QuanTrees: Histograms for Monitoring Multivariate Data Streams

Giacomo Boracchi

giacomo.boracchi@polimi.it

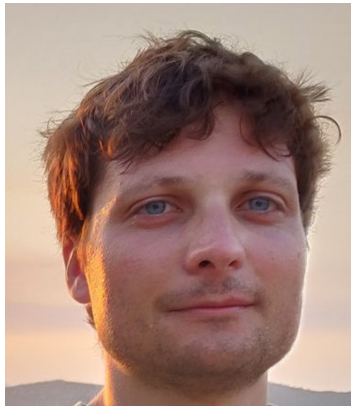
February 9th, 2023

University of Waikato, Hamilton, NZ

The “QuantTree Team”

Giacomo Boracchi

POLITECNICO MILANO 1863



Luca Frittoli

POLITECNICO MILANO 1863

Diego Stucchi

POLITECNICO MILANO 1863

Diego Carrera

POLITECNICO MILANO 1863

life.augmented

Cristiano Cervellera

Consiglio Nazionale delle Ricerche

Danilo Macciò

Consiglio Nazionale delle Ricerche

The Team

Faculties | PhD Students | Research Assistants


Giacomo Boracchi


Luca Magri


Federica Arrigoni


Filippo Leveni


Diego Stucchi


Loris Giulivi


Antonino Rizzo


Andrea Porfiri Dal Cin


Riccardo Margheritti


Michele Craighero


Edoardo Peretti


Giuseppe Bertolini

Our Research

Unsupervised Learning:

- Change and Anomaly Detection
- Domain Adaptation
- Learning in Nonstationary Environments

Computer Vision & Pattern Recognition:

- 3D vision
- Robust Model fitting
- Image Processing
- Image Analysis

Our Research

Unsupervised Learning:

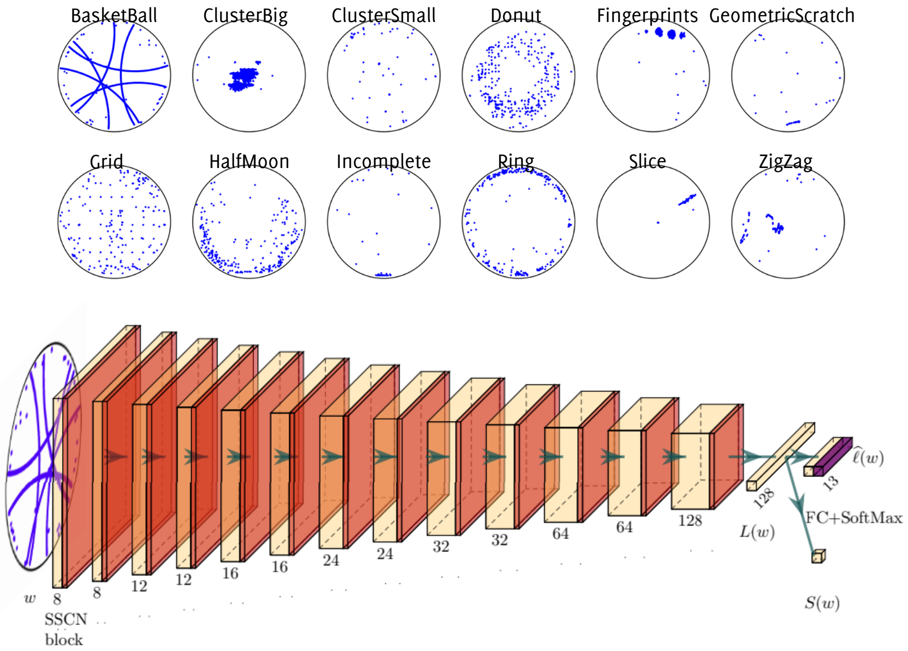
- Change and Anomaly Detection
- Domain Adaptation
- Learning in Nonstationary Environments

Computer Vision & Pattern Recognition:

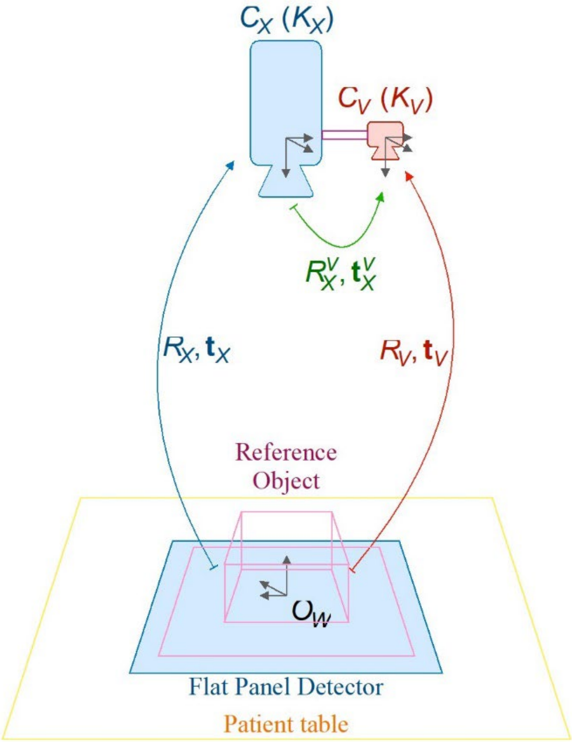
- 3D vision
- Robust Model fitting
- Image Processing
- Image Analysis

