QuantTree: Histograms for Change Detection in Multivariate Data Streams

Giacomo Boracchi [1] **Diego Carrera** [1] **Cristiano Cervellera** [2] **Danilo Macciò** [2]

Abstract

We address the problem of detecting distribution changes in multivariate data streams by means of histograms. Histograms are very general and flexible models, which have been relatively ignored in the change-detection literature as they often require a number of bins that grows unfeasibly with the data dimension. We present QuantTree, a recursive binary splitting scheme that adaptively defines the histogram bins to ease the detection of any distribution change. Our design scheme implies that *i*) we can easily control the overall number of bins and *ii*) the bin probabilities do not depend on the distribution of stationary data. This latter is a very relevant aspect in change detection, since thresholds of tests statistics based on these histograms (e.g., the Pearson statistic or the total variation) can be numerically computed from univariate and synthetically generated data, yet guaranteeing a controlled false positive rate. Our experiments show that the proposed histograms are very effective in detecting changes in high dimensional data streams, and that the resulting thresholds can effectively control the false positive rate, even when the number of training samples is relatively small.

1. Introduction

Change detection, namely the problem of analyzing a data stream to detect changes in the data-generating distribution, is very relevant in machine-learning and is typically addressed in an unsupervised manner. This approach is generally dictated by many practical aspects, which include the unpredictability of the change and the fact that the training set often contains only stationary data. As a matter of fact, most change-detection tests in the literature (Basseville & Nikiforov, 1993; Lung-Yut-Fong et al., 2011; Ross et al., 2011; Kuncheva, 2013) consist of three major ingredients: *i*) a model describing the distribution of stationary data, ϕ_0, that is typically learned from a training set, *ii*) a test statistic $\mathcal{T}$ used to assess the conformance of test data with the learned model, and *iii*) a decision rule that monitors $\mathcal{T}$ to detect changes in ϕ_0. Needless to say, all these have to be wisely designed and combined to yield a sound test that can provide prompt detections as well as a controlled False Positive Rate (FPR), which is one of the primary concerns in change detection. Unfortunately, when it comes to monitoring multivariate data, it is difficult to find good density models and test statistics that do not depend on ϕ_0: this represents a severe limitation for real-world monitoring problems, where the stream distribution is unknown. Our work presents an efficient change-detection test for multivariate data that overcomes this limitation.

The first change-detection tests were developed to monitor univariate data streams in the statistical process control literature (Basseville & Nikiforov, 1993). In classification problems, changes in the data stream are known as concept drift (Gama et al., 2014) and are detected by monitoring the sequence of classification errors on supervised data (Harel et al., 2014; Alippi et al., 2013; Bifet & Gavalda, 2007). Many change-detection tests are parametric, i.e., they assume that ϕ_0 belongs to a known family, e.g., (Page, 1954), or are based on ad-hoc statistics that detect specific changes, e.g., the Hotelling statistic (Lehmann & Romano, 2006). Most nonparametric statistics are instead based on ranking, e.g., the Kolmogorov-Smirnov (Ross & Adams, 2012) and Lepage (Ross et al., 2011) statistics, and can be applied exclusively to univariate data.

There exist a few multivariate tests able to detect any distribution change (Lung-Yut-Fong et al., 2011; Justel et al., 1997). Two popular approaches consists either in reducing the data dimension by PCA (Kuncheva, 2013; Qahtan et al., 2015) or computing the likelihood with respect to a model fitted on a training set, e.g., a Gaussian mixture (Kuncheva, 2013; Alippi et al., 2016), a Gaussian process (Saatçi et al., 2010) or a kernel density estimator (Krempl, 2011). In the latter case the change-detection problem boils down to monitoring a univariate stream. Unfortunately, in these cases, $\mathcal{T}$ often depends on ϕ_0, and detection rules be-

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy. [2]Institute of Intelligent Systems for Automation, National Research Council, Genova, Italy. Correspondence to: Diego Carrera <diego.carrera@polimi.it>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

come heuristic in nature (Kuncheva, 2013; Ditzler & Polikar, 2011) preventing a proper control over the FPR. Histograms, which are perhaps the most natural candidates for describing densities, enable a different form of monitoring that is based on a comparison among distributions (Ditzler & Polikar, 2011; Boracchi et al., 2017). However, they are often implemented over regular grids and require a number of bins that grows exponentially with the data dimension. Only a few change-detection solutions (Dasu et al., 2006; Boracchi et al., 2017) adopt alternative partitioning schemes that scale well in high dimensions. In particular, kqd-trees (Dasu et al., 2006) were introduced as a variant of kd-trees (Bentley, 1975) to guarantee that all the leaves contain a minimum number of training samples and have a minimum size. In (Boracchi et al., 2017) it is shown that histograms built on uniform-density partitions rather than regular grids provide superior detection performance.

Our main contribution is QuantTree, a recursive binary splitting scheme that defines histograms for change-detection purposes. The most prominent advantage of using QuantTree is that the distribution of any statistic defined over the resulting histograms does not depend on ϕ_0. This implies that decision rules to be used in multivariate change-detection problems do not depend on the data, and can be numerically computed from synthetically generated univariate sequences. Moreover, histograms defined by QuantTree can have a pre-assigned number of bins and can be represented as a tree, thus enabling a very efficient computation of test statistics.

