTree (Boracchi et al., 2018) uses a histogram to monitor incoming batches while controlling the FPR. Bins are constructed with axis-oriented splits along random components, and the changes are detected by the Pearson statistic. Detection thresholds are computed via Monte Carlo simulations. We test QT with and without PCA preprocessing of the data.

Equal Intensity K-Means (EIKM). EIKM (Liu et al., 2020) constructs a histogram with K bins using a K-means clustering to guarantee an equal proportion of stationary data in each bin. EIKM uses the Pearson statistic and its asymptotic approximation to set the detection thresholds.

Semiparametric Log-Likelihood (SPLL). SPLL (Kuncheva, 2011) models the stationary distribution ϕ_0 as a Gaussian Mixture Model (GMM) and suggests fitting $C = 3$ components. During inference, the test statistic associated with a batch W is computed by an upper bound of the log-likelihood of its samples. Since SPLL comes without a threshold computation strategy, we employ Welch's t-test (Welch, 1947) to detect batches whose average SPLL is statistically different from the training set.

PCA-SPLL. Presented in (Kuncheva & Faithfull, 2013), PCA-SPLL extends SPLL by transforming data through the PCA and monitoring by SPLL only the components with the lowest variance. Here, we keep up to 5% of the variance and fit $C = 3$ Gaussians.

Density Tree. Inspired by (Criminisi et al., 2012), the bins of Density Tree minimize the entropy after each split. Density Tree employs the Pearson test statistic, and sets the detection threshold via bootstrapping over a portion of training data. We test Density Tree with and without PCA preprocessing of the data.

6.4. Results and Discussion

Table 2 reports the FPR and AUC achieved by all the considered methods in the high-ratio setting, averaged over 500 runs. Here, we only report the performance of KQT maximizing the information gain (11) to select centroids. In the supplementary material, we also report the results for the low-ratio and a comparison showing that minimizing the Gini index (12) leads to comparable performance.

Table 3. FPR and AUC achieved by QuantTree and Kernel QuantTree on the Swarm dataset processed by a PCA to retain $d = 32$ components. In parenthesis, the standard deviation of the results.

	FPR	AUC
QuantTree	4.61% (1.65%)	1.00 (0.00)
KQT (Euclidean)	4.53% (1.63%)	1.00 (0.00)
KQT (Mahalanobis)	4.50% (1.58%)	1.00 (0.00)
KQT (Weighted Maha.)	4.61% (1.67%)	1.00 (0.00)

In most experiments, the empirical FPR achieved by KQT is

close to the target $\alpha = 5\%$ for all the considered kernel functions. However, the KQT with the Weighted Mahalanobis distance struggles to control the FPR when d is large. This is a known limitation of high-dimensional settings, where the estimated GMM might be poorly conditioned when TR is not sufficiently large. Therefore, when the GMM fit from TR yields Gaussians having covariances with large condition numbers, it is convenient to use KQT with the Mahalanobis distance. This latter, in fact, can control the FPR and usually achieves comparable detection performance with the Weighted Mahalanobis distance.

To further investigate how the dimension d influences the detection performance of KQT, we run another experiment where we train KQT on synthetic data with $d \in \{4, 8, 16, 32, 64, 128\}$ and with $N \in \{4096, 16384\}$. The results reported in the supplementary material prove that the FPR control worsens when d increases, but also that using large training sets heavily mitigates this problem. Moreover, this issue can be avoided by employing dimensionality reduction techniques on high-dimensional data. To this purpose, we run an experiment on the Swarm dataset ($d = 2400$) preprocessed by a PCA transformation that retains only the first 32 principal components. Table 3 shows that thanks to this preprocessing, KQT can seamlessly operate without incurring the loss of FPR control observed on some UCI datasets. The large AUC achieved by all the methods on this dataset proves that classes are very far apart, and we argue that a similar situation would have happened if we had artificially added a change (which would not correspond to a real-world problem) to offset each component. In fact, even a small perturbation would result in a very apparent change. This is probably the reason why change detection benchmarks are of lower dimensions (e.g., INSECTS d=33).

As for the other methods, QT and EIKM accurately control the FPR. In contrast, SPLL and PCA-SPLL mostly exceed the target, and we speculate that the distribution of the SPLL test statistic does not satisfy the t-test assumptions. Finally, Density Tree largely overshoots the target FPR, being unable to learn a detection threshold from bootstrapping.

KQT with the Weighted Mahalanobis and the Mahalanobis distance represent the best and second-best method in terms of AUC, mostly outperforming the alternatives in most settings. In particular, the advantage over the third-ranked method (QT w/ PCA) is considerable, and the p-values of the Nemenyi post-hoc test show that the advantage of KQT is also statistically significant. However, when using the Euclidean distance on real-world data, the performance of KQT worsens because its anisotropic bins cannot model the intricate data distributions of real data. As for the other methods, SPLL performs well over the synthetic datasets but fails over the INSECTS and UCI, achieving significantly low performance on the latter. Instead, PCA-SPLL mainly improves the performance of SPLL, even though it cannot compete with the top-performing methods. Finally, Density Tree mostly achieves low detection performance, except on the INSECTS dataset, where it surpasses the other methods.

Our experiments show that preprocessing by PCA is in general beneficial for QT, as the average rank of QT w/PCA is lower than QT w/o PCA. However, in some settings, QT w/o PCA performs better. In contrast, KQT achieves the best AUC independently of the PCA preprocessing, thanks to the invariance to roto-translation proved in Section 5.3, and Density Tree is also not affected by the PCA.

The supplementary material reports the detection results in the low-ratio setting ($K = 32$, $\nu = 64$). Overall, the results conform to those in the high-ratio setting and confirm that a larger expected number of points per bin improves the detection performance for all the methods. For the same reason, QT and KQT achieve lower FPR in the low-ratio setting than in the high-ratio, since the Pearson statistic assumes fewer distinct values. Thus, while still controlling the FPR, this results in a slightly lower percentage of false alarms. The KQT with the Weighted Mahalanobis distance achieves the highest AUC in the low-ratio setting, with a statistically significant advantage over the competitors.

We conclude by remarking that data in the INSECTS and UCI datasets are not drawn from multivariate Gaussian distributions, as suggested by the low performance achieved by SPLL, which is based on a GMM. To confirm this, we run the Shapiro-Wilk normality test on the marginals of our real-world data, showing that these are not univariate Gaussians. In the supplementary material, we report the p-values of these tests, which are in the range of 10^{-20}.

7. Conclusions

In this paper we presented KQT, a non-parametric multivariate change detection method for batch-wise monitoring. KQT constructs a space partitioning via kernel functions, resulting in bins which are compact and lead to superior detection power. We compare our method to several state-of-the-art approaches for CD, achieving the best results in terms of detection power and false positives control. Future work includes integrating KQT in a sequential monitoring scheme, where data are processed in a continuous stream.

8. Acknowledgments

This paper is supported by the PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

References

Alippi, C., Boracchi, G., and Carrera, D. Ccm: Controlling the change magnitude in high dimensional data. In Angelov, P., Manolopoulos, Y., Iliadis, L., Roy, A., and Vellasco, M. (eds.), *Advances in Big Data*, pp. 216–225, Cham, 2017. Springer International Publishing. ISBN 978-3-319-47898-2.

Basseville, M., Nikiforov, I. V., et al. *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs, 1993.

Boracchi, G., Carrera, D., Cervellera, C., and Maccio, D. Quanttree: Histograms for change detection in multivariate data streams. In *International Conference on Machine Learning*, pp. 639–648. PMLR, 2018.

Criminisi, A., Shotton, J., and Konukoglu, E. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and trends® in computer graphics and vision*, 7(2–3):81–227, 2012.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G. Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE transactions on neural networks and learning systems*, 29(8):3784–3797, 2017.

Demšar, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Frittoli, L., Matteo, B., Silvia, M., Carrera, D., Beatrice, R., Fragneto, P., Ruggero, S., and Boracchi, G. Strengthening sequential side-channel attacks through change detection. In *Conference on Cryptographic Hardware and Embedded Systems (CHES)*, volume 3, pp. 1–21, 2020.

Frittoli, L., Carrera, D., and Boracchi, G. Nonparametric and online change detection in multivariate datastreams using quanttree. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

Gini, C. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.[Fasc. I.]*. Tipogr. di P. Cuppini, 1912.