Industrial Projects

Explosives and Weapon Detection in Airport Inspection Systems



Wafer Defect Map Monitoring



Calibration of RX – RGB-D Systems

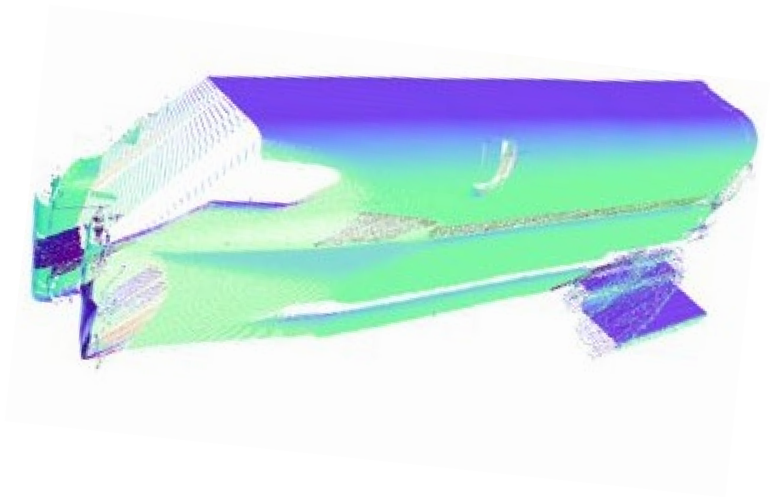
Giacomo Boracchi

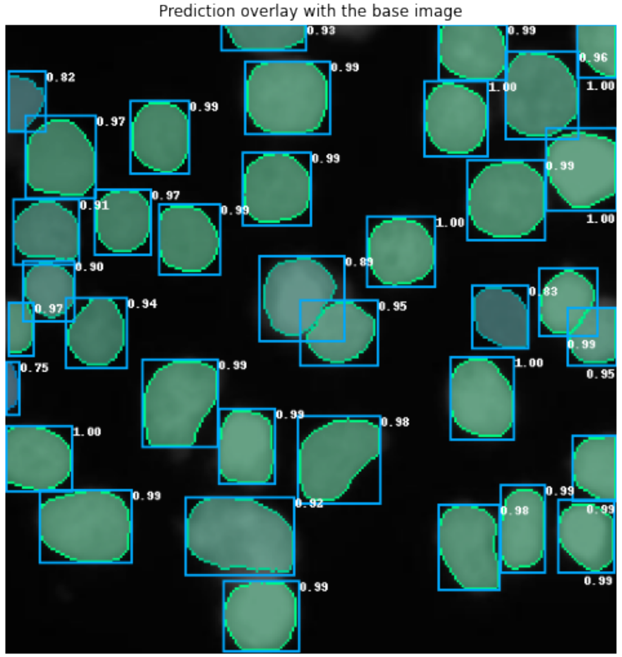
Industrial Projects

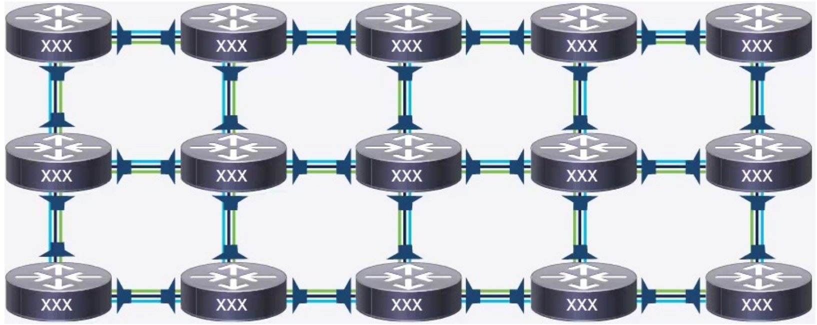
Meccanica del Sarca

ikonisys
FINDING THE CELLS THAT MATTER

cisco





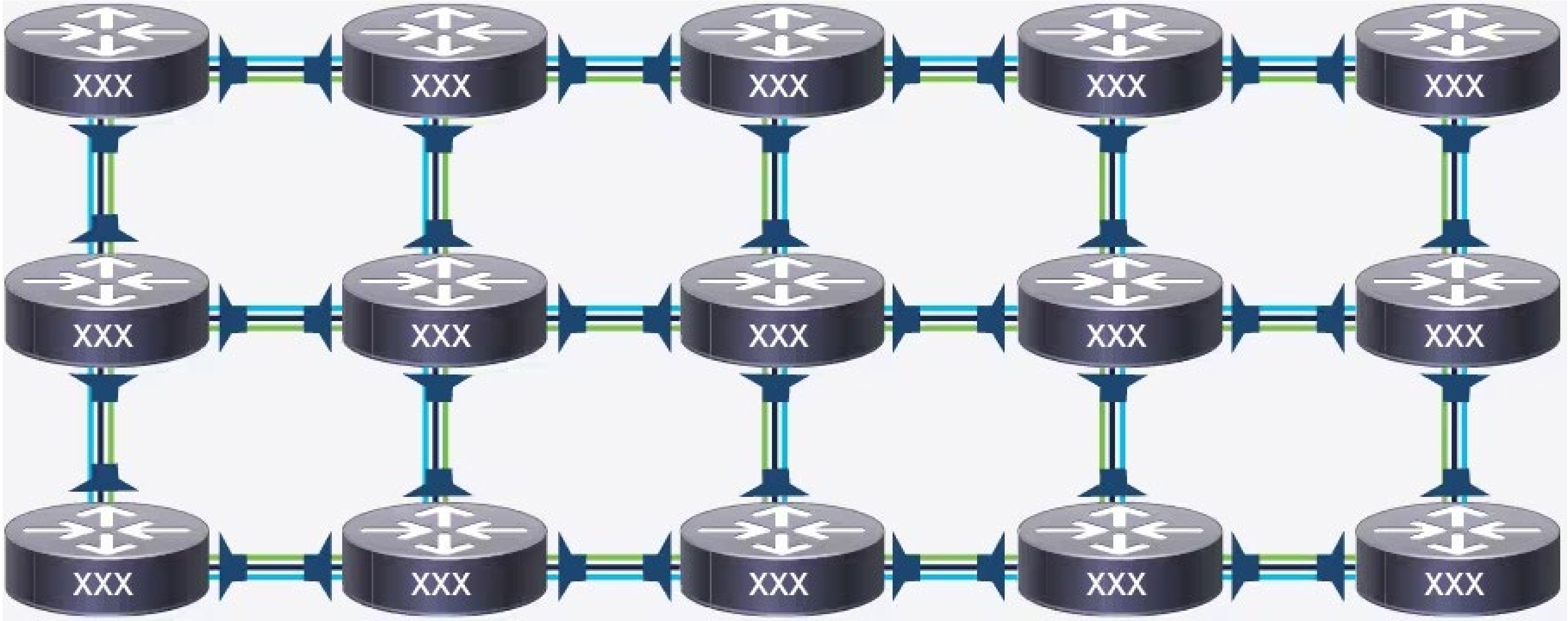


Defect Detection in 3D Scans

Automatic Cells Segmentation

Optical Networks Monitoring

Motivating Example: Routed Optical Networks

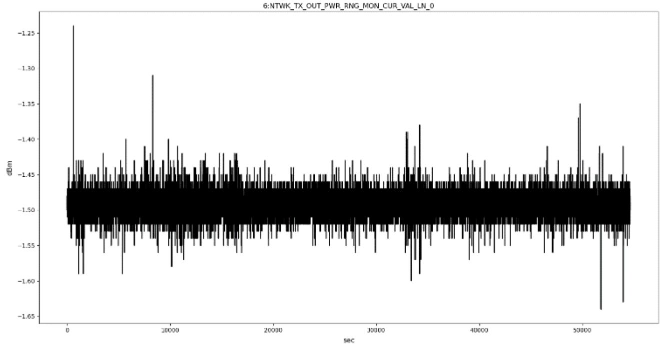


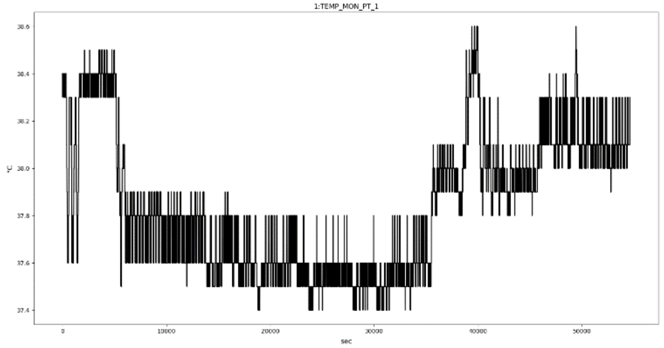
In collaboration with Cisco Photonics

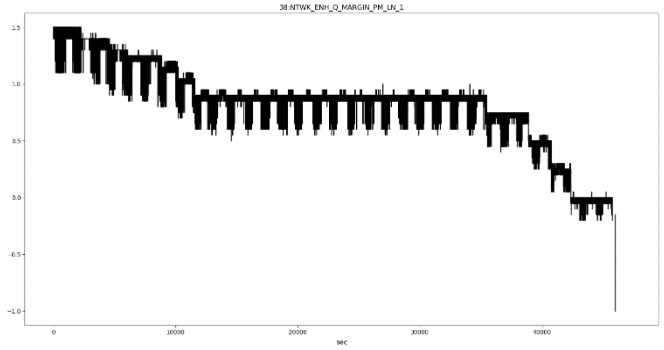
https://www.cisco.com/c/en/us/solutions/collateral/service-provider/routed-optical-networking/at-a-glance-c45-744217.html

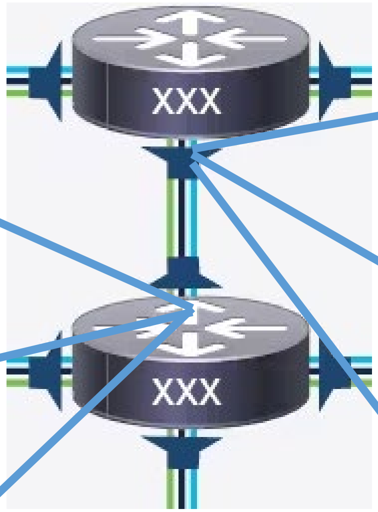
Routed Optical Networks

Datastreams @Router 1

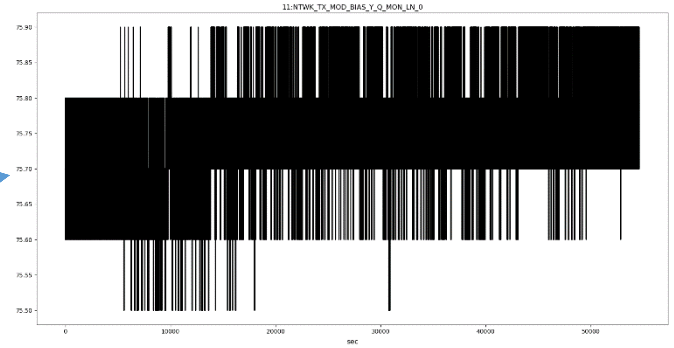


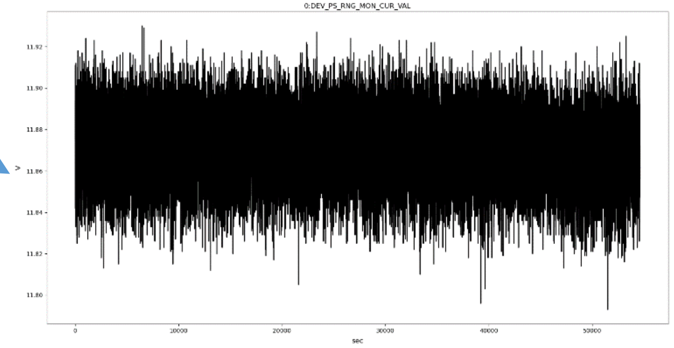


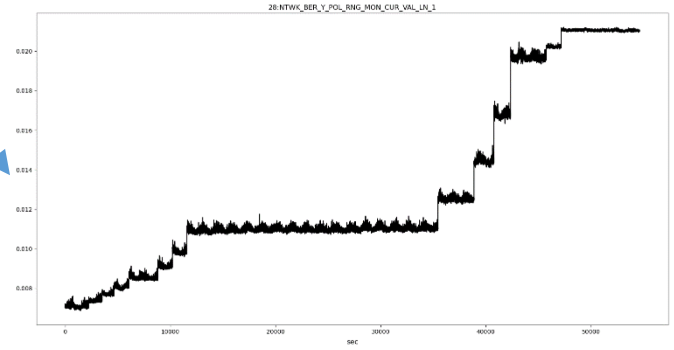




Datastreams @Router 2







In order to define the best routing strategies, each router needs to autonomously assess the transmission quality on each channel

https://www.cisco.com/c/en/us/solutions/collateral/service-provider/routed-optical-networking/at-a-glance-c45-744217.html

Problem Formulation

Change Detection in Data Streams…

…and often also in time series… as the problem boils down to this, once having computed independent residuals

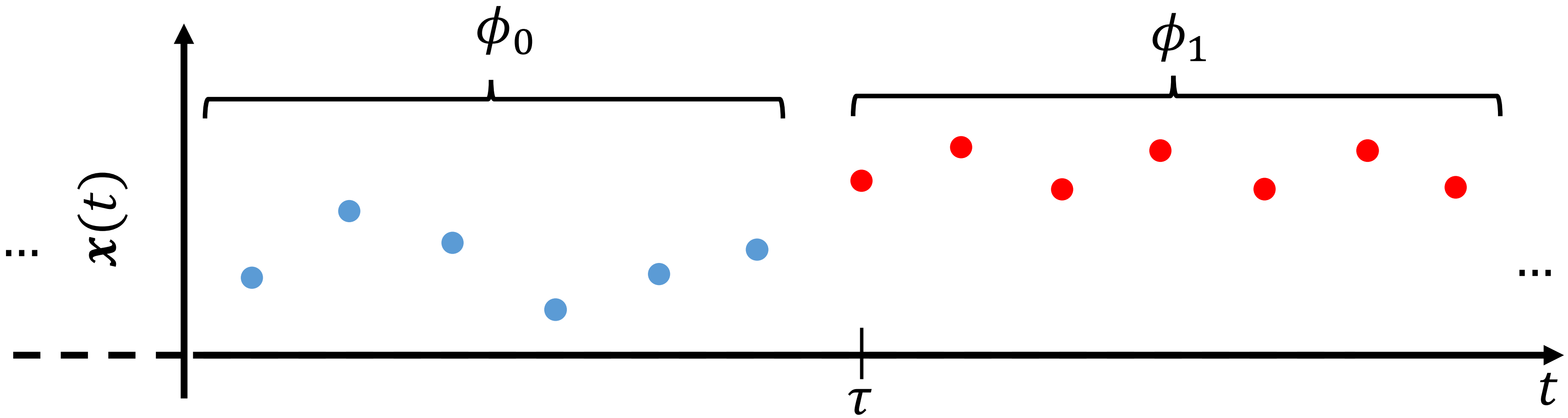
Change-Detection in a Statistical Framework