QuantTree (Section 3) iteratively divides the input space by means of binary splits on a single covariate, where the cutting points are defined by the quantiles of the marginal distributions. This splitting strategy is similar to the one adopted by kd-trees (Bentley, 1975), where the split is performed w.r.t. the median value of the marginal. Such a simple construction scheme can be handled analytically, as it is possible to prove (Section 4) that the distribution of each bin probability does not depend on ϕ_0. Our experiments (Section 5) show that QuantTree enables good detection performance in high dimensional streams. Moreover, when testing few samples, QuantTree guarantees a better FPR control than the Pearson goodness-of-fit test and tests based on empirical thresholds computed through bootstrap. We also show that histograms constructed with a few bins gathering the same density under ϕ_0 achieve higher power than monitoring schemes based on different histograms.

2. Problem Formulation

Before the change, namely in stationary conditions, data in the monitored stream $\mathbf{x} \in \mathbb{R}^d$ are independent and identically distributed (i.i.d.) realizations of a continuous random vector $\mathbf{X}_0$ having an unknown probability density function (pdf) ϕ_0, whose support is $\mathcal{X} \subseteq \mathbb{R}^d$. We assume that a training set $TR = \{\mathbf{x}_i \in \mathcal{X}, i = 1, \dots, N\}$ containing N stationary data (i.e., $\mathbf{x}_i \sim \phi_0$) is provided.

Histograms: we define a histogram as:

$$h = \{(S_k, \widehat{\pi}_k)\}_{k=1,\dots,K}, \tag{1}$$

where the K subsets $S_k \subseteq \mathcal{X}$ form a partition of $\mathbb{R}^d$, i.e., $\bigcup_{k=1}^K S_k = \mathbb{R}^d$ and $S_j \cap S_i = \emptyset$, for $j \neq i$, and each $\widehat{\pi}_k \in [0, 1]$ corresponds to the probability for data generated from ϕ_0 to fall inside S_k. Both the subsets $\{S_k\}_k$ and probabilities $\{\widehat{\pi}_k\}_k$ can be adaptively defined from training data TR, and in particular $\widehat{\pi}_k$ is typically estimated as $\widehat{\pi}_k = L_k/N$, i.e. the number of training samples L_K belonging to S_k over the number of points in TR.

Batch-wise monitoring: for the sake of simplicity, we analyze the incoming data in batches $W = \{\mathbf{x}_1, \dots, \mathbf{x}_\nu\}$ of ν samples. We detect changes by an hypothesis test (HT) which assesses whether data in W are consistent with a reference histogram h learned from TR. In particular, this hypothesis test can be stated as follows:

$$H_0 : W \sim \phi_0 \qquad \textit{vs} \qquad H_1 : W \sim \phi_1 \neq \phi_0 \tag{2}$$

where ϕ_1 represents the unknown post-change distribution. We focus on HTs that are based on a test statistic $\mathcal{T}_h$ defined over the histogram h, like for instance the Pearson statistic (Lehmann & Romano, 2006). Thus, $\mathcal{T}_h$ uniquely depends on $\{y_k\}_{k=1,\dots,K}$, where y_k denotes the number of samples in W falling in S_k. We detect a change in the incoming W when

$$\mathcal{T}_h(W) = \mathcal{T}_h(y_1, \dots, y_K) > \tau, \tag{3}$$

where $\tau \in \mathbb{R}$ is a threshold that controls the FPR, namely the proportion of type I errors (Lehmann & Romano, 2006).

Goal: our goal is two-fold, *i*) learn a histogram h from TR to be used for change-detection purposes and *ii*) for each given test statistic $\mathcal{T}_h$ and reference FPR value α, define a threshold τ such that

$$P_{\phi_0}(\mathcal{T}_h(W) > \tau) \leq \alpha, \tag{4}$$

where P_{ϕ_0} denotes the probability under the null hypothesis that W contains samples generated from ϕ_0.

There are two important comments. First, while (3) might seem an oversimplified monitoring scheme, this is enough to demonstrate that when histograms are built through QuantTree, the monitoring can be performed independently of ϕ_0. As a consequence, test statistics $\mathcal{T}_h$ can be potentially employed in sequential monitoring schemes like (Ross & Adams, 2012). Second, we focus on general-purpose tests, which are able to detect any distribution change $\phi_0 \rightarrow \phi_1$ as well as on histograms that can model densities in high dimensions, i.e., $d \gg 1$.

Algorithm 1 QuantTree

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

3. The QuantTree Algorithm

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

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

QuantTree divides $\mathcal{X}$ in a given number of subsets, where each S_k has an estimated probability $\widehat{\pi}_k \simeq \pi_k$, and the equality holds when $\pi_k N$ is integer. Since the probabilities π_k are set a priori, in what follows we use π_k in place of $\widehat{\pi}_k$. Indexes i and parameter γ are randomly chosen to add variability to the histogram construction. Figure 1(a) shows a tree obtained from a bivariate Gaussian training set, defined by $K = 4$ bins, each having probability $\pi_k = N/4$.

[1]The implementation of QuantTree is available at `http://home.deib.polimi.it/boracchi/Projects`

Algorithm 2 Numerical procedure to compute thresholds

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

3.1. Computation of Distribution-Free Test Statistics

A key feature of a histogram computed by QuantTree is that any statistic $\mathcal{T}_h$ built over it has a distribution that is independent from ϕ_0. This result follows from Theorem 1, that is proved in Section 4.