Hawkins, D. M., Qiu, P., and Kang, C. W. The changepoint model for statistical process control. *Journal of quality technology*, 35(4):355, 2003.

Krempl, G. The algorithm apt to classify in concurrence of latency and drift. In *Proceedings of the Intelligent Data Analysis (IDA)*, pp. 222–233, 2011.

Kuncheva, L. I. Change detection in streaming multivariate data using likelihood detectors. *IEEE transactions on knowledge and data engineering*, 25(5):1175–1180, 2011.

Kuncheva, L. I. and Faithfull, W. J. Pca feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems*, 25(1):69–80, 2013.

Lehmann, E. L., Romano, J. P., and Casella, G. *Testing statistical hypotheses*, volume 3. Springer, 2005.

Liu, A., Lu, J., and Zhang, G. Concept drift detection via equal intensity k-means space partitioning. *IEEE transactions on cybernetics*, 51(6):3198–3211, 2020.

Mitchell, T. M. *Machine learning*, volume 1. McGraw-hill New York, 1997.

Nemenyi, P. B. *Distribution-free multiple comparisons.* Princeton University, 1963.

Ross, G. J. and Adams, N. M. Two nonparametric control charts for detecting arbitrary distribution changes. *Journal of Quality Technology*, 44(2):102, 2012.

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

Saatçi, Y., Turner, R. D., and Rasmussen, C. E. Gaussian process change point models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 927–934, 2010.

Souza, V. M., dos Reis, D. M., Maletzke, A. G., and Batista, G. E. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858, 2020.

Tartakovsky, A. G., Rozovskii, B. L., Blazek, R. B., and Kim, H. A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. *IEEE TSP*, 54(9):3372–3382, 2006.

Thudumu, S., Branch, P., Jin, J., and Singh, J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7:1–30, 2020.

Tipping, M. E. Deriving cluster analytic distance functions from gaussian mixture models. 1999.

Welch, B. L. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

Kernel QuantTree (Supplementary Material)

1 Introduction

This document provides additional material omitted from the main article due to space limitations. Section 2 reports the proofs of the theoretical results supporting the Kernel QuantTree (KQT) algorithm, namely, the independence of the test statistic from the stationary distribution (Theorem 1) and the roto-translational invariance of KQT (Theorem 2). Then, Section 3 illustrates additional experimental settings that complete the empirical analysis of the KQT algorithm. In particular, we test the normality of the employed real-world datasets (Section 3.1), investigate the performance on high-dimensional datasets (Section 3.2), compare the performance of the proposed centroid selection strategies (Section 3.3), and report the complete results of the experiments from the main article (Section 3.4).

2 Theoretical Results

In this section, we report the proofs of the theorems introduced in Section 5 of the main article. To make this section self-contained and improve the overall readability, we recall some definitions that were already introduced in the article.

2.1 Controlling the False Alarm Rate

The Generalized QuantTree (GQT) histogram $h = \{(S_k, \widehat{\pi}_k)\}$ partitions the input space $\mathbb{R}^d$ such that the probability $\widehat{\pi}_k$ 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

$$\mathcal{T}_h(W) > \tau. \tag{1}$$

A peculiarity of GQT is that each bin S_K is defined as a subset of the sublevel set of a *measurable* kernel function $f_k : \mathbb{R}^d \to \mathbb{R}$. In this section, we prove Theorem 5.1 of the main article, which we recall here:

Theorem 1. *Let $h = \{(S_k, \widehat{\pi}_k)\}_{k=1}^K$ 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 $y_1, \dots, y_K$ 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 1 implies that the distribution of $\mathcal{T}_h$ 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 False Positive Rate (FPR) achieved by GQT is controlled. The threshold computation strategy is presented in Section 5.2 of the main article. Theorem 1 is a generalization of Theorem 1 from [Boracchi et al., 2018] and its proof follows the same structure based on three propositions. Here, we prove the first of these propositions:

Proposition 1. *Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ 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 $Z = f(X)$. We denote with $z_{(1)} \leq z_{(2)} \leq \cdots \leq z_{(M)}$ the sorted images of $\{\mathbf{x}_j\}$ through f. For any $L \in \{1, 2, \ldots, M\}$, we define the sublevel sets*

$$Q_{f,L} := \{\mathbf{x} \in \mathcal{D} : f(\mathbf{x}) \leq z_{(L)}\}. \tag{2}$$

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

Proof. We prove the proposition by showing that p is an order statistic of the uniform distribution, which in turn follows a Beta distribution [Lehmann et al., 2005]. Since f is a measurable function for the considered probability space and X is a continuous random variable (r.v.) in $\mathbb{R}^d$, by the properties of continuous r.v. [Papoulis and Pillai, 2002], we have that $Z = f(X)$ is also a continuous r.v. in $\mathbb{R}$. Then, we define $U = F_Z(Z)$, where F_Z is the cdf of Z. Since F_Z is monotonically non-decreasing, we can also define the inverse cdf as:

$$F_Z^{-1}(t) = \inf\{z \in \mathbb{R} \mid F_Z(z) \geq t\}. \tag{3}$$

Then, we have that

$$\begin{aligned} F_U(u) = P_U(U \leq u) = P_Z(F_Z(Z) \leq u) = \\ = P_Z(Z \leq F_Z^{-1}(u)) = F_Z(F_Z^{-1}(u)) = u, \end{aligned} \tag{4}$$

hence U is a uniform random variable, since its cumulative density function is the identity. Recall that we assumed that $\mathbf{X}$ is defined over $\mathcal{D}$, i.e., $P_{\mathbf{X}}(\mathbb{R} \setminus \mathcal{D}) = 0$. Then, exploiting (4), we can express p as follows:

$$\begin{aligned} p = P_X(Q_{f,L}) = P_X(\mathbf{x} \in \mathcal{D} \mid f(\mathbf{x}) \leq z_{(L)}) = \\ = P_Z(z \in \mathbb{R} \mid z \leq z_{(L)}) = \\ = P_U(u \in [0,1] \mid u \leq u_{(L)}) = u_{(L)}, \end{aligned} \tag{5}$$

where we define $u_{(L)} = F_Z(z_{(L)})$. From (5), we have that p is the L-th order statistic of M samplings of the uniform distribution, and its distribution is Beta(L, $M-L+1$) [Balakrishnan and Rao, 1998]. □

We refer the reader to [Boracchi et al., 2018] for a thorough description of the derivation of the proof of Theorem 1 from Proposition 1.

2.2 Centroid Selection and Invariance to Roto-Translation

Kernel QuantTree (KQT) defines a partition of the input space by iteratively splitting it in bins S_k that match a target probability, as shown in Figure 2 of the main article. The KQT bins are defined as subsets of sublevel sets of the adopted measurable kernel functions f_k. We report here the formal definition of the KQT histogram bins:

$$\begin{cases} S_1 = \{\mathbf{x} \in \mathbb{R}^d \mid f_1(\mathbf{x}) \leq q_1\} \\ S_k = \{\mathbf{x} \in \bigcap_{j<k} \overline{S_j} \mid f_k(\mathbf{x}) \leq q_k\} \text{ for } k < K \\ S_K = \mathbb{R}^d \setminus \bigcup_{j<K} S_j \end{cases}, \tag{6}$$

where $\overline{S_j}$ denotes the complement of S_j in $\mathbb{R}^d$, and q_k is the split value computed as a quantile of the training samples projected via f_k.