Monitor a **stream** $\{\boldsymbol{x}(t), t = 1, \dots\}$, $\boldsymbol{x}(t) \in \mathbb{R}^d$ of realizations of a **random variable**, and **detect the change-point** τ,

$$\boldsymbol{x}(t) \sim \begin{cases} \phi_0 & t < \tau \quad \text{in control state} \\ \phi_1 & t \geq \tau \quad \text{out of control state} \end{cases},$$

where $\{\boldsymbol{x}(t),\ t < \tau\}$ **are i.i.d.** and $\phi_0 \neq \phi_1$

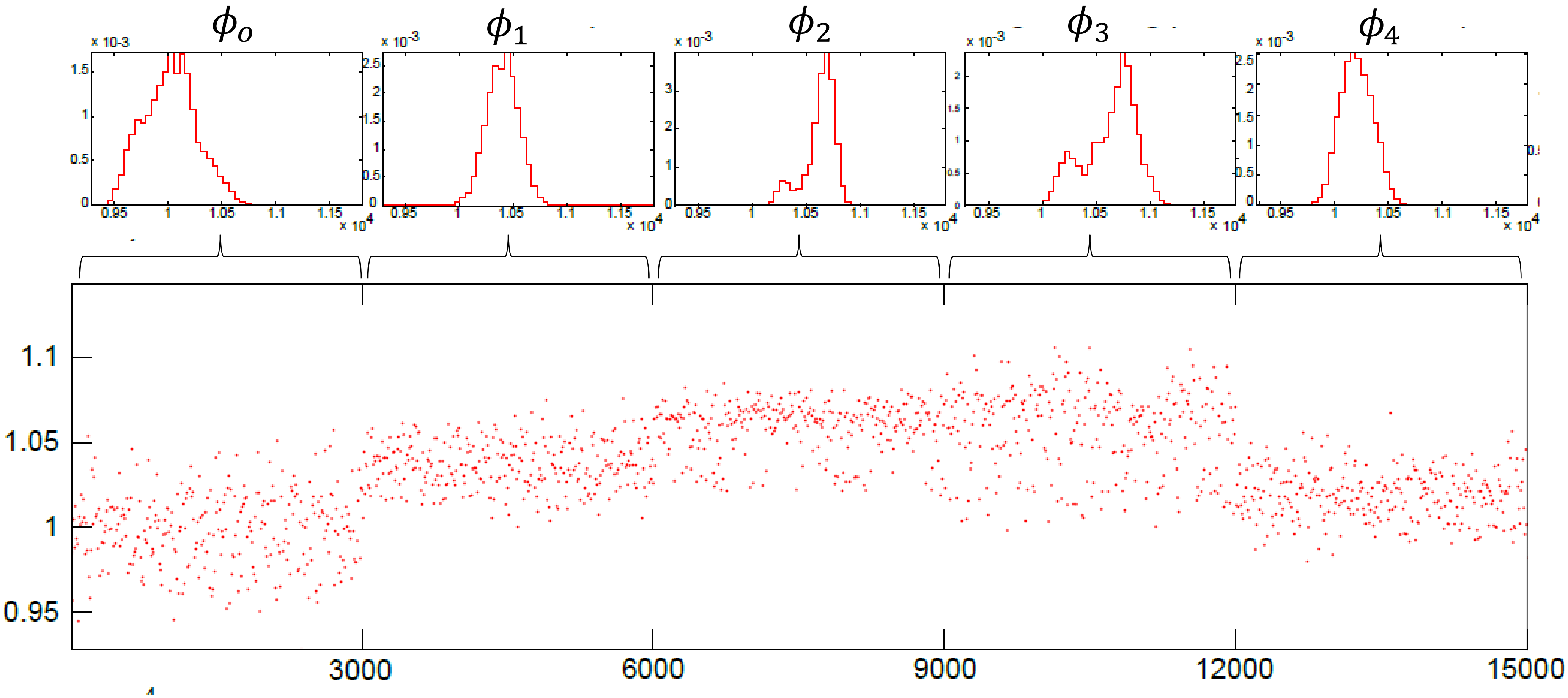
Typically, ϕ_1 is unknown and only $TR = \{\boldsymbol{x}(t) \sim \phi_0\}$ is given



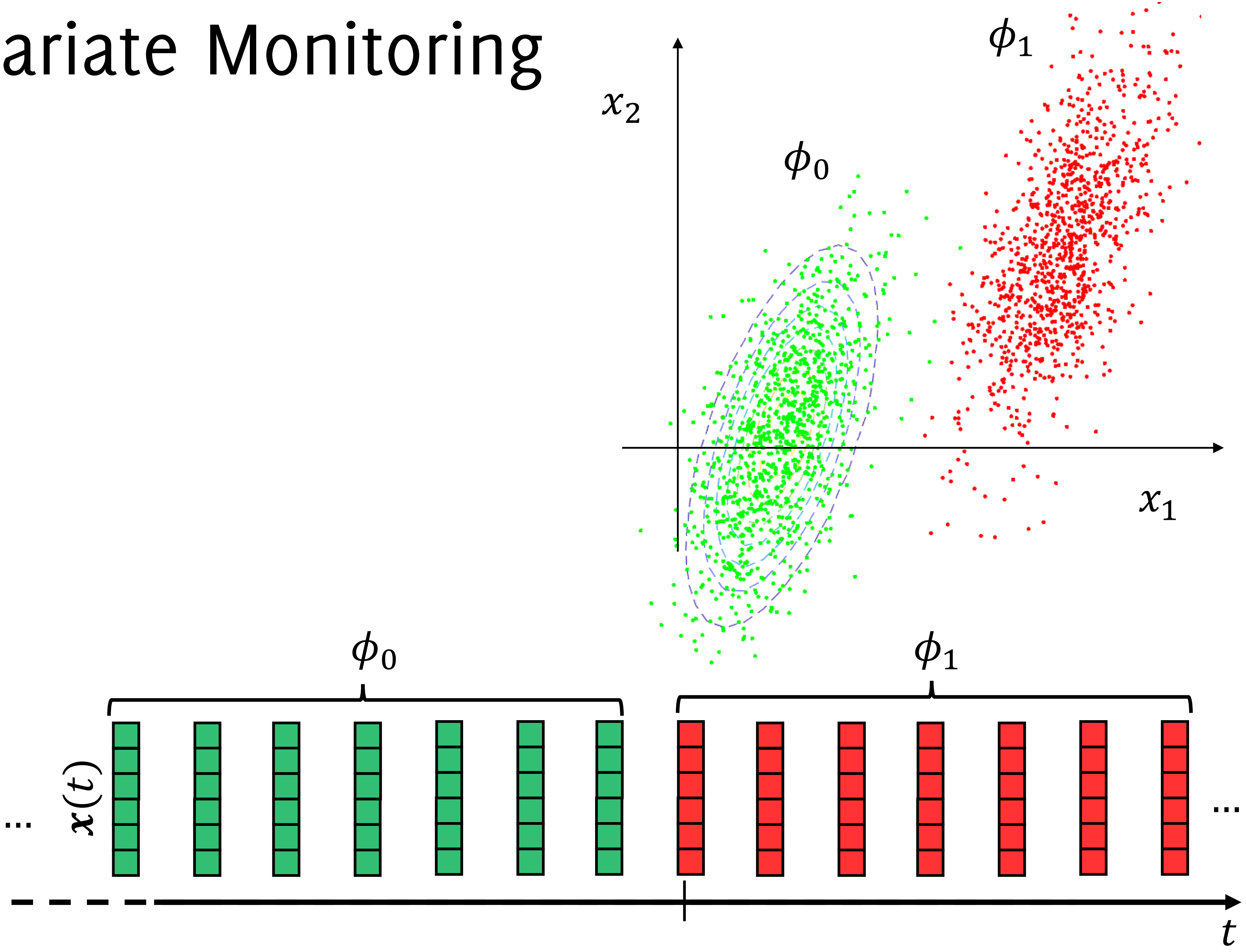
Change-Detection in a Statistical Framework

Here are data from an X-ray monitoring apparatus.

There are 4 changes $\phi_o \to \phi_1 \to \phi_2 \to \phi_3 \to \phi_4$ corresponding to different monitoring conditions and/or analyzed materials