Theorem 1. *Let $\mathcal{T}_h(\cdot)$ be defined as in* (3) *over the histogram h computed by QuantTree. When $W \sim \phi_0$, the distribution of $\mathcal{T}_h(W)$ depends only on ν, N and $\{\pi_k\}_k$.*

Theorem 1 implies that we can numerically compute the thresholds for any statistic $\mathcal{T}_h$ defined on histograms, provided ν, N and $\{\pi_k\}$, thus disregarding ϕ_0 and the data dimension d. To this end, we synthetically generate data from a conveniently chosen distribution ψ_0, and we follow the procedure outlined in Algorithm 2 to estimate the threshold τ for HT in (2) yielding a desired FPR α. At first we generate B training sets $\{TR_b\}_{b=1,\dots,B}$, sampling N points from ψ_0 and, for each training set, we build a histogram h_b using QuantTree (lines 2-3). Then, for each h_b we generate a batch W_b of ν points drawn from ψ_0, and compute the value of the statistic $t_b = \mathcal{T}_h(W_b)$ (lines 4-5). Finally, we estimate τ (line 7) from the set $T_B = \{t_1, \dots, t_B\}$ as the $1 - \alpha$ quantile of the empirical distribution of $\mathcal{T}_h$ over the generated batches, i.e.

$$\tau = \min\Big\{ t \in T_B \;:\; \#\{v \in T_B \;:\; v > t\} \leq \alpha B \Big\}, \quad (5)$$

where $\#A$ denotes the cardinality of a set A.

To take full advantage of the distribution-free nature of the procedure, we set ψ_0 to a univariate uniform distribution

$U(0,1)$. This allows to obtain high accuracy on the estimation of the thresholds, since we can use very large values of B with limited computational cost.

3.2. Considered Statistics

We consider two meaningful examples of statistics $\mathcal{T}_h$ that can be employed for batch-wise monitoring through histograms: the Pearson statistic and the total variation (Lehmann & Romano, 2006). The Pearson statistic is defined as

$$\mathcal{T}_h^P(W) = \sum_{k=1}^{K} \frac{(y_k - \nu\pi_k)^2}{\nu\pi_k}, \tag{6}$$

while the total variation is defined as

$$\mathcal{T}_h^{TV}(W) = \frac{1}{2}\sum_{k=1}^{K} |y_k - \nu\pi_k| \ . \tag{7}$$

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

These two statistics will be used for our experiments in Section 5, using thresholds reported in Table 1 for different values of N, K, ν and choosing $\pi_k = 1/K, k = 1, \dots, K$. These values have been computed applying the procedure described in Algorithm 2 with $B = 2.5 \cdot 10^6$. We note that both statistics $\mathcal{T}_h^P$ and $\mathcal{T}_h^{TV}$ assume only discrete values, therefore it is not always possible to set the threshold τ yielding the FPR exactly equal to α, but only to ensure that the FPR does not exceed α.

3.3. Computational Remarks

We remark that since the histogram h computed by QuantTree is exclusively defined on the marginal probabilities of single components, the dimensionality of the input data d does not impact the overall computational cost. In fact, the computational cost of building a QuantTree is dominated by sorting the covariates (Algorithm 1 line 6), which is performed K times on an progressively smaller number of samples at each iteration. Therefore, the overall complexity of constructing a QuantTree is $O(KN \log N)$. In case of univariate distribution (i.e., $d = 1$), the complexity is reduced to $O(N \log N)$, since the partition $\{S_k\}_k$ can be defined through a single sorting operation.

Since any histogram h computed by QuantTree can be represented as a tree structure, it is very efficient to identify the bin where any testing point belongs to. In fact, during monitoring, at most K IF-THEN operations (that reduces to $\log K$ when $d = 1$) have to be performed for each input sample $\mathbf{x}$. Moreover, in contrast with histograms based on regular grids, the number of bins K is here a priori defined, and does not need to grow exponentially with d.

α	Pearson		Total Variation			
	$K=32$	$K=128$	$K=32$	$K=128$	N	ν
0.001	64	192	25	43	4096	64
	62.75	187	52	85	16384	256
0.01	54	172	23	42	4096	64
	53.25	171	47	81	16384	256
0.05	46	156	21	41	4096	64
	45.75	157	44	78	16384	256

Table 1: Examples of thresholds τ that guarantee FPR below α in HT (2) using a uniform histogram h, i.e. by settings $\pi_k = 1/K$, $k = 1, \dots, K$. The thresholds are computed by Algorithm 2 using $U(0,1)$ as ψ_0 and different values of N, ν and K.

4. Theoretical Analysis

We prove Theorem 1 showing that the distribution of any test statistic $\mathcal{T}_h$ defined over an histogram h computed by QuantTree does not depend on ϕ_0. To this end, we first prove some preliminary propositions to characterize the distribution of the true probability of each bin S_k under ϕ_0:

$$p_k = P_{\phi_0}(S_k), \tag{8}$$

which is also a random variable as it depends on the training data TR.

For the sake of simplicity, we assume that QuantTree always splits with respect to the left tail, i.e., $\gamma = 0$ in line 8 of Algorithm 1 (proofs hold when $\gamma \sim$ Bernoulli(0.5)) and, to simplify the notation, we will omit the subscript ϕ_0 from P_{ϕ_0}, thus P denotes the probability computed w.r.t. ϕ_0. The following proposition will be used to derive the distributions of p_k.

Proposition 1. *Let $\mathbf{x}_1, \dots, \mathbf{x}_M$ be i.i.d. realizations of a continuous random vector $\mathbf{X}$ defined over $\mathcal{D} \subseteq \mathbb{R}^d$. Let us define the i-th component of $\mathbf{x}$ as $z = [\mathbf{x}]_i$, and denote with $z_{(1)} \leq z_{(2)} \leq \cdots \leq z_{(M)}$ the M sorted components of $\mathbf{x}_1, \dots, \mathbf{x}_M$. For any $L \in \{1, \dots, M\}$ we define the set*

$$Q_{i,L} := \{\mathbf{x} \in \mathcal{D} \ : \ [\mathbf{x}]_i \leq z_{(L)}\}. \tag{9}$$

Then, for each $i \in \{1, \dots, d\}$, the random variable $p = P_{\mathbf{X}}(Q_{i,L})$ is distributed as a Beta$(L, M - L + 1)$.