Section 4.2 of the main article illustrates the kernel functions $f_k : \mathbb{R}^d \to \mathbb{R}$ adopted by KQT, which are defined as distances from a selected *centroid* $\mathbf{c}_k \in \text{TR}$:

$$f_k(\mathbf{x}) = (\mathbf{x} - \mathbf{c}_k)^\intercal A (\mathbf{x} - \mathbf{c}_k), \tag{7}$$

where $A \in \mathbb{R}^{d \times d}$ is the *kernel matrix*, which determines the employed distance. In our experiments, we construct KQT using the Euclidean, the Mahalanobis, and the Weighted Mahalanobis [Tipping, 1999] distances. The corresponding kernel matrices are $A = \mathbb{I}_d$ for the Euclidean distance, $A = \mathrm{cov}[\mathrm{TR}]^{-1}$ for the Mahalanobis distance and

$$A = \frac{\sum_{m=1}^{M} \rho_m \cdot i_m(\mathbf{x}, \mathbf{c}) \cdot C_m^{-1}}{\sum_{m=1}^{M} \rho_m \cdot i_m(\mathbf{x}, \mathbf{c})} \tag{8}$$

for the Weighted Mahalanobis distance, where μ_m, C_m, and ρ_m denote the mean, covariance matrix, and mixing probability of the m-th Gaussian component of a GMM fitted to TR, and the term $i_m(\mathbf{x}, \mathbf{c})$ approximates the integral over the path from $\mathbf{x}$ to $\mathbf{c}$ with respect to the measure induced by the Gaussian Mixture Model (GMM). We refer the reader to [Tipping, 1999] for an explanation of the rationale behind this distance.

2.2.1 Selecting Centroids by Maximizing the Information Gain

In Section 4.3 of the paper, we propose a centroid selection strategy that consists in maximizing the information gain introduced by the split that divides $\mathcal{X}_{k-1}$ in $\mathcal{X}_k$ and $\overline{\mathcal{X}_k} = \mathcal{X}_{k-1} \setminus \mathcal{X}_k$, namely:

$$\mathbf{c}_k = \underset{\mathbf{c} \in \mathcal{X}_{k-1}}{\operatorname{argmax}} I[\mathbf{c}] = \underset{\mathbf{c} \in \mathcal{X}_{k-1}}{\operatorname{argmax}} \left\{ H(\mathcal{X}_{k-1}) - \frac{\left|\overline{\mathcal{X}_k}\right| H(\overline{\mathcal{X}_k}) + |\mathcal{X}_k| H(\mathcal{X}_k)}{|\mathcal{X}_{k-1}|} \right\}, \tag{9}$$

where $H(B)$ is the entropy of a set of points $B \subset \mathbb{R}^d$, which we compute by its Gaussian approximation, that is

$$H(B) = (1/2) \log \Big((2\pi e)^d \det(\mathrm{cov}[B]) \Big), \tag{10}$$

where e is Euler's number. This approximation is only used to ease the computation of $H(B)$ for centroid selection purposes, and does not influence the non-parametric nature of KQT. The expression in (10) can be reformulated:

$$H(B) = \frac{d}{2} \left(\log(2\pi) + 1 \right) + \widetilde{H}(B) \tag{11}$$

where $\widetilde{H}(B) = \mathrm{logdet}\left(\mathrm{cov}[B]\right)$. This gives rise to an optimization problem equivalent to (9), where the centroid is selected by

$$\mathbf{c}_k = \underset{\mathbf{c} \in \mathcal{X}_{k-1}}{\operatorname{argmin}} \Big\{ \widetilde{H}(\overline{\mathcal{X}}_k) + \beta \widetilde{H}(\mathcal{X}_k) \Big\}, \tag{12}$$

where β is a constant that can be derived by (9) through algebraic manipulation. Solving this minimization problem is computationally less demanding than the original maximization, thus lowering the computational burden of such centroid selection strategy.

2.2.2 Invariance to roto-translations

In Section 5.3 of the main article, we state that KQT is invariant under roto-translations when the employed kernel function is either the Euclidean, Mahalanobis or Weighted Mahalanobis distance. Here, we prove it together with an intermediate result. In the following, we define a roto-translation $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ as

$$\Phi(\mathbf{x}) = R(\mathbf{x} - \mu), \tag{13}$$

where $R \in SO(d)$ is the rotation matrix and $\mu \in \mathbb{R}^d$ is the shift vector. Moreover, we denote as $\Phi(B) = \{\Phi(\mathbf{x}) \mid \mathbf{x} \in B\}$ the image of a set $B \subset \mathbb{R}^d$. From basic calculus, it is easy to show that the covariance of a set $B \subset \mathbb{R}^d$ after roto-translation Φ factorizes as

$$\mathrm{cov}[\Phi(B)] = R\,\mathrm{cov}[B] R^{\intercal}. \tag{14}$$

Moreover, in our discussion, we will denote as $D : \mathbb{R}^d \to \mathbb{R}^d$ the distance employed by KQT when no preprocessing is employed, while we denote as D' the same distance when data are preprocessed by a roto-translation Φ. D and D' coincide when we employ the Euclidean distance, where A is simply the indentity matrix. However, the kernel matrices for the Mahalanobis and Weighted Mahalanobis distances depend on the training set TR, thus change when we transform it to $\mathrm{TR}' = \Phi(\mathrm{TR})$. Nevertheless, all the considered distances are invariant under roto-translation, namely it holds that

$$D(\mathbf{x}, \mathbf{y}) = D'\left(\Phi(\mathbf{x}), \Phi(\mathbf{y})\right) \tag{15}$$

for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. The identity in (15) can be derived from algebraic manipulation of the definition of the adopted distances and considering (14).

Theorem 5.5 of the main article, which we report here, states that the histograms $h = \{(S_k, \widehat{\pi}_k)\}$ and $h' = \{(S'_k, \widehat{\pi}'_k)\}$, respectively constructed by KQT with and without preprocessing TR by Φ, are equivalent.

Theorem 2. *Let $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ be a roto-translation. Let $h = \{(S_k, \widehat{\pi}_k)\}$ and $h' = \{(S'_k, \widehat{\pi}'_k)\}$ be the KQT histograms constructed from the training sets $\mathrm{TR} \subset \mathbb{R}^d$ and $\mathrm{TR}' = \Phi(\mathrm{TR})$, where the employed kernel function is either the Euclidean, Mahalanobis or Weighted Mahalanobis distance. Then, we have that $S'_k = \Phi(S_k)$ and $\widehat{\pi}'_k = \widehat{\pi}_k$ for $k = 1 \dots, K$. In particular, for any batch W and $W' = \Phi(W)$, we have that $\mathcal{T}_h(W) = \mathcal{T}_{h'}(W')$.*

Theorem 2 proves that, for specific choices of kernel functions, the value of the test statistic computed over a batch W of data does not change if we employ a roto-translation-based preprocessing. As such, KQT does not require preprocessing by PCA, which is sometimes necessary for QT to achieve good detection performance. To prove the theorem, we first prove an intermediate result regarding the centroid selection:

Lemma 1 (Information Gain)**.** *Let $\mathcal{X}_{k-1}$ and $\mathcal{X}'_{k-1} = \Phi(\mathcal{X}_{k-1})$ be the set of points used to construct the KQT histogram bins S_k and S'_k, respectively. Then, the centroid selection by maximizing the information gain as in* (9) *results in centroids $\mathbf{c}_k$ and $\mathbf{c}'_k = \Phi(\mathbf{c}_k)$.*

Proof. As showed in Section 2.2.1, maximizing (9) is equivalent to minimizing (12). Let $\mathbf{c} \in \mathcal{X}_{k-1}$ be an available training sample, then there exists $\mathbf{c}' = \Phi(\mathbf{c}) \in \mathcal{X}'_{k-1}$. If we assume that $\mathcal{X}'_k$ is the set of training samples falling in S'_k when we use $\mathbf{c}$ as a centroid, we have that

$$\begin{aligned}
\mathcal{X}'_k &= \{\mathbf{x}' \in \mathcal{X}'_{k-1} \mid D'(\mathbf{x}', \mathbf{c}') \leq q'_k\} = \\
&= \{\Phi(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}_{k-1}, D'(\Phi(\mathbf{x}), \Phi(\mathbf{c})) \leq q'_k\} = \\
&= \{\Phi(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}_{k-1}, D(\mathbf{x}, \mathbf{c}) \leq q_k\} = \\
&= \Phi\left(\{\mathbf{x} \in \mathcal{X}_{k-1} \mid D(\mathbf{x}, \mathbf{c}) \leq q_k\}\right) = \Phi(\mathcal{X}_k),
\end{aligned} \tag{16}$$

where we used (15) to substitute q'_k with q_k. Analogously, we have that $\overline{\mathcal{X}}'_k = \Phi(\overline{\mathcal{X}}_k)$. Then, from (12), we have that