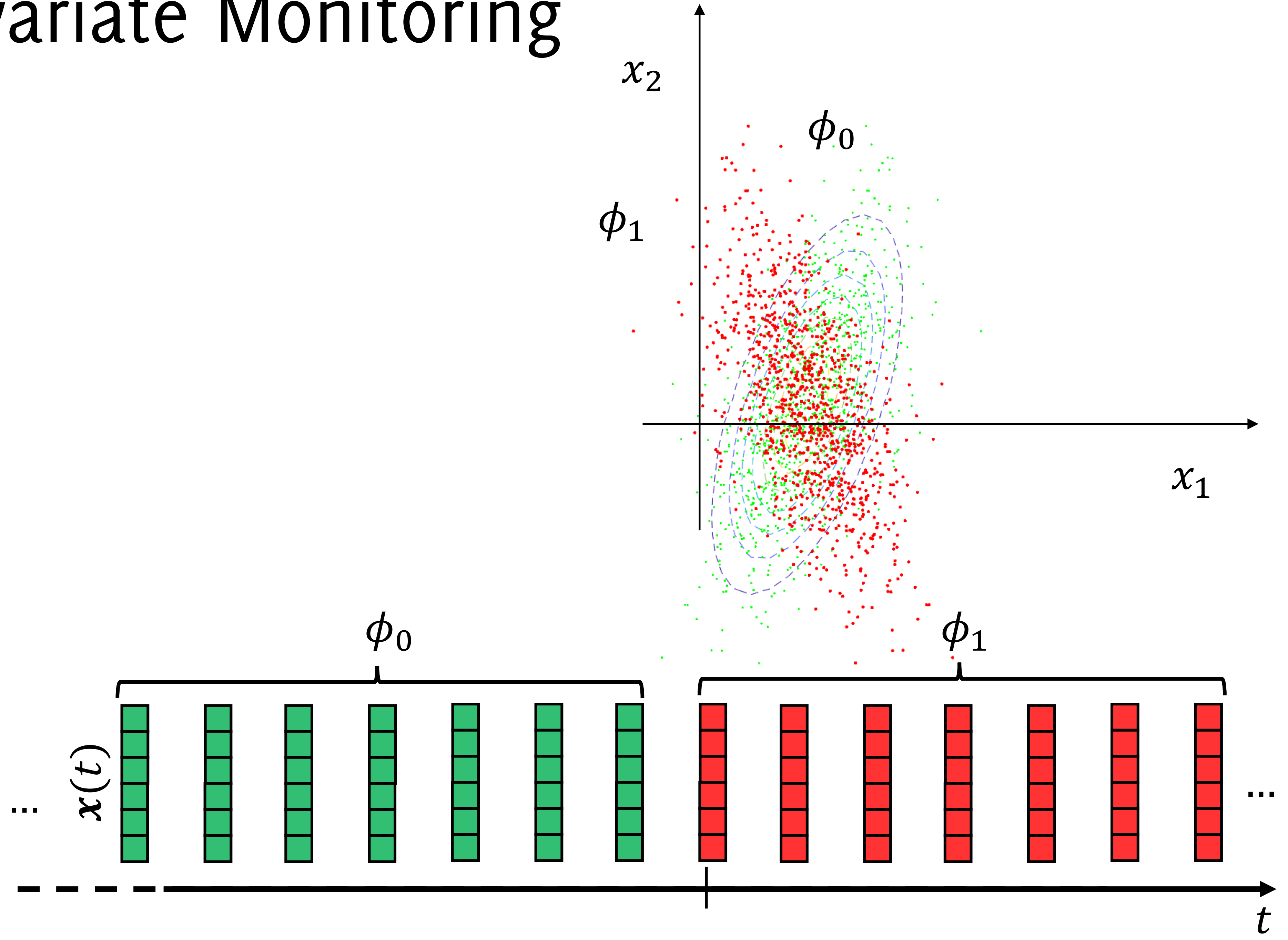


Multivariate Monitoring



Giacomo Boracchi

Multivariate Monitoring



Giacomo Boracchi

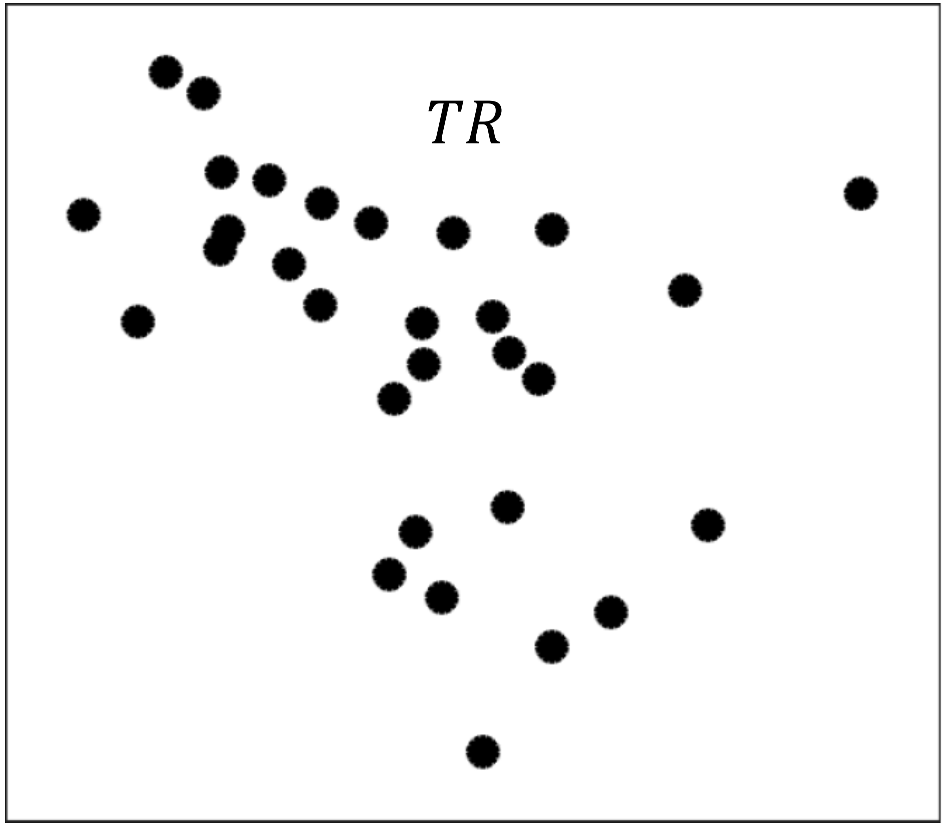
QuantTrees:
Histograms for Change Detection

A partitioning scheme specifically designed for change detection

G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

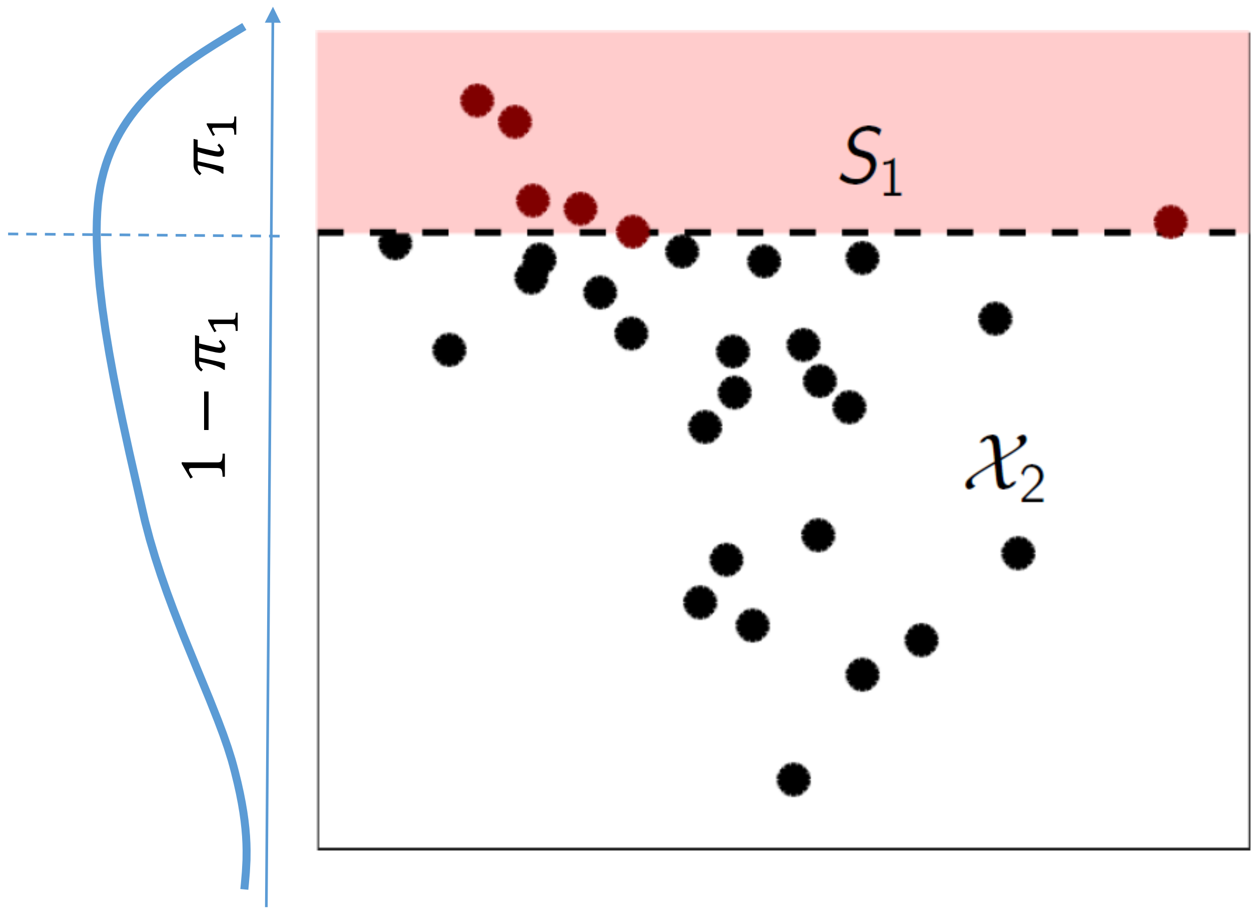
Assume you are given a set of target probabilities $\{\pi_i\}_{i=1,..,K}$ and a training set TR

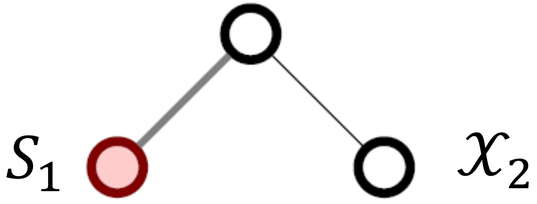


TR

QuantTrees: Histograms for change detection

Choose a dimension j at random, define the S_1 as the set containing the $1 - \pi_1$ quantile of the marginal distribution of training samples along j



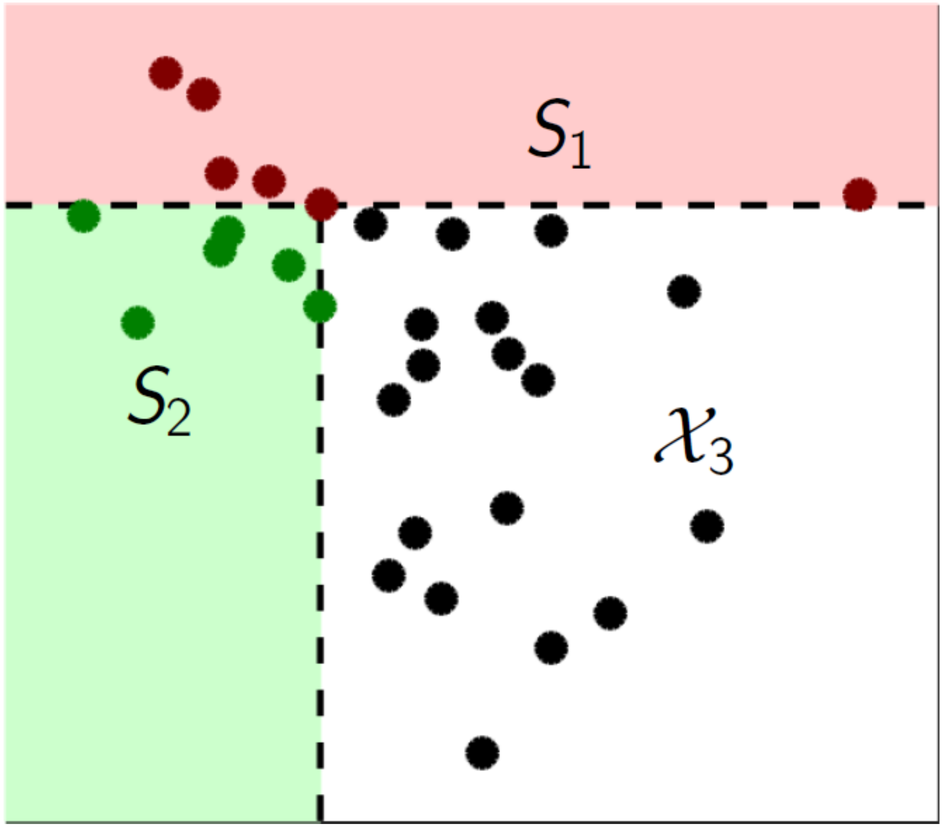


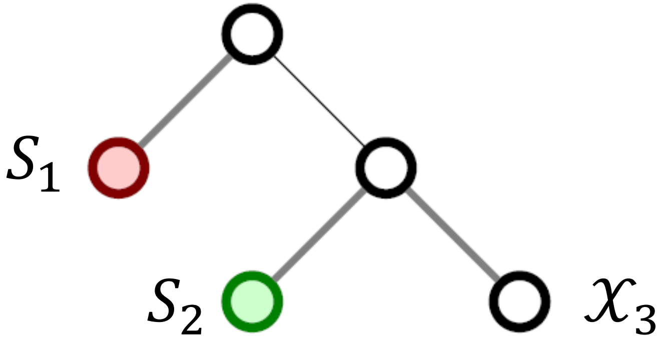
Call $\mathcal{X}_2$ the remaining samples

G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

The procedure is iterated on the training samples that have not been included in a bin.