Proof. The proof consists of showing that p is an order statistic of the uniform distribution, which in turns follows a Beta distribution. For this purpose, we consider $\mathbf{X}$ defined over $\mathbb{R}^d$ and $P_{\mathbf{X}}(\mathbb{R}^d \backslash \mathcal{D}) = 0$, thus p can be expressed as

$$\begin{aligned} p = P_{\mathbf{X}}(Q_{i,L}) &= P_{\mathbf{X}}(\mathbf{x} \in \mathbb{R}^d \ : \ [\mathbf{x}]_i \leq z_{(L)}) = \\ &= P_Z(z \in \mathbb{R} \ : \ z \leq z_{(L)}), \end{aligned} \tag{10}$$

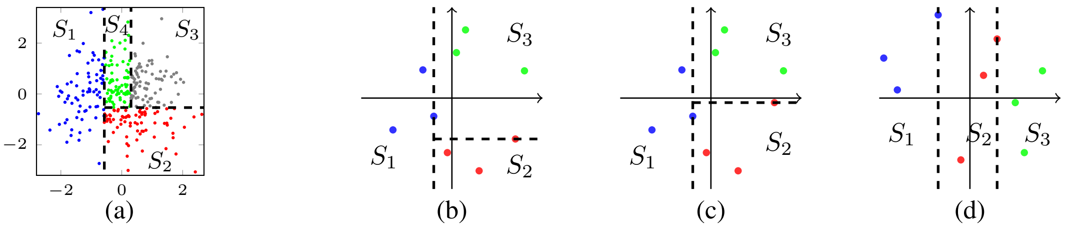

Figure 1: (a) A histogram $\{S_k\}_{k=1,\dots,4}$ computed by QuantTree to yield uniform density on the bins. (b)-(d) Examples of values assumed by $\widetilde{L}_k$ in three different configurations, when $N = 9$ and $L_1 = L_2 = L_3 = 3$. In these cases $\widetilde{L}_1 = L_1$, while in (b) $\widetilde{L}_2 = 3$, in (c) $\widetilde{L}_2 = 5$, and in (d) $\widetilde{L}_2 = 6$. Note that when QuantTree chooses always the same component, we have that $\widetilde{L}_2 = L_1 + L_2$, as in (d).

where P_Z denotes the marginal probability of $Z = [\mathbf{X}]_i$, namely the marginal of $\mathbf{X}$ w.r.t. the component i. We denote with F_Z the cumulative distribution of Z and define $U = F_Z^{-1}(Z)$ and $u_n = F_Z^{-1}(z_n)$, $n = 1, \dots, M$, where

$$F_Z^{-1}(z) = \inf\{t \in \mathbb{R} \,:\, F_Z(t) > z\}. \tag{11}$$

The function $F_Z^{-1}(\cdot)$ is monotonically nondecreasing, thus it preserves the order and the L-th sorted value of $\{u_n\}$ can be computed as $u_{(L)} = F_Z^{-1}(z_{(L)})$. Then, (10) becomes

$$\begin{aligned} p = P_Z(z \in \mathbb{R} \,:\, z \le z_{(L)}) = & \qquad (12)\\ = P_U(u \in [0,1] \,:\, u \le u_{(L)}) = F_U(u_{(L)}) = u_{(L)}. \end{aligned}$$

Since U follows a uniform distribution over $[0, 1]$, it follows that p is the L-th order statistic of the uniform distribution, that is a distributed as a $\text{Beta}(L, M - L + 1)$ (Balakrishnan & Rao, 1998). □

Thus, p_1 in (8), namely the probability of S_1 under ϕ_0, is distributed as a $\text{Beta}(L_1, N - L_1 + 1)$. To derive the distribution of the remaining p_k, $k \ge 2$, we define the conditional probability

$$P_{S_1}(\mathbf{x} \in A) = P_{\phi_0}(\mathbf{x} \in A \mid \mathbf{x} \notin S_1), \tag{13}$$

where A is any Borel subset of $\mathcal{X}$. Then, from the definition of conditional probability and the fact that $\mathbf{x}_1, \dots, \mathbf{x}_N$ are i.i.d. according to ϕ_0, it can be easily proved that the $N - L_1$ points that do not belong to S_1 are i.i.d. according to P_{S_1}. Therefore, we can apply Proposition 1 to the subset of the $N - L_1$ points that do not fall in S_1 by setting $\mathcal{D} = \mathbb{R}^d \setminus S_1$ and considering P_{S_1} in place of $P_{\mathbf{X}}$. Thus, the random variable $\widetilde{p}_2 = P_{S_1}(S_2)$ is distributed as $\text{Beta}(L_2, N_2 - L_2, 1)$, where $N_2 = N - N_1$. Iterating the above procedure, we obtain that all the random variables $\widetilde{p}_k$, $k = 1, \dots, K$, defined as[2]

$$\widetilde{p}_k = P_{\bigcup_{j=1}^{k-1} S_j}(S_k), \tag{14}$$

are distributed as $\text{Beta}(L_k, N_k - L_k + 1)$, where $N_k = N - \sum_{j=1}^{k-1} N_j$.