$$\begin{aligned}
\widetilde{H}(\mathcal{X}'_k) &= \operatorname{logdet}\left(\operatorname{cov}[\mathcal{X}'_k]\right) = \\
&= \operatorname{logdet}\left(R \operatorname{cov}[\mathcal{X}_k] R^{\intercal}\right) = \\
&= \operatorname{logdet}\left(\operatorname{cov}[\mathcal{X}_k]\right) + 2 \operatorname{logdet}\left(R\right) = \widetilde{H}(\mathcal{X}_k) + \gamma,
\end{aligned} \tag{17}$$

where γ is a constant which depends only on R. In (17), we used the factorization (14) and the fact

that R is orthogonal. The same relation holds for $\overline{\mathcal{X}}'_k$, and we can finally prove that

$$\begin{aligned}\mathbf{c}'_k &= \underset{\mathbf{c}\in\mathcal{X}'_{k-1}}{\operatorname{argmin}} \left\{ \widetilde{H}(\overline{\mathcal{X}}'_k) + \beta \widetilde{H}(\mathcal{X}'_k) \right\} = \\ &= \Phi\left(\underset{\mathbf{c}\in\mathcal{X}_{k-1}}{\operatorname{argmin}} \left\{ \widetilde{H}(\overline{\mathcal{X}}_k) + \beta \widetilde{H}(\mathcal{X}_k) + \gamma(1+\beta) \right\} \right) = \\ &= \Phi\left(\underset{\mathbf{c}\in\mathcal{X}_{k-1}}{\operatorname{argmin}} \left\{ \widetilde{H}(\overline{\mathcal{X}}_k) + \beta \widetilde{H}(\mathcal{X}_k) \right\} \right) = \Phi(\mathbf{c}_k).\end{aligned} \tag{18}$$

□

Lemma 2 (Gini Index)**.** *Let $\mathcal{X}_{k-1}$ and $\mathcal{X}'_{k-1} = \Phi(\mathcal{X}_{k-1})$ be the set of points used to construct the KQT histogram bins S_k and S'_k, respectively. Then, the centroid selection by minimizing the Gini index in results in centroids $\mathbf{c}_k$ and $\mathbf{c}'_k = \Phi(\mathbf{c}_k)$.*

Proof. It can be shown by simple algebraic manipulation of the definition of Gini index. □

Lemma 1 and Lemma 2 ensure that the construction of the histograms h and h' will maintain the correspondance through Φ of all their elements, including the selected centroids. We can now prove Theorem 2.

Proof of Theorem 2. Here, we show by induction that every bin S'_k of h' is the result of the roto-translation of the corresponding bin S_k of h. First, we have that $\mathcal{X}'_0 = \mathrm{TR}' = \Phi(\mathrm{TR}) = \Phi(\mathcal{X}_0)$ by definition. Then, for $k = 1$, Lemma 1 and 2 state that $\mathbf{c}'_1 = \Phi(\mathbf{c}_1)$. Moreover,

$$\begin{aligned}S'_1 &= \left\{ \mathbf{x}' \in \mathbb{R}^d \mid D'(\mathbf{x}', \mathbf{c}'_1) \le q'_1 \right\} = \\ &= \left\{ \Phi(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^d, D'(\Phi(\mathbf{x}), \mathbf{c}'_1) \le q'_1 \right\} = \\ &= \left\{ \Phi(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^d, D(\mathbf{x}, \mathbf{c}_1) \le q_1 \right\} = \Phi(S_1).\end{aligned} \tag{19}$$

In the same manner, we prove that

$$\begin{aligned}X'_1 &= \{ \mathbf{x}' \in \mathcal{X}'_0 \mid D'(\mathbf{x}', \mathbf{c}'_1) > q'_1 \} = \\ &= \{ \Phi(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}_0, D(\mathbf{x}, \mathbf{c}_1) > q'_1 \} = \Phi(\mathcal{X}_1),\end{aligned} \tag{20}$$

and $\overline{\mathcal{X}}'_1 = \Phi(\overline{\mathcal{X}}_1)$.

Now, suppose that $\forall j < k$ we have that $\mathbf{c}'_j = \Phi(\mathbf{c}_j)$, $S'_j = \Phi(S_j)$ and $\mathcal{X}'_j = \Phi(\mathcal{X}_j)$. Then, we have that $\mathbf{x} \in \bigcap_{j<k} \overline{S_j} \iff \mathbf{x}' = \Phi(\mathbf{x}) \in \bigcap_{j<k} \overline{S'_j}$, and, with the same derivation as in the case $k = 1$,

$$\begin{aligned}S'_k &= \left\{ \mathbf{x}' \in \bigcap_{j<k} \overline{S'_j} \mid D'(\mathbf{x}', \mathbf{c}'_k) \le q_k \right\} = \\ &= \Phi\Big(\Big\{ \mathbf{x} \in \bigcap_{j<k} \overline{S_j} \mid D(\mathbf{x}, \mathbf{c}_k) \le q_k) \Big\} \Big) = \Phi(S_k).\end{aligned} \tag{21}$$

and also $\mathcal{X}'_k = \Phi(\mathcal{X}_k)$. In conclusion, we proved that $S'_k = \Phi(S_k)$ for $\forall k = 1, \ldots, K$. In particular, we conclude that

$$\begin{aligned}\mathbf{x} \in S_k &\iff D(\mathbf{x}, \mathbf{c}_j) > q_j \ \forall j < k \ \wedge \ D(\mathbf{x}, \mathbf{c}_k) \le q_k \iff \\ &\iff D'(\Phi(\mathbf{x}), \mathbf{c}'_j) > q_j \ \forall j < k \ \wedge \ D'(\Phi(\mathbf{x}), \mathbf{c}'_k) \le q_k \iff \\ &\iff \Phi(\mathbf{x}) \in S'_k,\end{aligned} \tag{22}$$

and, consequently, the number of samples from any batch $W \subset \mathbb{R}^d$ falling in the S_k is the same as the number of samples of $W' = \Phi(W)$ falling in S'_k. Then, we have that $\widehat{\pi}_k = \widehat{\pi}'_k$ and $\mathcal{T}_h(W) = \mathcal{T}_{h'}(W')$. □

3 More Experiments and Discussion

This section extends the experimental evaluation of KQT from Section 6 of the main article to corroborate the findings discussed there. First, we investigate the real-world datasets employed in our experiments, proving that these do not follow a Gaussian distribution. Then, we perform additional experiments on high-dimensional data to investigate the control of the FPR in this challenging scenario. Finally, we extend the results from the main article by comparing the proposed centroid selection strategies and reporting the complete results for both the low- and high-ratio settings.

3.1 Remarks about the real-world datasets

In Section 6.1 of the main article, we introduce the real-world datasets that are used in our experiments. The INSECTS dataset [Souza et al., 2020] is a benchmark for concept-drift detection algorithms and comprises data describing the wing-beat frequency of six species of insects at different temperatures. The other datasets are from the UCI Machine Learning Repository [Dua and Graff, 2017] and from [Dal Pozzolo et al., 2017], and comprise data following a unique distribution, thus require the introduction of artificial distribution changes for our experiments. We standardize these datasets and add a negligible amount of noise $\eta \sim N(0, \sigma)$ to each component to prevent the many repeated values from harming the histogram construction. Table 1 lists all the datasets and reports their dimension d and the level σ of noise applied to their components.

Since data in the synthetic settings are drawn from Gaussian distributions, one could argue that KQT provided with the Mahalanobis or Weighted Mahalanobis kernels have an advantage over the alternatives. However, this is not true for the real-world datasets considered in our experiments, which are far from Gaussian. This claim is empirically supported by the low detection performance of SPLL, which is itself based on a GMM. To confirm this intuition, we also run the Shapiro-Wilk normality test [Shapiro and Wilk, 1965], an Hypothesis Test used to determine whether a population $\{x_i\}_{i=1}^n \subset \mathbb{R}$ is drawn from a univariate Gaussian distribution. If the p-values associated to the HT is lower than 0.05, than we can conclude that the population is not normally distributed. Since the marginals of a multivariate Gaussian distribution are univariate Gaussian distributions, we show that the real-world datasets introduced in Section 6.1 of the main article are not drawn from multivariate Gaussians by showing that their covariates are not. For this purpose, we extract a subset of $n = 4096$ samples from the real-world datasets and perform the Shapiro-Wilk test on each of their covariates. Table 1 reports the p-values yielded by the test, averaged over the covariates and over 250 iteration of the test performed over different subsets. The p-values obtained in these tests are in the range of 10^{-20}, thus confirming that the real-world datasets employed in our experiments are not drawn from multivariate Gaussian distributions.