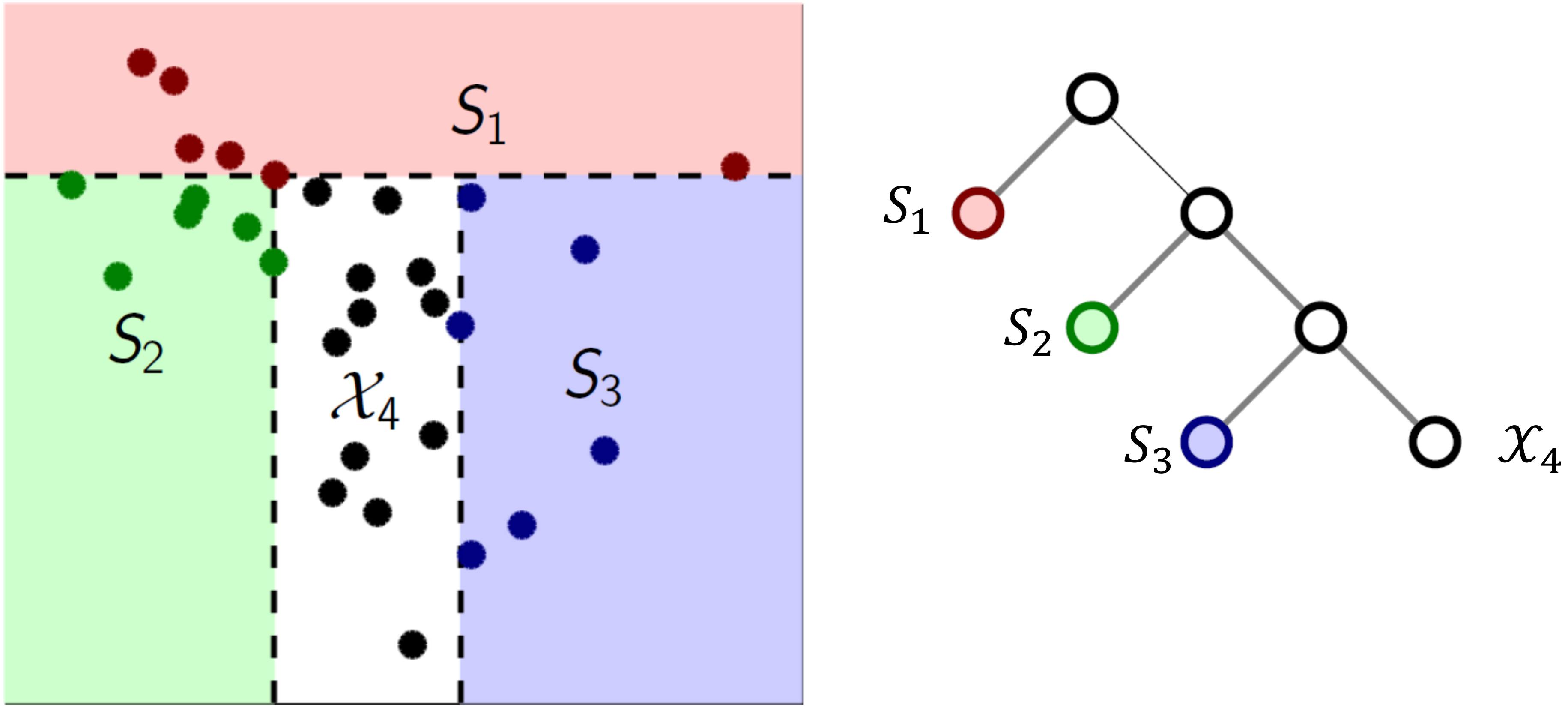




G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

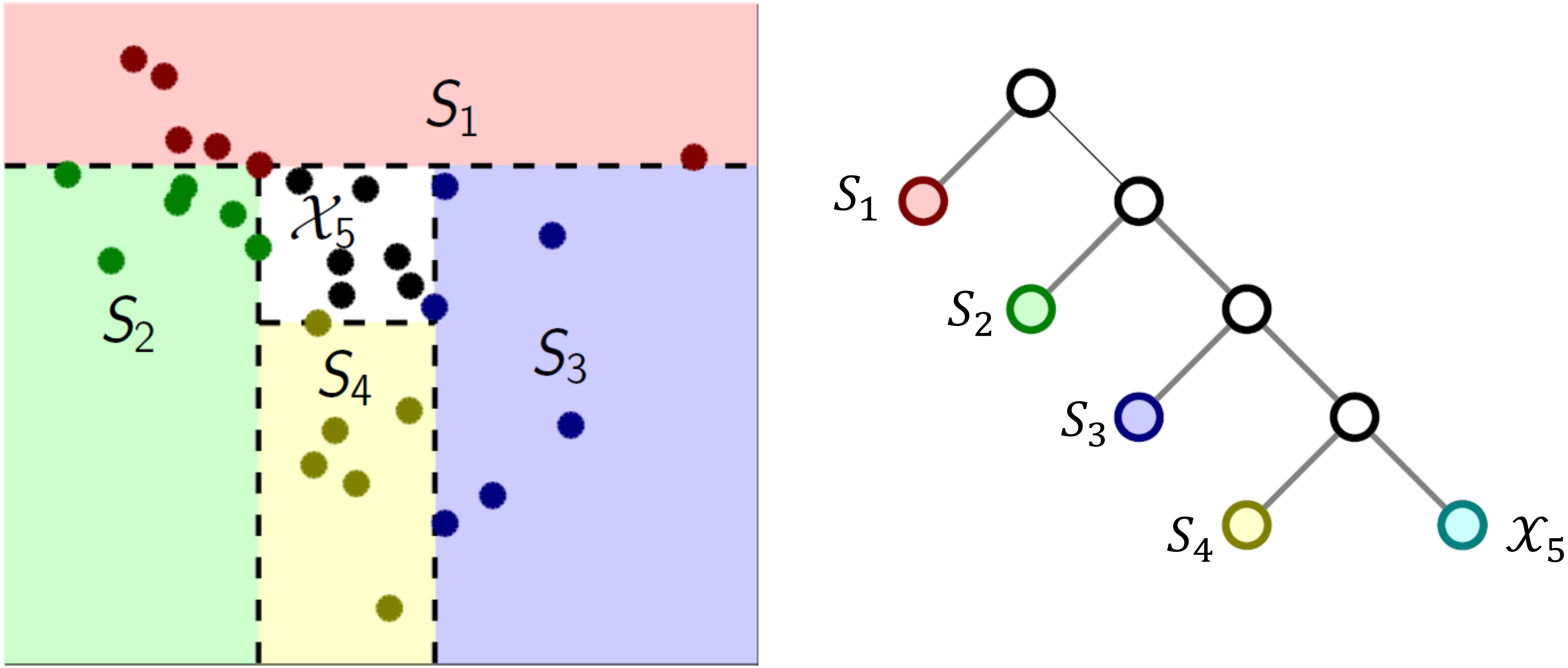
The procedure is iterated on the training samples that have not been included in a bin.



G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

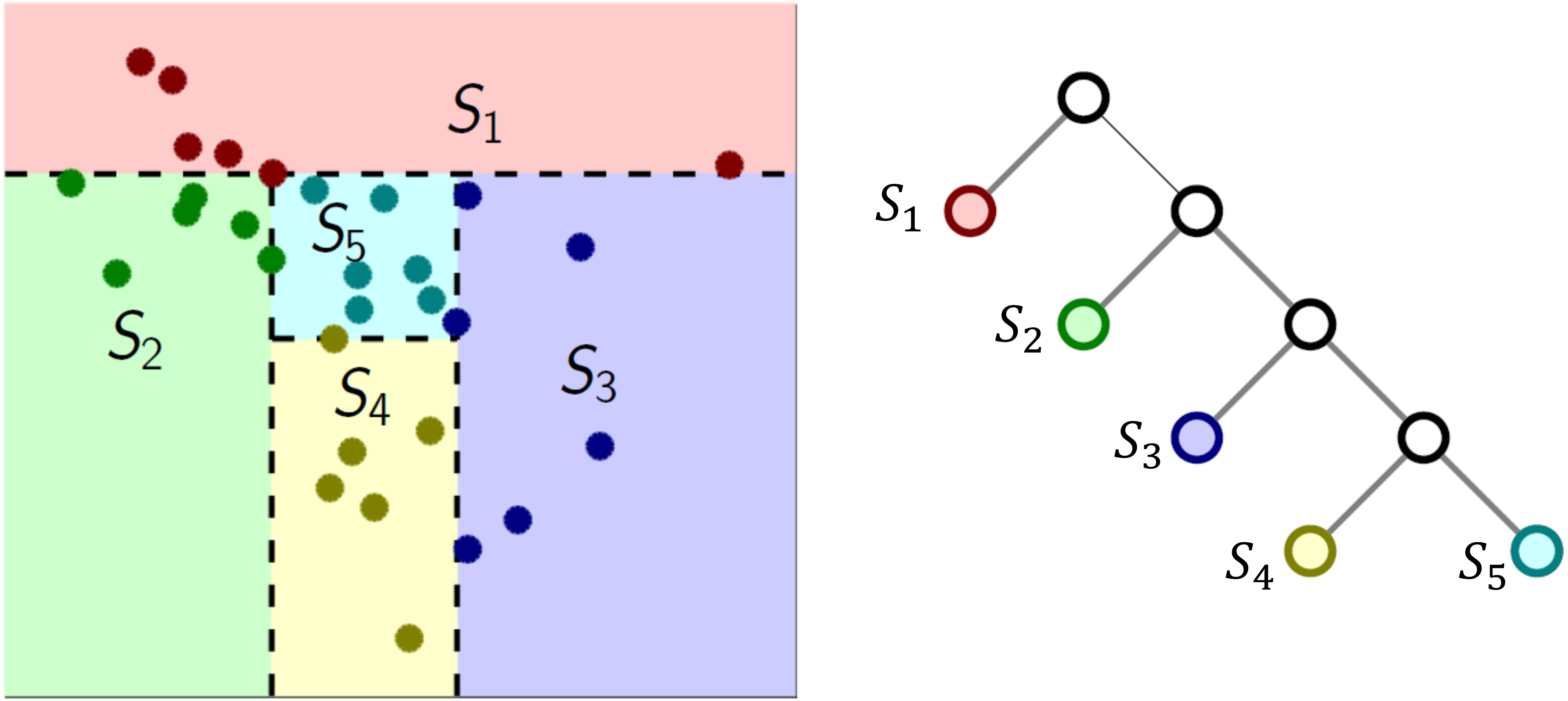
The procedure is iterated on the training samples that have not been included in a bin.



G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

The procedure is iterated on the training samples that have not been included in a bin.



G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTree Construction

QuantTree **iteratively divides** the input space by **binary splits along a single covariate**, where the cutting points are defined by the **quantiles of the marginal distributions**

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 $h = \{(S_k, \widehat{\pi}_k)\}_k$.