We remark the different roles of p_k and $\widetilde{p}_k$. While p_k in (8) the measure of the bin S_k under ϕ_0, $\widetilde{p}_k$ in (14) is the ratio between p_k and the measure under ϕ_0 of $\mathcal{X}_k = \mathbb{R}^d \setminus \bigcup_j^{k-1} S_j$, namely the space that remains to be partitioned at step k. As an example, for a tree with $K = 3$ leaves, if we set target probabilities $\pi_1 = \pi_2 = \pi_3 = 1/3$, we obtain $\widetilde{p}_1 = 1/3$, $\widetilde{p}_2 = 1/2$ and $\widetilde{p}_3 = 1$. To prove Theorem 1 we need to derive the distribution of p_k, that are expressed in terms of $\widetilde{p}_k$ by the following proposition.

Proposition 2. *In case of histograms defined by QuantTree, the following relation holds between p_k and $\widetilde{p}_k$:*

$$p_k = \widetilde{p}_k \cdot (1 - \sum_{j=1}^{k-1} p_j) = \widetilde{p}_k \prod_{j=1}^{k-1} (1 - \widetilde{p}_j). \tag{15}$$

Proof. From the law of total probability we have that

$$\begin{aligned} p_k &= P_{\phi_0}(\mathbf{x} \in S_k) = \\ &= P_{\phi_0}\left(\mathbf{x} \in S_k \mid \mathbf{x} \notin \cup_{j=1}^{k-1} S_j\right) \cdot P_{\phi_0}\left(\mathbf{x} \notin \cup_{j=1}^{k-1} S_j\right) + \\ &+ P_{\phi_0}\left(\mathbf{x} \in S_k \mid \mathbf{x} \in \cup_{j=1}^{k-1} S_j\right) \cdot P_{\phi_0}\left(\mathbf{x} \in \cup_{j=1}^{k-1} S_j\right). \end{aligned} \tag{16}$$

Since sets $\{S_k\}$ defined by QuantTree are disjoint, it follows that S_k and $\bigcup_{j=1}^{k-1} S_j$ are also disjoint, thus the second term in the sum in (16) is equal to 0. The first equality in (15) follows from the definition of $\widetilde{p}_k = P_{\phi_0}(\mathbf{x} \in S_k \mid \mathbf{x} \notin \bigcup_{j=1}^{k-1} S_j)$ and the fact that $P_{\phi_0}(\mathbf{x} \notin \bigcup_{j=1}^{k-1} S_j) = 1 - \sum_{j=1}^{k-1} p_j$. The second equality in (15) can be proved by induction over j. □

The following proposition allows us to express p_j as a product of independent Beta distributions.

Proposition 3. *The random variables $\widetilde{p}_k$ defined over histograms computed by QuantTree are independent.*

Proof. To prove the independence of the $\widetilde{p}_k$, $k = 1, \dots, K$, we show that $\widetilde{p}_k$ is independent from $\widetilde{p}_j$, $j = 1, \dots, k - 1$. In particular, we prove that

$$P_{\phi_0}(\widetilde{p}_k \le t_k \mid \widetilde{p}_j = t_j,\, j = 1, \dots, k{-}1) = P_{\phi_0}(\widetilde{p}_k \le t_k). \tag{17}$$

To this end, we follow the proof of Proposition 1, and express $\widetilde{p}_k$ as an order statistic of the uniform distribution.

[2] We adopt the following conventions: an empty union of sets is the empty set, an empty sum is zero, and an empty product is 1.

At iteration k, QuantTree randomly selects a dimension i_k and performs a split w.r.t. the L_k-th order statistic of the i_k components over the remaining N_k points (line 9 of Algorithm 1). Let $\widetilde{L}_k$ be the position of this splitting point in $\{z_n = [\mathbf{x}_n]_{i_k}, n = 1, \dots, N\}$, namely the sequence of ordered i_k components of all the points in TR. The value of $\widetilde{L}_k \in \mathbb{N}$ depends on realizations $\mathbf{x}_1, \dots, \mathbf{x}_N$, and is a random variable ranging in $\{L_k, \dots, M_k\}$, where $M_k = \sum_{j=1}^{k} L_j$. Obviously, at the first iteration $L_1 = \widetilde{L}_1$ but then the two may differ, as shown in Figure 1. Let us now consider the splitting point with respect to $\widetilde{L}_k$, i.e., $z_{(\widetilde{L}_k)}$. From the definition of $\widetilde{p}_k$ we have that

$$\widetilde{p}_k = P_{\bigcup_{j=1}^{k-1} S_j}(S_k) = P_{\bigcup_{j=1}^{k-1} S_j}(z \leq z_{(\widetilde{L}_k)}). \tag{18}$$

As in the proof of Proposition 1, we denote with F_Z the cdf of $Z = [\mathbf{X}]_{i_k}$, and define $U = F_Z^{-1}(Z)$, that has a uniform distribution on $[0, 1]$. Therefore it holds that

$$\widetilde{p}_k = P_{\bigcup_{j=1}^{k-1} S_j}(z \leq z_{(\widetilde{L}_k)}) = P_{\bigcup_{j=1}^{k-1} S_j}(u \leq u_{(\widetilde{L}_k)}) = \\ = F_U(u_{(\widetilde{L}_k)}) = u_{(\widetilde{L}_k)}. \tag{19}$$