Table 1: List of the real-world datasets employed in the experiments. For each dataset, we report the dimension d, the noise level σ and the average p-value of the Shapiro-Wilk test computed on the marginals.

Dataset	Name	d	σ	p-value	Reference
El Nino Southern Oscillation	nino	5	10^{-3}	3.3×10^{-3}	[Dua and Graff, 2017]
Physicochemical Properties of PTS	protein	9	–	7.5×10^{-8}	[Dua and Graff, 2017]
ForestCovertype I	spruce	10	10^{-1}	2.5×10^{-9}	[Dua and Graff, 2017]
ForestCovertype II	lodgepole	10	10^{-1}	1.8×10^{-8}	[Dua and Graff, 2017]
Credit Card Fraud Detection	credit	28	10^{-3}	9.9×10^{-8}	[Dal Pozzolo et al., 2017]
Insects' Flying Behavior	INSECTS	33	–	$< 10^{-16}$	[Souza et al., 2020]
Sensorless Drive Diagnosis	sensorless	48	10^{-3}	4.9×10^{-8}	[Dua and Graff, 2017]
MiniBooNE Particle Identification	particle	50	10^{-3}	8.5×10^{-3}	[Dua and Graff, 2017]
UNSW Swarm Behavior	swarm	2400	–	1.8×10^{-9}	[Dua and Graff, 2017]

3.2 Curse of dimensionality

In this section, we investigate the ability of KQT to control the FPR as the data dimension d increases. Our experiments (Section 6.4 of the main article) have shown that the Kernel QuantTree with the Weighted Mahalanobis distance deviates from the desired FPR when the data dimension grows. As discussed in the article, this deviation is due to the challenge of fitting a Gaussian Mixture Model (GMM) to high-dimensional data. To analyze the impact of the dimensionality on Kernel QuantTree, we perform experiments in three synthetic settings with $d \in \{4, 8, 16, 32, 64, 128\}$. In these settings, denoted as *unimodal*, *bimodal*, and *trimodal*, the stationary distribution ϕ_0 is defined as a GMM with 1, 2, and 3 Gaussian components, respectively. Then, we use the CCM framework [Alippi et al., 2017] to generate a post-change distribution by applying a roto-translation to each Gaussian component of ϕ_0 such that the Kullback-Leibler distance between these and the three resulting post-change components is fixed to 1. We perform each experiment twice, one with $N = 4096$ training samples and the other with $N = 16384$, to show that when a large training set is available the limitation of the KQT with Weighted Mahalanobis is avoidable.

Table 2 reports the FPR achieved by KQT adopting different distances in the unimodal, bimodal and trimodal settings for all the considered dimensions d and training set sizes N. In all experiments we construct a KQT histogram with $K = 16$ bins, we set the detection thresholds to yield an FPR $\alpha = 5\%$, and we test KQT on 5000 stationary batches with $\nu = 128$ samples. In the experiment with $N = 4096$ (left columns), we notice that when d increases, the FPR achieved by KQT when using the Mahalanobis and Weighted Mahalanobis distances deviates further from the target value. In contrast, when we train KQT on $N = 16384$ samples (right columns), the deviation from the target FPR is significantly reduced. To further corroborate our hypothesis that the issue is in the GMM fitting, we compute the average condition number of the covariance matrices of the GMM components yielded when using KQT with the Weighted Mahalanobis distance. These results show that, in the high-dimensional datasets, using a larger training set yields covariance matrices with smaller condition numbers.

3.3 Comparing the centroid selection strategies

In the main article, we propose two strategies for the centroid selection, namely, maximizing the information gain introduced by splitting $\mathcal{X}_{k-1}$ in $\mathcal{X}_k$ and $\overline{\mathcal{X}}_k$ and minimizing the Gini index of the distances between the centroid and the training samples in $\mathcal{X}_{k-1}$. In Table 1 of the article, we report the average FPR and AUC achieved by KQT using the maximization of the information gain as a centroid selection strategy. Here, Table 3 reports the results achieved by KQT with the Euclidean, Mahalanobis, and Weighted Mahalanobis distance for both strategies and proves that their performance is comparable in every experimental setting.

3.4 Complete experimental results

In this section, we report the complete results of the high- and low-ratio experiments presented in Section 6 of the main article. For each result, we include the corresponding confidence interval.

Table 5 reports the FPR and AUC achieved by the methods presented in Section 6.3 of the main article in the high-ratio setting, namely when $\nu = 128$ and $K = 16$. As already discussed in the paper, QuantTree, Kernel QuantTree and EIKM achieve an empirical FPR close to the target $\alpha = 5\%$ in most experiments. In contrast, SPLL and PCA-SPLL mostly exceed the target and Density Tree largely overshoots it. However, the KQT with the Weighted Mahalanobis distance does not control the FPR accurately when d increases, and we speculate that this is due to the GMM underlying the definition of distance. This known limitation is discussed in Section 6.4 of the main article and investigated in Section 3.2 of this document.

As for the AUC, KQT with the Weighted Mahalanobis distances outperforms the alternatives in most settings. At the bottom of Table 5, we report the ranking of each method computed from the

Table 2: Comparison between the FPR achieved by KQT using the Euclidean, Mahalanobis, and Weighted Mahalanobis distances in the synthetic settings for various dimensions d and training set sizes N. In parenthesis, the average condition numbers of the covariance matrices of the GMM used by KQT with the Weighted Mahalanobis distance. The underlined values indicate an FPR above the target of 5%.

	KQT(Euclidean)		KQT(Mahalanobis)		KQT(Weighted Mahalanobis)		
	N=4096	N=16384	N=4096	N=16384	N=4096	N=16384	
$d = 4$	4.88%	-	4.77%	4.84%	4.79% (20.1)	4.85% (16.3)	unimodal
$d = 8$	4.83%	-	4.81%	4.86%	4.71% (32.1)	4.83% (37.7)	
$d = 16$	4.81%	-	4.81%	4.89%	4.88% (99.5)	4.79% (67.7)	
$d = 32$	4.84%	-	4.95%	4.88%	4.99% (150.6)	4.81% (123.6)	
$d = 64$	4.84%	-	5.80%	4.95%	5.81% (315.2)	4.87% (223.8)	
$d = 128$	4.91%	-	16.52%	5.31%	77.74% (344.0)	5.45% (307.0)	
$d = 4$	4.83%	-	4.76%	4.88%	4.77% (12.9)	4.89% (13.7)	bimodal
$d = 8$	4.88%	-	4.86%	4.87%	4.79% (40.0)	4.84% (36.3)	
$d = 16$	4.83%	-	4.88%	4.82%	4.88% (68.3)	4.87% (90.3)	
$d = 32$	4.86%	-	4.95%	4.86%	5.36% (177.2)	4.83% (120.4)	
$d = 64$	4.89%	-	5.66%	4.86%	5.70% (253.9)	5.03% (220.3)	
$d = 128$	4.84%	-	15.44%	5.32%	76.60% (276.9)	5.46% (244.7)	
$d = 4$	4.72%	-	4.86%	4.85%	4.82% (24.9)	4.84% (16.1)	trimodal
$d = 8$	4.84%	-	4.79%	4.82%	4.83% (31.7)	4.80% (38.2)	
$d = 16$	4.85%	-	4.85%	4.80%	4.86% (66.1)	4.83% (59.9)	
$d = 32$	4.81%	-	4.91%	4.84%	5.13% (108.7)	4.86% (120.7)	
$d = 64$	4.93%	-	5.67%	4.87%	5.53% (209.4)	5.01% (176.9)	
$d = 128$	4.81%	-	15.86%	5.37%	77.49% (258.1)	5.47% (214.2)	

AUC, together with the p-value of the Nemenyi post-hoc statistic, which proves that the advantage of the best-performing method is statistically significant. In the main article, we also discuss the performance of QuantTree, which shows how the preprocessing by PCA decreases the detection performance in some cases. Remarkably, the KQT monitoring is invariant under roto-translations (see Section 5.3 of the main article) and surpasses QuantTree independently of the application of the PCA preprocessing.