Set $N_0 = N$, $L_0 = 0$.
for $k = 1, \dots, K$ **do**
  Set $N_k = N_{k-1} - L_{k-1}$, $\mathcal{X}_k = \mathcal{X} \setminus \bigcup_{j<k} S_j$, and $L_k = \text{round}(\pi^k N)$.
  Choose a random component $i \in \{1, \dots, d\}$.
  Define $z_n = [\mathbf{x}_n]_i$ for each $\mathbf{x}_n \in \mathcal{X}_k$.
  Sort $\{z_n\}$: $z_{(1)} \leq z_{(2)} \leq \dots z_{(N_k)}$.
  Draw $\gamma \in \{0, 1\}$ from a Bernoulli(0.5).
  if $\gamma = 0$ **then**
    Define $S_k = \{\mathbf{x} \in \mathcal{X}_k \quad [\mathbf{x}]_i \leq z_{(L_k)}\}$.
  else
    Define $S_k = \{\mathbf{x} \in \mathcal{X}_k \quad [\mathbf{x}]_i \geq z_{(N_k - L_k + 1)}\}$.
  end if
  Set $\widehat{\pi}_k = L_k / N$.
end for

G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTree Construction

QuantTree **iteratively divides** the input space by **binary splits along a single covariate**, where the cutting points are defined by the **quantiles of the marginal distributions**

The QuantTree construction is randomized by the random selection of the component for each split and whether to take the π_i or $1-\pi_i$ quantile

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 $h = \{(S_k, \widehat{\pi}_k)\}_k$.

```
Set N_0 = N, L_0 = 0.
for k = 1, ..., K do
    Set N_k = N_{k-1} - L_{k-1}, X_k = X \ ∪_{j<k} S_j, and
    L_k = round(π^k N).
    Choose a random component i ∈ {1, ..., d}.
    Define z_n = [x_n]_i for each x_n ∈ X_k.
    Sort {z_n}: z_(1) ≤ z_(2) ≤ ... z_(N_k).
    Draw γ ∈ {0, 1} from a Bernoulli(0.5).
    if γ = 0 then
        Define S_k = {x ∈ X_k   [x]_i ≤ z_(L_k)}.
    else
        Define S_k = {x ∈ X_k   [x]_i ≥ z_(N_k - L_k + 1)}.
    end if
    Set π̂_k = L_k / N.
end for
```

Set $N_0 = N$, $L_0 = 0$.
for $k = 1, \dots, K$ **do**
  Set $N_k = N_{k-1} - L_{k-1}$, $\mathcal{X}_k = \mathcal{X} \setminus \bigcup_{j<k} S_j$, and $L_k = \mathrm{round}(\pi^k N)$.
  Choose a random component $i \in \{1, \dots, d\}$.
  Define $z_n = [\mathbf{x}_n]_i$ for each $\mathbf{x}_n \in \mathcal{X}_k$.
  Sort $\{z_n\}$: $z_{(1)} \leq z_{(2)} \leq \dots z_{(N_k)}$.
  Draw $\gamma \in \{0, 1\}$ from a Bernoulli(0.5).
  if $\gamma = 0$ **then**
    Define $S_k = \{\mathbf{x} \in \mathcal{X}_k \quad [\mathbf{x}]_i \leq z_{(L_k)}\}$.
  else
    Define $S_k = \{\mathbf{x} \in \mathcal{X}_k \quad [\mathbf{x}]_i \geq z_{(N_k - L_k + 1)}\}$.
  end if
  Set $\widehat{\pi}_k = L_k / N$.
end for

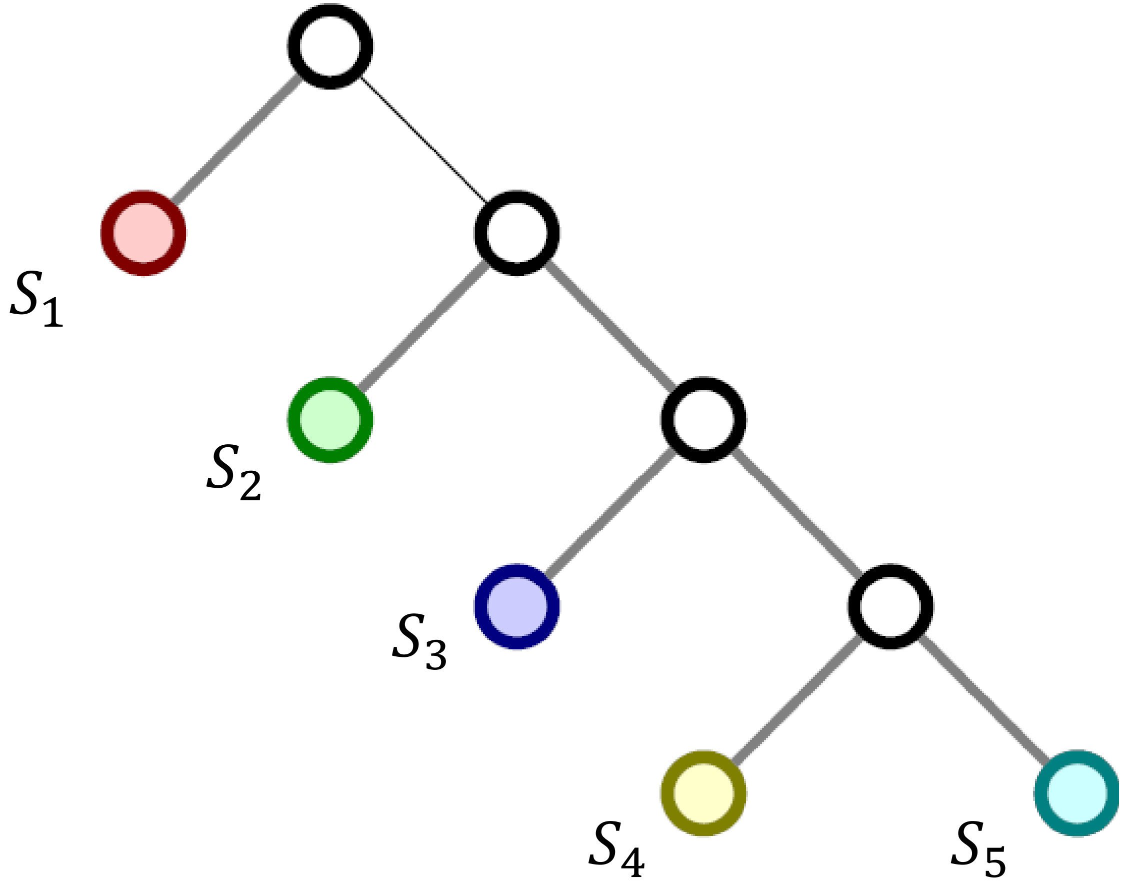
G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTree Partitioning

Each QuantTree is associated with a partitioning of the input domain

$$\{S_k, \hat{\pi}_k\}$$

Where $\hat{\pi}_k$ are the probabilities estimated from TR, can slightly depart from the target $\{\pi_k\}$ (they match when $\pi_k N$ is an integer)



Change Detection By QuantTrees

Batch-wise change detection

1. Monitor a batch of ν test samples

$$W = \{x(t), \dots, x(t+\nu)\}$$

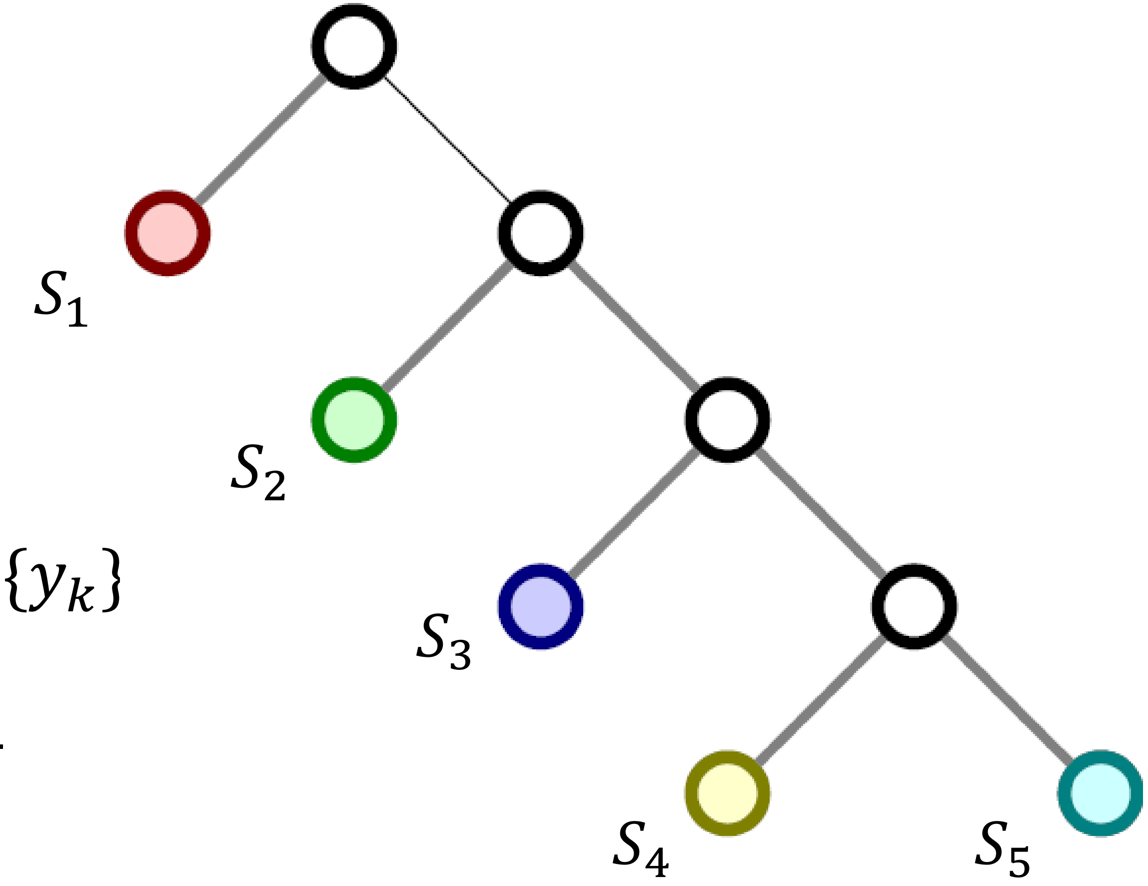
2. Dispatch samples in bins $\{S_k\}$ and compute the number of samples in each bin $\{y_k\}$

3. Compute **any test statistic** depending on $\{y_k\}$

$$e.g., \qquad \mathcal{T}_h(W) = \sum_{k=1}^{K} \frac{(y_k - \nu\pi_k)^2}{\nu\pi_k}$$

4. Compare it against a threshold γ

$$\mathcal{T}_h(W) > \gamma$$



G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees Statistics

Theorem (ICML18)

Let $T_h(\cdot)$ be a statistic defined over the bin probabilities of a histogram h computed by QuantTree.