We use the law of total probability w.r.t. the events $\{\widetilde{L}_k = a\}$, $a \in \{L_k, \dots, M_k\}$, to decompose the left hand side in (17) as (for simpler notation we omit the expression $j = 1, \dots, k-1$ in what follows):

$$P_{\phi_0}(\widetilde{p}_k \leq t_k \mid \widetilde{p}_j = t_j) = P_{\phi_0}(u_{(\widetilde{L}_k)} \leq t_k \mid \widetilde{p}_j = t_j) = \\ = \sum_{a=L_k}^{M_k} P_{\phi_0}(u_{(\widetilde{L}_k)} \leq t_k \mid \widetilde{L}_k = a, \widetilde{p}_j = t_j) \cdot P_{\phi_0}(\widetilde{L}_k = a) \\ = \sum_{a=L_k}^{M_k} P_{\phi_0}(u_{(a)} \leq t_k \mid \widetilde{p}_j = t_j) \cdot P_{\phi_0}(\widetilde{L}_k = a). \tag{20}$$

Since the distribution of $u_{(a)}$ does not depend on $\widetilde{p}_j$, we have that $P_{\phi_0}(u_{(a)} \leq t_k \mid \widetilde{p}_j = t_j) = P_{\phi_0}(u_{(a)} \leq t_k)$, therefore it follows

$$P_{\phi_0}(\widetilde{p}_k \leq t_k \mid \widetilde{p}_j = t_j) = \\ = \sum_{a=L_k}^{M_k} P_{\phi_0}(u_{(a)} \leq t_k) \cdot P_{\phi_0}(\widetilde{L}_k = a) \\ = \sum_{a=L_k}^{M_k} P_{\phi_0}(u_{(\widetilde{L}_k)} \leq t_k \mid \widetilde{L}_k = a) \cdot P_{\phi_0}(\widetilde{L}_k = a) = \\ = P_{\phi_0}(u_{(\widetilde{L}_k)} \leq t_k) = P_{\phi_0}(\widetilde{p}_k \leq t_k), \tag{21}$$

and (17) is proved. □

The proof of Theorem 1 follows from Proposition 3.

Proof of Theorem 1. For any stationary distribution ϕ_0, the random vector $[y_1, \dots, y_K]$ conditioned on $p_1, \dots, p_K$ follows a Multinomial distribution with parameters $(\nu, p_1, \dots, p_K)$ (White et al., 2009). From Proposition 3 each p_k is a product of independent Beta distributions, thus depends only on $\{L_k\}$ and it is independent from ϕ_0. Therefore any statistic $\mathcal{T}_h$ that is a function of $\{y_k\}$ depends only on ν, and on N and $\{\pi_k\}$ which determines $\{L_k\}$. □

5. Experiments

We quantitatively assess the advantages of change-detection tests based on QuantTree w.r.t. other general-purpose tests able to detect any distribution change $\phi_0 \to \phi_1$. In particular, we show that: *i*) thresholds provided by Algorithm 2 can better control the FPR w.r.t. alternatives based on asymptotic results or bootstrap *ii*) HT based on histograms provided by QuantTree yielding a uniform-density partition of $\mathbb{R}^d$ achieve higher power than other partitioning schemes.

5.1. Datasets and Change Models

We employ both synthetic and real-world datasets: Synthetic datasets are generated by choosing, for each dimension $d \in \{2, 8, 32, 64\}$, 250 pairs (ϕ_0, ϕ_1) of Gaussians, where ϕ_0 has a randomly defined covariance, and $\phi_1 = \phi_0(Q \cdot +\mathbf{v})$ is a roto-transalation of ϕ_0 such that the symmetric Kullback-Leibler divergence $\text{sKL}(\phi_0, \phi_1) = 1$. We control $\text{sKL}(\phi_0, \phi_1)$ by the CCM framework (Carrera & Boracchi, 2017), which guarantees all the changes to have the same magnitude. This is required when comparing detection performance in different dimensions.

We also employ four real-world high-dimensional sets: *MiniBooNE particle identification* ("particle", $d = 50$), *Physicochemical Properties of Protein Tertiary Structure* ("protein", $d = 9$), *Sensorless Drive Diagnosis* ("sensorless", $d = 48$) from the UCI Machine Learning Repository (Lichman, 2013), and *Credit Card Fraud Detection* ("credit", $d = 29$) from (Dal Pozzolo et al., 2015). We standardize these datasets and add to each component of the "particle" and "sensorless" an imperceivable amount of noise $\eta \sim N(0, 0.001)$ to scramble the many repeated values, which harms histogram construction. For each dataset we simulate 150 changes $\phi_0 \to \phi_1$ by randomly selecting TR and defining a random shift drawn from a normal distribution.

5.2. Change Detection Methods

Four of the considered methods rely on the same histogram computed through QuantTree (Algorithm 1) to provide a uniform density partition of $\mathbb{R}^d$, i.e. the target probabilities are $\pi_k = 1/K$, $\forall k$. These methods differ only for the threshold adopted and have been considered mainly to investigate the control over false positives.

Pearson Distribution Free / TV Distribution Free: thresholds are computed by Algorithm 2 for the Pearson $\mathcal{T}_h^P$ (6) and the total variation $\mathcal{T}_h^{TV}$ statistics (7), respectively. The adopted thresholds are reported in Table 1.