Table 4 reports the FPR and AUC achieved by the considered methods in the low-ratio setting, namely when $\nu = 64$ and $K = 32$. The results of this experiment are overall in line with the high-ratio setting. However, as we speculate in Section 6.3 of the main article, histogram can better model a data distribution when the expected number of points per bin ν/K is large. This low-ratio setting confirms our speculation, as the considered methods achieve an AUC lower than in the high-ratio experiment on most datasets. However, KQT with the Weighted Mahalanobis distance still achieves the best AUC with a statistically significant advantage over the alternatives, as demonstrated by the Nemenyi post-hoc test. Moreover, the Pearson test statistic is discrete and in the low-ratio setting assumes fewer distinct values. Thus, it is more challenging to set detection thresholds and the results show that the empirical FPR of QuantTree and Kernel QuantTree is slightly lower than in the high-ratio experiment.

Table 3: Comparison between the detection performance achieved by KQT with the Euclidean, Mahalanobis and Weighted Mahalanobis distances, when selecting the centroids by maximization of the Information Gain (left) and minimization of the Gini Index (right), in the high-ratio setting ($K = 16$, $\nu = 128$ points). The table reports the achieved FPR (top) and AUC (bottom). In parenthesis the standard deviation.

	Information Gain			Gini Index		
	Euclidean	Mahalanobis	Weighted Maha.	Euclidean	Mahalanobis	Weighted Maha.
unimodal	4.86% (0.47%)	4.82% (0.45%)	4.83% (0.48%)	4.83% (0.47%)	4.81% (0.48%)	4.83% (0.49%)
bimodal	4.80% (0.46%)	4.81% (0.44%)	4.80% (0.45%)	4.81% (0.47%)	4.84% (0.46%)	4.83% (0.46%)
nino	5.00% (0.53%)	5.02% (0.53%)	5.01% (0.54%)	5.02% (0.55%)	5.06% (0.54%)	5.02% (0.54%)
protein	4.97% (0.52%)	4.98% (0.54%)	5.03% (0.55%)	4.99% (0.54%)	5.03% (0.56%)	5.06% (0.53%)
spruce	4.82% (0.47%)	4.84% (0.49%)	4.90% (0.47%)	4.84% (0.49%)	4.85% (0.48%)	4.88% (0.49%)
lodgepole	4.85% (0.49%)	4.80% (0.47%)	4.90% (0.50%)	4.84% (0.50%)	4.82% (0.49%)	4.93% (0.51%)
credit	4.89% (0.48%)	4.85% (0.46%)	5.06% (0.56%)	4.89% (0.50%)	4.90% (0.52%)	5.10% (0.61%)
insects ($1 \to 2$)	4.91% (0.50%)	4.93% (0.52%)	5.19% (0.64%)	4.90% (0.49%)	4.92% (0.54%)	5.19% (0.61%)
insects ($2 \to 3$)	4.92% (0.52%)	4.96% (0.52%)	5.25% (0.62%)	4.88% (0.50%)	4.93% (0.52%)	5.24% (0.63%)
insects ($3 \to 4$)	4.90% (0.52%)	4.88% (0.53%)	5.22% (0.64%)	4.91% (0.55%)	4.92% (0.53%)	5.24% (0.64%)
insects ($4 \to 5$)	4.91% (0.51%)	4.92% (0.54%)	5.25% (0.65%)	4.92% (0.52%)	4.95% (0.49%)	5.28% (0.66%)
insects ($5 \to 6$)	4.90% (0.56%)	4.92% (0.53%)	5.26% (0.72%)	4.90% (0.55%)	4.91% (0.57%)	5.29% (0.71%)
sensorless	4.82% (0.49%)	5.01% (0.56%)	7.42% (1.61%)	4.87% (0.48%)	4.98% (0.58%)	7.54% (1.56%)
particle	4.81% (0.46%)	4.94% (0.52%)	5.80% (1.02%)	4.84% (0.48%)	4.93% (0.52%)	5.86% (1.00%)
unimodal	0.946 (0.105)	0.993 (0.016)	0.994 (0.013)	0.946 (0.103)	0.994 (0.015)	0.994 (0.014)
bimodal	0.904 (0.118)	0.954 (0.060)	0.968 (0.042)	0.903 (0.119)	0.955 (0.056)	0.970 (0.039)
nino	0.607 (0.072)	0.904 (0.138)	0.922 (0.122)	0.609 (0.071)	0.903 (0.139)	0.922 (0.122)
protein	0.617 (0.074)	0.993 (0.035)	0.995 (0.027)	0.615 (0.074)	0.993 (0.030)	0.994 (0.030)
spruce	0.601 (0.066)	1.000 (0.000)	1.000 (0.000)	0.600 (0.068)	1.000 (0.000)	1.000 (0.000)
lodgepole	0.654 (0.099)	1.000 (0.000)	1.000 (0.000)	0.653 (0.099)	1.000 (0.000)	1.000 (0.000)
credit	0.602 (0.053)	0.780 (0.146)	1.000 (0.000)	0.605 (0.055)	0.787 (0.141)	1.000 (0.000)
insects ($1 \to 2$)	0.962 (0.035)	0.972 (0.039)	0.993 (0.019)	0.961 (0.038)	0.970 (0.037)	0.994 (0.015)
insects ($2 \to 3$)	1.000 (0.001)	1.000 (0.000)	1.000 (0.000)	1.000 (0.001)	1.000 (0.000)	1.000 (0.000)
insects ($3 \to 4$)	0.904 (0.054)	0.942 (0.049)	0.990 (0.024)	0.903 (0.058)	0.946 (0.044)	0.989 (0.025)
insects ($4 \to 5$)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
insects ($5 \to 6$)	0.985 (0.016)	0.992 (0.008)	1.000 (0.000)	0.986 (0.014)	0.991 (0.009)	1.000 (0.001)
sensorless	0.542 (0.027)	1.000 (0.000)	1.000 (0.000)	0.543 (0.026)	1.000 (0.000)	1.000 (0.000)
particle	0.555 (0.030)	0.976 (0.051)	0.985 (0.039)	0.556 (0.032)	0.977 (0.051)	0.985 (0.042)

Table 4: FPR (top) and AUC (bottom) achieved by the considered methods in the high-ratio setting, namely when histogram-based methods define $K = 32$ bins and the monitored batches contain $\nu = 64$ points. In parenthesis, the width of the 95%-confidence interval of the results.