For any stationary batch $W \sim \phi_0$, the distribution of $T_h(W)$ depends only on:

- *the number of training samples $N = \#TR$,*
- *the batch size W ,*
- *the expected probabilities in each bin $\{\pi_i\}_{i=1,\ldots,K}$*

G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

Implications

In histograms constructed by QuantTrees, test statistics do not depend on ϕ_0, nor data dimension d.

Detection threshold γ can be numerically computed from synthetic data:

1. Generating data according to a 1D ψ_0 (e.g., ψ_0 is uniform $[0,1]$)
2. Define a QT histogram $h = \{S_k, \pi_k\}$ on TR
3. Generate stationary test batches $W \sim \psi_0$, the test statistic
4. Compute the threshold γ from the empirical distribution of $T_h(W)$

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

Example of Thresholds γ

G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTree Statistics

Theorem (TKDE22)

Let $h = \{S_k, \pi_k\}$ *be a partitioning of the input domain in* K *bins built using the QuantTree algorithm with target probabilities* $\{\pi_k\}_{k=1,\dots,K}$.

Let p_k *be the expected probability of* S_k *under* ϕ_0, *namely* $p_k = P_{\phi_0}(S_k)$.

Then, the probabilities $(p_1, \dots, p_K)$ *follow a Dirichlet distribution*

$$(p_1, \dots, p_K) \sim D\left(\pi_1 N, \pi_2 N, \dots, \left(1 - \sum_{j=1}^{K-1} \pi_j\right) N + 1\right)$$

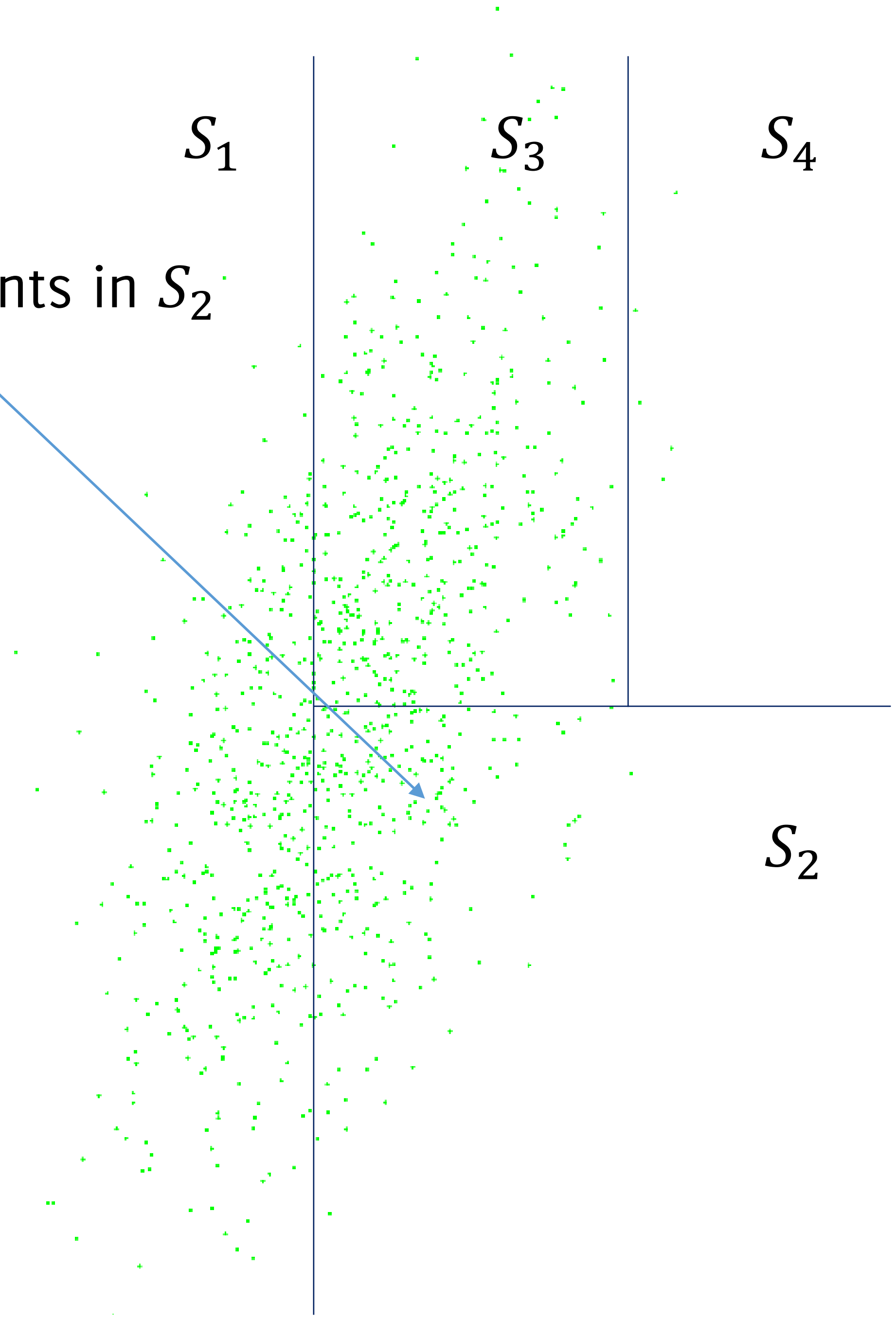
L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

Differences between π_k and p_k

π_k and $\hat{\pi}_k$ represent the empirical frequency of points in the bin S_k. Sometimes they do coincide (often we assume they do)

These are used to construct the QuantTree histogram, but might not corresponds to the true bin probabilities

$\hat{\pi}_2 = \#$ training points in S_2

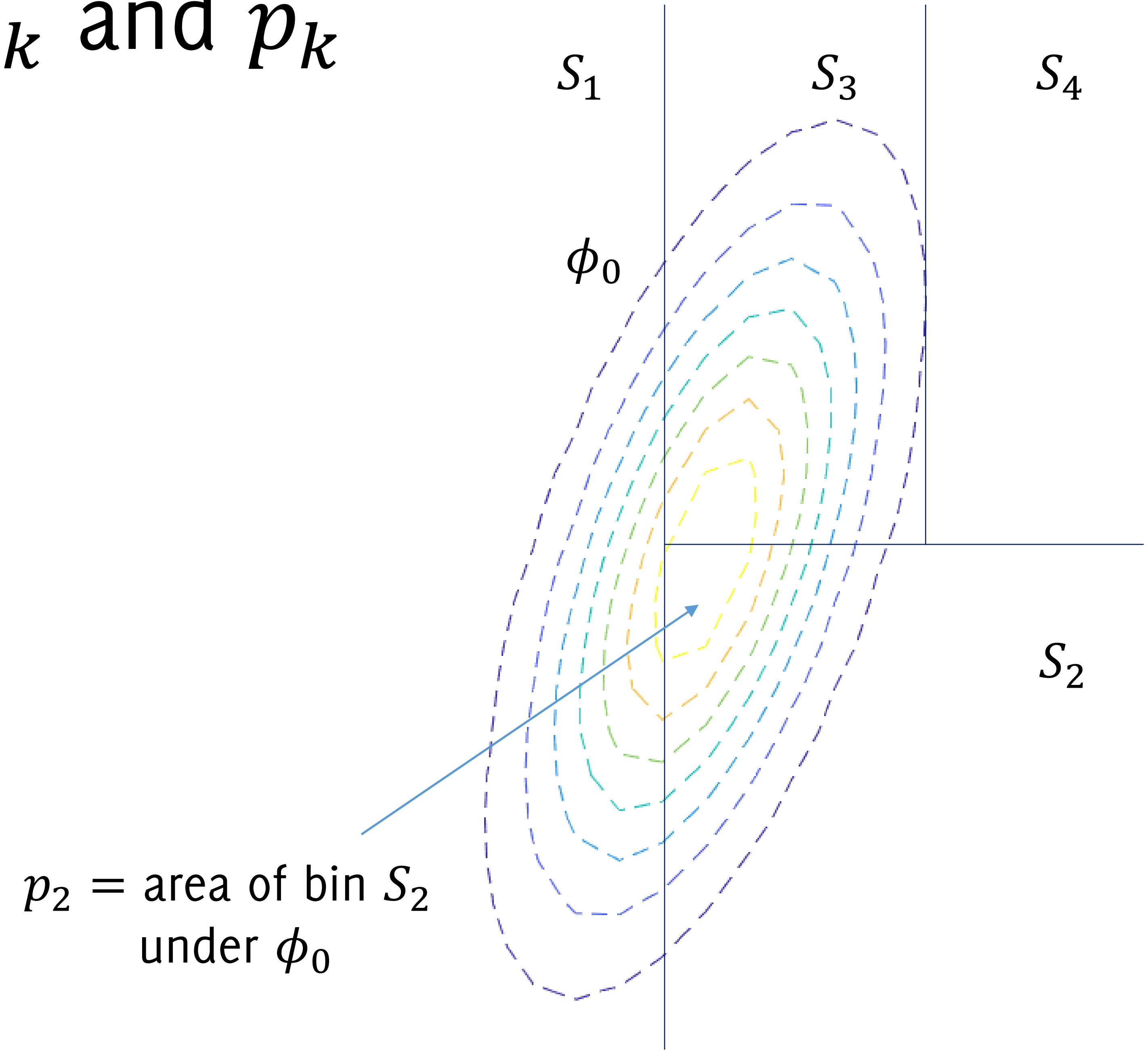


Differences between π_k and p_k

p_k is the true bin probability, namely the area of the bin S_k under the unknown probability density function ϕ_0

Given a batch W, the number of points falling in the bin $\{y_k\}$ is a realization of a multinomial distribution

$$\mathcal{M}(p_1, \dots, p_K, \nu, K)$$



Implications

In histograms constructed by QuantTrees, the bin probabilities do not depend on ϕ_0, nor data dimension d.