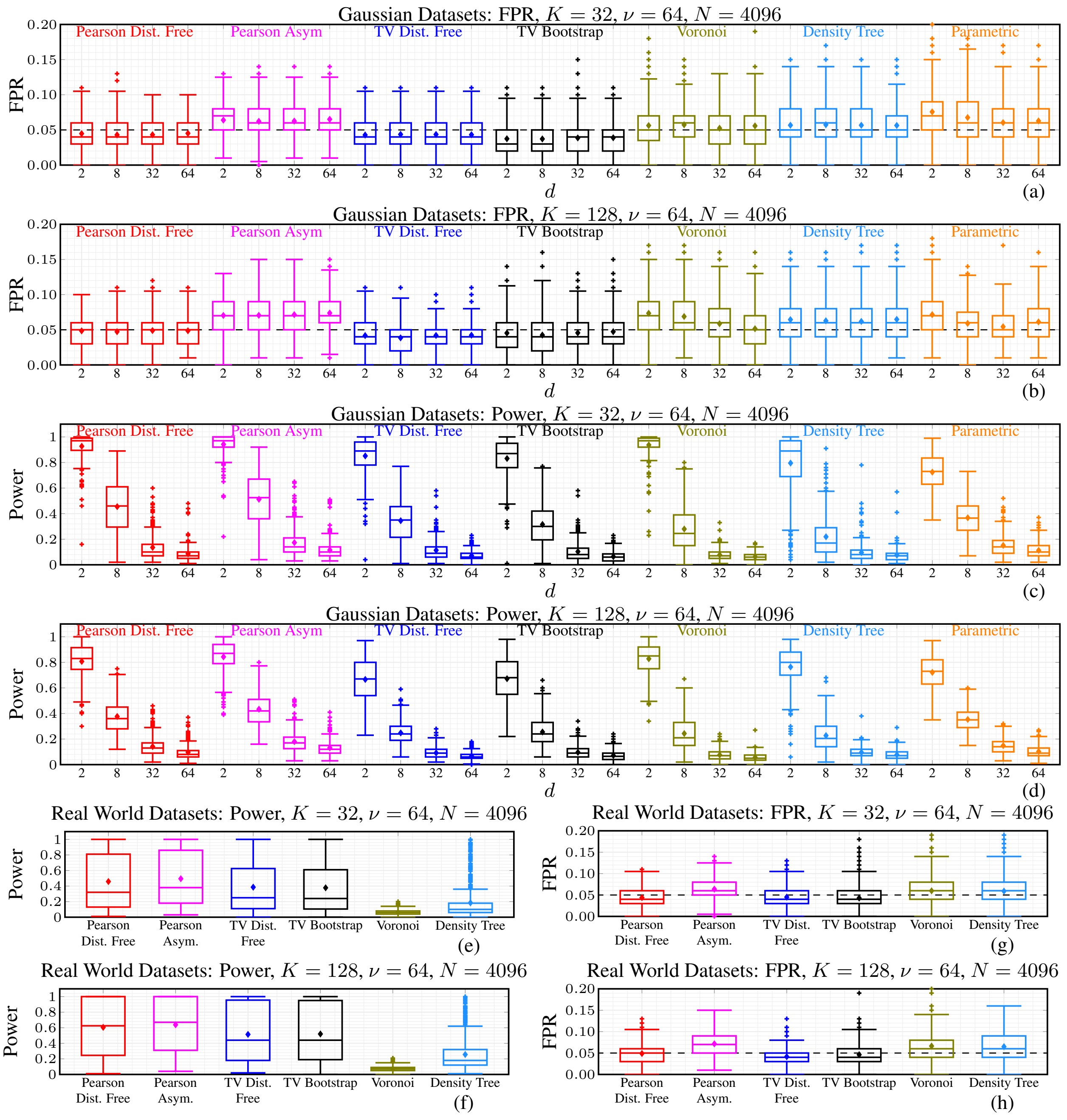

Figure 2: Results on both synthetic (a)-(d) and real world (e)-(h) datasets using the *small* TR configuration. In (a) and (b) we report the FPR computed on the Gaussian datasets using $K = 32$ and $K = 128$ bins, respectively, while (c) and (d) reports the corresponding powers. Thresholds computed by Algorithm 2 successfully yield averaged FPR values smaller than the desired value α disregarding the data dimension d, as expected by Theorem 1. Moreover, the methods based on histograms computed by QuantTree achieve the highest power. The FPR obtained on the real datasets are shown in (e) and (f), for $K = 32$ and $K = 128$, respectively, and the powers are reported in (g) and (h). Also in this cases Algorithm 2 provides thresholds that successfully control the FPR and, as on Gaussian datasets, methods based on uniform histograms outperform the other in terms of power.

Pearson Asymptotic: thresholds for $\mathcal{T}_h^P$ are provided from the classic χ^2 goodness-of-fit test (Lehmann & Romano, 2006), which provides an asymptotic control over the FPR.

TV Bootstrap: thresholds for $\mathcal{T}_h^{TV}$ are computed empirically by bootstrapping TR.

Three other methods built on different density models have been considered to assess the advantages – also in terms of HT power – of histograms providing uniform density.

Voronoi: a histogram where the $\{S_k\}_k$ are defined as Voronoi cells around K randomly chosen centers in TR.

Here we compute $\mathcal{T}_h^{TV}$ and use thresholds estimated by bootstrapping over TR.

Density Tree: A binary tree aiming at approximating ϕ_0, where splits are defined by a maximum information-gain criterion, in a similar fashion to random density trees like (Criminisi et al., 2011). We use $\mathcal{T}_h^{TV}$ with thresholds empirically computed by bootstrap over TR.

Parametric: in the synthetic experiments we consider also an HT based on a parametric density model. In particular, we fit a Gaussian density on TR, compute the log-likelihood (Song et al., 2007; Kuncheva, 2013) of each incoming batch W, and detect changes by means of the t-test. Since this method exploits the true density model, it has to be considered as an ideal reference.