	QT	QT (PCA)	KQT (Euclidean)	KQT (Mahalanobis)	KQT (Weighted Maha.)	ElkM	SPLL(C=3)	SPLL(C=3) (PCA)	DT	DT (PCA)
unimodal	4.29% (0.34%)	4.27% (0.32%)	4.30% (0.32%)	4.30% (0.34%)	4.28% (0.33%)	4.91% (0.44%)	6.10% (0.66%)	7.24% (1.03%)	7.08% (1.01%)	7.06% (1.00%)
bimodal	4.31% (0.32%)	4.27% (0.32%)	4.29% (0.33%)	4.32% (0.33%)	4.29% (0.33%)	4.91% (0.44%)	6.29% (0.79%)	7.57% (1.26%)	6.99% (1.00%)	6.92% (0.97%)
nino	5.01% (0.38%)	5.18% (0.38%)	5.23% (0.37%)	5.18% (0.38%)	5.21% (0.38%)	4.91% (0.48%)	7.28% (1.16%)	9.60% (1.75%)	6.77% (0.98%)	6.86% (1.09%)
protein	4.98% (0.38%)	5.13% (0.39%)	5.17% (0.39%)	5.19% (0.40%)	5.21% (0.43%)	4.92% (0.47%)	15.38% (3.01%)	10.74% (2.28%)	6.81% (0.98%)	6.73% (1.08%)
spruce	4.26% (0.33%)	4.29% (0.32%)	4.30% (0.32%)	4.29% (0.33%)	4.29% (0.34%)	4.85% (0.45%)	13.42% (3.15%)	13.64% (3.18%)	6.76% (0.99%)	6.89% (1.08%)
lodgepole	4.27% (0.32%)	4.27% (0.34%)	4.29% (0.35%)	4.30% (0.36%)	4.31% (0.34%)	4.93% (0.52%)	12.57% (3.45%)	12.73% (3.46%)	6.89% (1.00%)	6.74% (1.06%)
credit	4.34% (0.39%)	4.46% (0.46%)	4.39% (0.43%)	4.38% (0.42%)	4.47% (0.44%)	4.91% (0.61%)	11.88% (2.16%)	23.23% (3.66%)	6.74% (1.04%)	6.75% (1.02%)
insects ($1 \to 2$)	4.69% (0.47%)	4.84% (0.57%)	4.84% (0.58%)	4.84% (0.56%)	4.97% (0.59%)	4.91% (0.45%)	6.73% (1.63%)	8.05% (1.95%)	6.86% (1.04%)	6.77% (1.10%)
insects ($2 \to 3$)	4.73% (0.50%)	4.87% (0.59%)	4.85% (0.56%)	4.86% (0.60%)	5.04% (0.61%)	4.90% (0.45%)	6.67% (1.51%)	8.02% (1.77%)	6.83% (1.03%)	6.77% (1.04%)
insects ($3 \to 4$)	4.72% (0.49%)	4.84% (0.56%)	4.85% (0.58%)	4.84% (0.58%)	5.02% (0.60%)	4.93% (0.48%)	7.29% (1.75%)	8.71% (2.03%)	6.80% (1.07%)	6.75% (1.03%)
insects ($4 \to 5$)	4.69% (0.50%)	4.84% (0.58%)	4.82% (0.57%)	4.83% (0.60%)	4.99% (0.64%)	4.90% (0.44%)	6.56% (1.55%)	7.89% (1.87%)	6.83% (1.03%)	6.71% (1.03%)
insects ($5 \to 6$)	4.77% (0.53%)	4.90% (0.56%)	4.90% (0.56%)	4.89% (0.58%)	5.07% (0.60%)	4.86% (0.43%)	7.18% (1.72%)	8.09% (1.89%)	6.77% (1.04%)	6.76% (1.00%)
sensorless	4.29% (0.35%)	4.43% (0.39%)	4.30% (0.34%)	4.35% (0.36%)	5.32% (0.65%)	4.94% (0.49%)	4.93% (1.19%)	4.29% (0.71%)	6.53% (1.06%)	6.67% (1.07%)
particle	4.28% (0.32%)	4.32% (0.36%)	4.30% (0.35%)	4.33% (0.35%)	4.71% (0.46%)	4.86% (0.50%)	6.92% (1.52%)	8.80% (1.96%)	6.70% (1.12%)	6.78% (1.12%)
unimodal	0.883 (0.101)	0.936 (0.059)	0.881 (0.115)	0.957 (0.031)	0.957 (0.031)	0.779 (0.157)	**0.975** (0.016)	0.972 (0.047)	0.676 (0.148)	0.712 (0.174)
bimodal	0.796 (0.109)	0.825 (0.102)	0.821 (0.112)	0.868 (0.075)	**0.885** (0.062)	0.709 (0.130)	0.859 (0.132)	0.845 (0.154)	0.636 (0.106)	0.652 (0.122)
nino	0.736 (0.143)	0.804 (0.170)	0.555 (0.042)	0.809 (0.173)	**0.830** (0.163)	0.511 (0.012)	0.739 (0.176)	0.771 (0.191)	0.630 (0.111)	0.545 (0.048)
protein	0.848 (0.104)	0.980 (0.055)	0.582 (0.048)	0.985 (0.050)	**0.991** (0.035)	0.508 (0.009)	0.906 (0.118)	0.945 (0.098)	0.638 (0.117)	0.599 (0.081)
spruce	1.000 (0.003)	1.000 (0.000)	0.590 (0.060)	**1.000** (0.000)	**1.000** (0.000)	0.504 (0.005)	1.000 (0.002)	1.000 (0.002)	1.000 (0.001)	1.000 (0.001)
lodgepole	1.000 (0.001)	1.000 (0.001)	0.639 (0.085)	**1.000** (0.000)	**1.000** (0.000)	0.506 (0.006)	1.000 (0.002)	1.000 (0.002)	1.000 (0.000)	1.000 (0.000)
credit	0.611 (0.043)	0.813 (0.144)	0.550 (0.021)	0.685 (0.137)	**1.000** (0.000)	0.504 (0.005)	0.565 (0.060)	0.624 (0.108)	0.603 (0.051)	0.739 (0.116)
insects ($1 \to 2$)	0.976 (0.022)	0.887 (0.054)	0.902 (0.041)	0.967 (0.025)	0.959 (0.037)	0.698 (0.057)	0.733 (0.033)	0.786 (0.031)	**1.000** (0.000)	0.999 (0.002)
insects ($2 \to 3$)	0.968 (0.031)	0.981 (0.018)	0.994 (0.005)	0.999 (0.001)	**1.000** (0.001)	0.895 (0.052)	1.000 (0.000)	0.999 (0.000)	0.987 (0.008)	0.996 (0.009)
insects ($3 \to 4$)	0.915 (0.045)	0.804 (0.068)	0.821 (0.046)	0.909 (0.036)	0.940 (0.051)	0.688 (0.066)	0.691 (0.021)	0.687 (0.023)	**0.995** (0.003)	0.987 (0.007)
insects ($4 \to 5$)	0.989 (0.015)	0.991 (0.015)	0.998 (0.003)	1.000 (0.001)	**1.000** (0.000)	0.878 (0.071)	1.000 (0.000)	1.000 (0.000)	0.999 (0.001)	0.994 (0.010)
insects ($5 \to 6$)	0.974 (0.017)	0.890 (0.044)	0.922 (0.023)	0.961 (0.019)	0.982 (0.011)	0.860 (0.051)	0.932 (0.007)	0.933 (0.008)	**0.997** (0.001)	0.996 (0.002)
sensorless	0.832 (0.112)	0.999 (0.008)	0.523 (0.013)	**1.000** (0.000)	**1.000** (0.000)	0.501 (0.003)	**1.000** (0.000)	**1.000** (0.000)	0.715 (0.139)	0.582 (0.083)
particle	0.860 (0.129)	0.876 (0.107)	0.530 (0.015)	0.922 (0.101)	**0.941** (0.089)	0.503 (0.004)	0.786 (0.143)	0.861 (0.127)	0.706 (0.143)	0.526 (0.035)
Average Ranking	5.32	4.96	7.28	3.78	**2.96**	9.54	5.26	4.97	5.33	5.60
Nemenyi p-value	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	-	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$

Table 5: FPR (top) and AUC (bottom) achieved by the considered methods in the high-ratio setting, namely when histogram-based methods define $K = 16$ bins and the monitored batches contain $\nu = 128$ points. In parenthesis, the width of the 95%-confidence interval of the results.

	QT	QT (PCA)	KQT (Euclidean)	KQT (Mahalanobis)	KQT (Weighted Maha.)	ElkM	SPLL(C=3)	SPLL(C=3) (PCA)	DT	DT (PCA)
unimodal	4.83% (0.48%)	4.81% (0.46%)	4.86% (0.47%)	4.82% (0.45%)	4.83% (0.48%)	4.82% (0.53%)	5.46% (0.75%)	5.92% (1.04%)	7.84% (1.16%)	7.75% (1.17%)
bimodal	4.80% (0.45%)	4.81% (0.46%)	4.80% (0.46%)	4.81% (0.44%)	4.80% (0.45%)	4.82% (0.51%)	5.53% (0.75%)	6.02% (1.06%)	7.65% (1.20%)	7.62% (1.09%)
nino	5.04% (0.49%)	4.99% (0.50%)	5.00% (0.53%)	5.02% (0.53%)	5.01% (0.54%)	4.83% (0.55%)	6.14% (1.21%)	7.69% (2.05%)	7.55% (1.20%)	7.57% (1.16%)
protein	4.97% (0.50%)	4.98% (0.56%)	4.97% (0.52%)	4.98% (0.54%)	5.03% (0.55%)	4.88% (0.61%)	13.15% (3.54%)	8.42% (2.33%)	7.65% (1.25%)	7.64% (1.25%)
spruce	4.81% (0.50%)	4.83% (0.48%)	4.82% (0.47%)	4.84% (0.49%)	4.90% (0.47%)	4.86% (0.59%)	11.43% (3.93%)	11.56% (3.97%)	7.56% (1.21%)	7.57% (1.16%)
lodgepole	4.83% (0.47%)	4.82% (0.50%)	4.85% (0.49%)	4.80% (0.47%)	4.90% (0.50%)	4.92% (0.57%)	10.78% (4.64%)	10.89% (4.68%)	7.60% (1.14%)	7.58% (1.12%)
credit	4.83% (0.47%)	4.96% (0.54%)	4.89% (0.48%)	4.85% (0.46%)	5.06% (0.56%)	4.96% (0.68%)	8.67% (2.26%)	16.06% (3.63%)	7.63% (1.15%)	7.59% (1.23%)
insects ($1 \to 2$)	4.92% (0.50%)	4.93% (0.51%)	4.91% (0.50%)	4.93% (0.52%)	5.19% (0.64%)	4.93% (0.63%)	5.90% (2.04%)	6.48% (2.15%)	7.57% (1.16%)	7.60% (1.20%)
insects ($2 \to 3$)	4.93% (0.53%)	4.91% (0.54%)	4.92% (0.52%)	4.96% (0.52%)	5.25% (0.62%)	4.96% (0.65%)	5.54% (1.85%)	6.16% (1.96%)	7.60% (1.19%)	7.59% (1.23%)
insects ($3 \to 4$)	4.92% (0.48%)	4.89% (0.52%)	4.90% (0.52%)	4.88% (0.53%)	5.22% (0.64%)	4.89% (0.58%)	6.09% (1.99%)	6.69% (2.11%)	7.59% (1.19%)	7.54% (1.17%)
insects ($4 \to 5$)	4.92% (0.50%)	4.95% (0.52%)	4.91% (0.51%)	4.92% (0.54%)	5.25% (0.65%)	4.91% (0.61%)	5.48% (1.76%)	6.01% (1.84%)	7.63% (1.24%)	7.56% (1.17%)
insects ($5 \to 6$)	4.91% (0.54%)	4.90% (0.54%)	4.90% (0.56%)	4.92% (0.53%)	5.26% (0.72%)	4.90% (0.64%)	5.86% (2.05%)	6.19% (2.11%)	7.61% (1.22%)	7.63% (1.24%)
sensorless	4.84% (0.50%)	5.01% (0.55%)	4.82% (0.49%)	5.01% (0.56%)	7.42% (1.61%)	4.93% (0.61%)	4.33% (1.03%)	4.83% (0.76%)	7.55% (1.19%)	7.58% (1.22%)
particle	4.85% (0.50%)	4.87% (0.51%)	4.81% (0.46%)	4.94% (0.52%)	5.80% (1.02%)	4.84% (0.61%)	5.93% (2.01%)	6.07% (2.05%)	7.52% (1.10%)	7.60% (1.19%)
unimodal	0.957 (0.079)	0.976 (0.057)	0.946 (0.105)	0.993 (0.016)	0.994 (0.013)	0.874 (0.154)	**0.996** (0.006)	0.989 (0.040)	0.786 (0.167)	0.806 (0.190)
bimodal	0.900 (0.110)	0.930 (0.090)	0.904 (0.118)	0.954 (0.060)	**0.968** (0.042)	0.821 (0.158)	0.915 (0.126)	0.895 (0.164)	0.751 (0.155)	0.767 (0.160)
nino	0.845 (0.143)	0.905 (0.135)	0.607 (0.072)	0.904 (0.138)	**0.922** (0.122)	0.528 (0.029)	0.816 (0.172)	0.841 (0.183)	0.726 (0.152)	0.582 (0.081)
protein	0.899 (0.104)	0.985 (0.051)	0.617 (0.074)	0.993 (0.035)	**0.995** (0.027)	0.514 (0.015)	0.918 (0.118)	0.954 (0.093)	0.704 (0.148)	0.595 (0.085)
spruce	0.999 (0.014)	1.000 (0.000)	0.601 (0.066)	**1.000** (0.000)	**1.000** (0.000)	0.507 (0.007)	1.000 (0.000)	1.000 (0.000)	1.000 (0.002)	1.000 (0.002)
lodgepole	1.000 (0.000)	1.000 (0.000)	0.654 (0.099)	**1.000** (0.000)	**1.000** (0.000)	0.511 (0.016)	1.000 (0.002)	1.000 (0.002)	1.000 (0.000)	1.000 (0.000)
credit	0.698 (0.079)	0.867 (0.127)	0.602 (0.053)	0.780 (0.146)	**1.000** (0.000)	0.508 (0.011)	0.597 (0.085)	0.660 (0.132)	0.695 (0.091)	0.820 (0.131)
insects ($1 \to 2$)	0.998 (0.005)	0.962 (0.048)	0.962 (0.035)	0.972 (0.039)	0.993 (0.019)	0.836 (0.071)	0.810 (0.035)	0.866 (0.029)	**1.000** (0.000)	1.000 (0.000)
insects ($2 \to 3$)	0.993 (0.017)	0.995 (0.012)	1.000 (0.001)	**1.000** (0.000)	**1.000** (0.000)	0.962 (0.014)	**1.000** (0.000)	1.000 (0.000)	0.999 (0.002)	1.000 (0.001)
insects ($3 \to 4$)	0.983 (0.029)	0.897 (0.078)	0.904 (0.054)	0.942 (0.049)	0.990 (0.024)	0.835 (0.084)	0.753 (0.025)	0.745 (0.028)	**1.000** (0.000)	1.000 (0.001)
insects ($4 \to 5$)	0.998 (0.008)	0.997 (0.008)	1.000 (0.000)	**1.000** (0.000)	**1.000** (0.000)	0.950 (0.021)	**1.000** (0.000)	1.000 (0.000)	1.000 (0.000)	0.999 (0.004)
insects ($5 \to 6$)	0.999 (0.003)	0.971 (0.033)	0.985 (0.016)	0.992 (0.008)	1.000 (0.000)	0.963 (0.017)	0.979 (0.004)	0.979 (0.005)	**1.000** (0.000)	1.000 (0.000)
sensorless	0.862 (0.120)	1.000 (0.004)	0.542 (0.027)	**1.000** (0.000)	**1.000** (0.000)	0.502 (0.003)	**1.000** (0.000)	**1.000** (0.000)	0.738 (0.179)	0.595 (0.104)
particle	0.886 (0.116)	0.931 (0.090)	0.555 (0.030)	0.976 (0.051)	**0.985** (0.039)	0.506 (0.006)	0.838 (0.135)	0.901 (0.112)	0.798 (0.140)	0.542 (0.054)
Average Ranking	5.24	4.93	7.08	3.82	**2.98**	9.37	5.57	5.34	5.11	5.56
Nemenyi p-value	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	-	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-16}$

References

[Alippi et al., 2017] Alippi, C., Boracchi, G., and Carrera, D. (2017). Ccm: Controlling the change magnitude in high dimensional data. In Angelov, P., Manolopoulos, Y., Iliadis, L., Roy, A., and Vellasco, M., editors, *Advances in Big Data*, pages 216–225, Cham. Springer International Publishing.

[Balakrishnan and Rao, 1998] Balakrishnan, N. and Rao, C. R. (1998). Handbook of statistics. v. 16: Order statistics: theory and methods.

[Boracchi et al., 2018] Boracchi, G., Carrera, D., Cervellera, C., and Maccio, D. (2018). Quanttree: Histograms for change detection in multivariate data streams. In *International Conference on Machine Learning*, pages 639–648. PMLR.

[Dal Pozzolo et al., 2017] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G. (2017). Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE transactions on neural networks and learning systems*, 29(8):3784–3797.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[Lehmann et al., 2005] Lehmann, E. L., Romano, J. P., and Casella, G. (2005). *Testing statistical hypotheses*, volume 3. Springer.

[Papoulis and Pillai, 2002] Papoulis, A. and Pillai, S. U. (2002). *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education.

[Shapiro and Wilk, 1965] Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

[Souza et al., 2020] Souza, V. M., dos Reis, D. M., Maletzke, A. G., and Batista, G. E. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858.

[Tipping, 1999] Tipping, M. E. (1999). Deriving cluster analytic distance functions from gaussian mixture models.