All the methods are configured and tested on the same TR and tested on the same batches W. We perform a PCA transformation, estimated from TR, to all the methods based on trees as density models. We have in fact experienced that this improves the change-detection performance, since it aligns the coordinate axes – along which splits are performed – with the principal components that become parallel to the bin boundaries.

5.3. Test Design and Performance Measures

We consider a *small* TR configuration, where $N = 4096$ and $\nu = 64$, and a *large* TR configuration, where $N = 16384$ and $\nu = 256$. Both configurations have been tested with a number of bins $K = 32$ and $K = 128$, leading to 4 different combinations (N, ν, K). In all our experiments, the target FPR has been set to $\alpha = 0.05$.

We empirically compute the FPR as the ratio of detections over 100 stationary batches $W \sim \phi_0$, for each considered ϕ_0. Similarly, for each change $\phi_0 \to \phi_1$, we estimate the test power over 100 batches $W \sim \phi_1$. The average FPR and power computed over the whole datasets are reported as dots in Figure 2. To illustrate the distribution of the FPR and power we report their boxplots.

5.4. Results and Discussion

Figure 2 shows the FPR and the power of all the methods in the *small* TR configuration. Figures 2(a-b),(e-f) confirm that QuantTree effectively controls the FPR, for both the Pearson and total variation statistics, which is very important in change-detection. The peculiar QuantTree construction and Algorithm 2 provide very accurate thresholds resulting in FPR below the reference value $\alpha = 0.05$. Moreover, even if histograms defined by QuantTree feature a small number of bins, they are able to effectively monitor high-dimensional datastreams.

The FPRs of the total variation statistic are typically lower than others: this is due to the discrete nature of the statistics, which affects both testing and quantile estimation. The same problem occurs, but to a lesser extent, in the Pearson statistic, since the expression (6) contains a square that allows this statistic to assume a larger number of distinct values. Clearly, increasing K attenuates this problem, bringing the FPR closer to α. Thresholds used in the traditional Pearson test achieve larger FPR values, as the number of training samples in each bin is too low for the asymptotic approximation to hold: in the *large* TR configuration, the problem attenuates (plots and tables of average values are reported in the Appendix). Since the likelihood values do not follow a Gaussian distribution, the FPR are not properly controlled in the t-test of the Parametric method either. In all these tests, smaller values of K provide a better control over FPR, since the number of samples in each bin is larger.

Concerning the power, Figures 2(c-d) show a clear decay when d increases: this is consistent with the *Detectability loss problem*, which has been analytically studied in (Alippi et al., 2016) when monitoring the log-likelihood (as the Parametric). In general, all the methods on Synthetic datasets achieve satisfactory performance, and uniform histograms obtained through the QuantTree appear a better choice than Density Tree and Voronoi. There are minor differences among methods based on QuantTree which are nevertheless consistent with the FPR in Figure 2(a-b). Uniform density histograms outperforms others on real world datasets, see Figure 2(g-h), indicating that their partitioning scheme is better at detecting changes. Obviously, increasing N and ν provides superior performance (see the results reported in the supplementary materials).

6. Conclusions

In this paper we have presented QuantTree, an algorithm to build histograms for change detection through a recursive binary splitting of the input space. Our theoretical analysis allows a characterization of the probability of each bin defined by QuantTree and shows that this probability is independent from the distribution ϕ_0 of stationary data. This implies that statistics defined over such histograms are non parametric and thresholds can be estimated through numerical simulation on synthetically generated data. Experiments show that our thresholds (estimated using samples drawn from a univariate uniform distribution) enable a better control of the FPR than asymptotic ones or those estimated by bootstrap, which is no longer necessary when using such histograms. Ongoing work investigates how to mitigate the impact of test statistics assuming a limited number of discrete values, asymptotic results for histograms generated by QuantTree, and extensions to sequential monitoring schemes.

References

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

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

Balakrishnan, N. and Rao, C. R. *Order statistics: theory & methods*. Elsevier Amsterdam, 1998.

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

Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Bifet, A. and Gavalda, R. Learning from time-changing data with adaptive windowing. In *Proceedings of the SIAM International Conference on Data Mining*, volume 7, pp. 2007–2023, 2007.

Boracchi, G., Cervellera, C., and Macciò, D. Uniform histograms for change detection in multivariate data. In *Proceedings of the IEEE International Joint Conference of Neural Networks (IJCNN)*, pp. 1732–1739, 2017.

Carrera, D. and Boracchi, G. Generating high-dimensional datastreams for change detection. *Big Data Research*, 2017.

Criminisi, A., Shotton, J., and Konukoglu, E. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research*, 2011.

Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G. Calibrating probability with undersampling for unbalanced classification. In *Proceedings of the IEEE Symposium Series on Computational Intelligence and Data Mining (CIDM)*, pp. 159–166, 2015.

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

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

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–44, 2014.

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

Justel, A., Peña, D., and Zamar, R. A multivariate kolmogorov-smirnov test of goodness of fit. *Statistics & Probability Letters*, 35(3):251–259, 1997.

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, 2013.

Lehmann, E. L. and Romano, J. P. *Testing statistical hypotheses*. Springer, 2006.

Lichman, M. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Lung-Yut-Fong, A., Lévy-Leduc, C., and Cappé, O. Robust changepoint detection based on multivariate rank statistics. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3608–3611, 2011.

Page, E. S. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Qahtan, A. A., Alharbi, B., Wang, S., and Zhang, X. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 935–944, 2015.

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.

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

White, L. F., Bonetti, M., and Pagano, M. The choice of the number of bins for the m statistic. *Computational statistics & data analysis*, 53(10):3640–3649, 2009.