Detection threshold γ can be numerically computed from synthetic data:

1. Draw the expected bin probabilities $(p_1, \dots, p_K)$ from the Dirichlet before
2. Draw the number of samples $(y_1, \dots, y_K)$ falling in each bin from a multinomial distribution having parameters $(p_1, \dots, p_K)$
$$(y_1, \dots, y_K) \sim \mathcal{M}(p_1, \dots, p_K, \nu, K)$$
3. Compute the values of test statistics $T_h(\cdot)$
4. Compute the threshold γ from the empirical distribution of $T_h(\cdot)$

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

Change Detection By QuantTrees

Training:

- Define a QT $h = \{S_k, \hat{\pi}_k\}$ from TR with target probabilities $\{\pi_i\}_{i=1,\dots,K}$
- Compute threshold γ on synthetic data using $\{\hat{\pi}_k\}_{i=1,\dots,K}$, ν, $N = \#TR$

Testing

- Gather a batch of test samples W
- Compute the test statistic

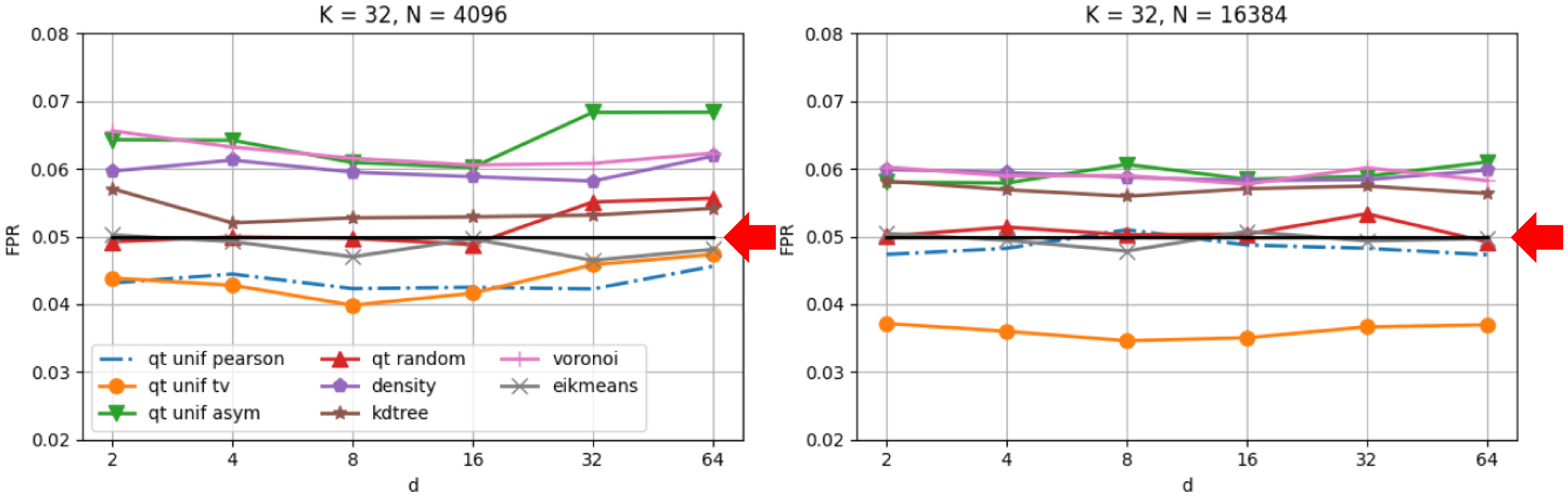
$$\mathcal{T}_h(W) = \sum_{k=1}^{K} \frac{(y_k - \nu\pi_k)^2}{\nu\pi_k}$$

- Detect a change when $\mathcal{T}_h(W) > \gamma$

G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

Experiments on False Positive Control

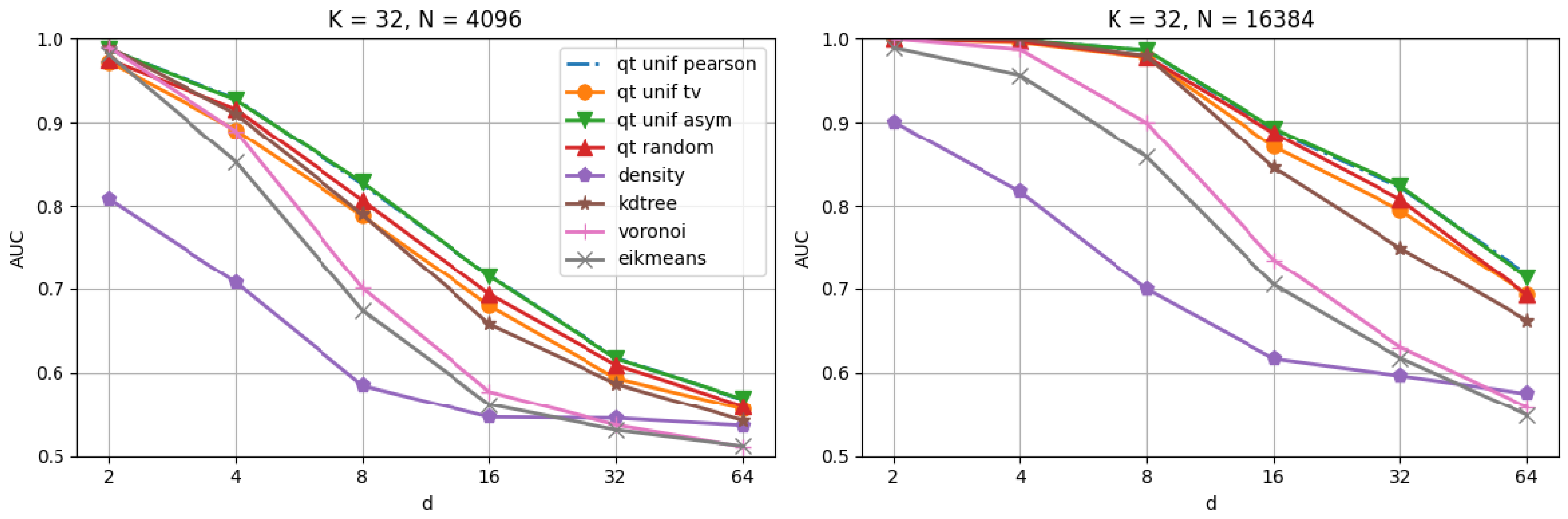
QT algorithms can control FPR (target $\alpha = 0.05$) without resorting to bootstrap and better than asymptotic approximation



Test on synthetic data ϕ_0 is a Gaussian. High dispersion in statitsics from random bin probabilities $\{\pi_k\}$

Experiments on Detection Power (AUC)

- QT with Pearson Statistics are among the most powerful CD algorithms
- Uniform bin probabilities $\pi_k = 1/K$ are better than random probabilities



Test on synthetic data such as $sKL(\phi_0, \phi_1) = 1$

Experiments on Real World Datasets

dataset		qt unif pearson	qt unif asym	qt unif tv	kdtree	voronoi	density	qt random	eikmeans
particle	FPR	0.042	0.065	0.044	0.053	0.063	0.057	0.054	0.049
	AUC	0.876	**0.886**	0.865	0.841	0.530	0.529	0.842	0.512
protein	FPR	0.046	0.064	0.046	0.055	0.065	0.059	0.050	0.047
	AUC	**0.978**	**0.978**	0.972	0.969	0.564	0.591	0.962	0.527
credit	FPR	0.045	0.064	0.046	0.051	0.060	0.061	0.054	0.049
	AUC	0.800	**0.810**	0.781	0.788	0.532	0.721	0.753	0.515
sensorless	FPR	0.043	0.063	0.044	0.053	0.058	0.059	0.055	0.050
	AUC	**1.000**	**1.000**	**1.000**	**1.000**	0.517	0.627	**1.000**	0.503
nino	FPR	0.041	0.063	0.042	0.053	0.064	0.058	0.050	0.047
	AUC	**0.833**	0.825	0.811	0.819	0.558	0.546	0.802	0.543
spruce	FPR	0.042	0.067	0.041	0.056	0.065	0.058	0.052	0.050
	AUC	**1.000**	**1.000**	**1.000**	**1.000**	0.560	**1.000**	**1.000**	0.509
lodgepole	FPR	0.043	0.061	0.045	0.053	0.066	0.062	0.052	0.051
	AUC	**1.000**	**1.000**	**1.000**	**1.000**	0.580	**1.000**	**1.000**	0.517
insects	FPR	0.042	0.063	0.043	0.052	0.062	0.058	0.051	0.049
	AUC	0.912	0.910	0.892	0.854	0.897	**0.994**	0.877	0.854

Table 2: Results for the QuantTree algorithm on real datasets for $N = 4096$, $K = 32$. For each dataset, the FPR and AUC are repoted, averaged over 100 runs for each method.

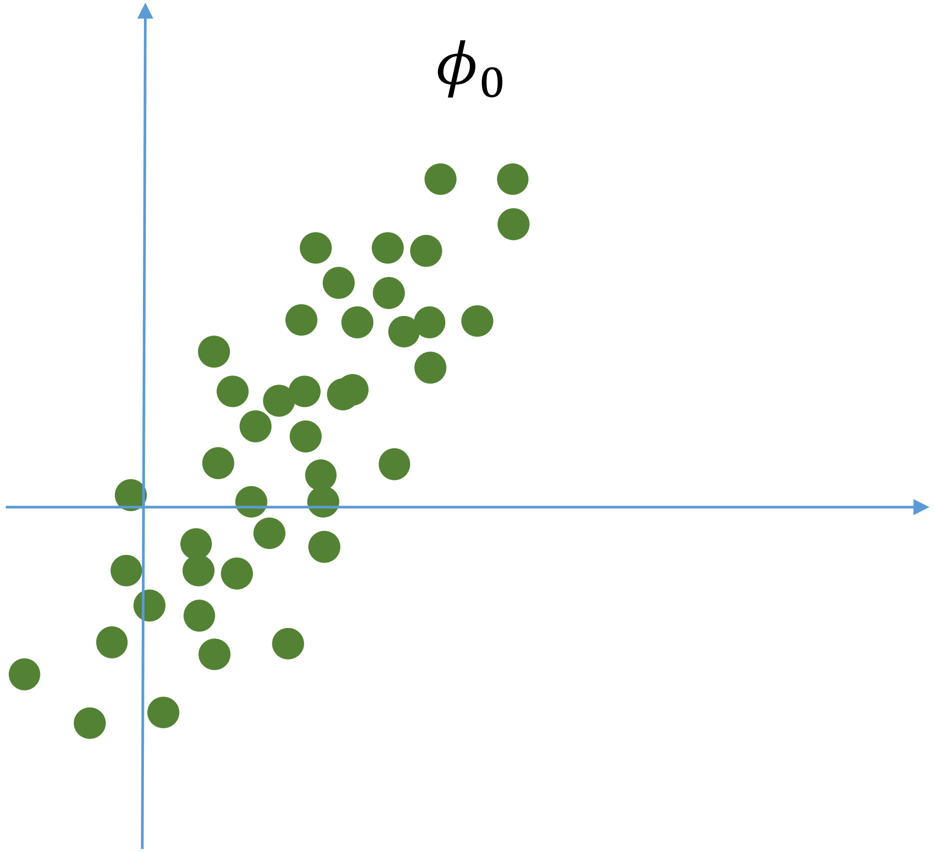
Advantages of QT

Provide a truly **multivariate** monitoring scheme that:

- enables change detection in a **nonparametric manner** (no assumption on ϕ_0), possibly in high dimensional data d
- guarantees a control over the false positives for any statistic $\mathcal{T}_h(W)$
- it does not require many training data TR (while alternatives based on bootstrap do)
- it is rather efficient to use, compared to other schemes

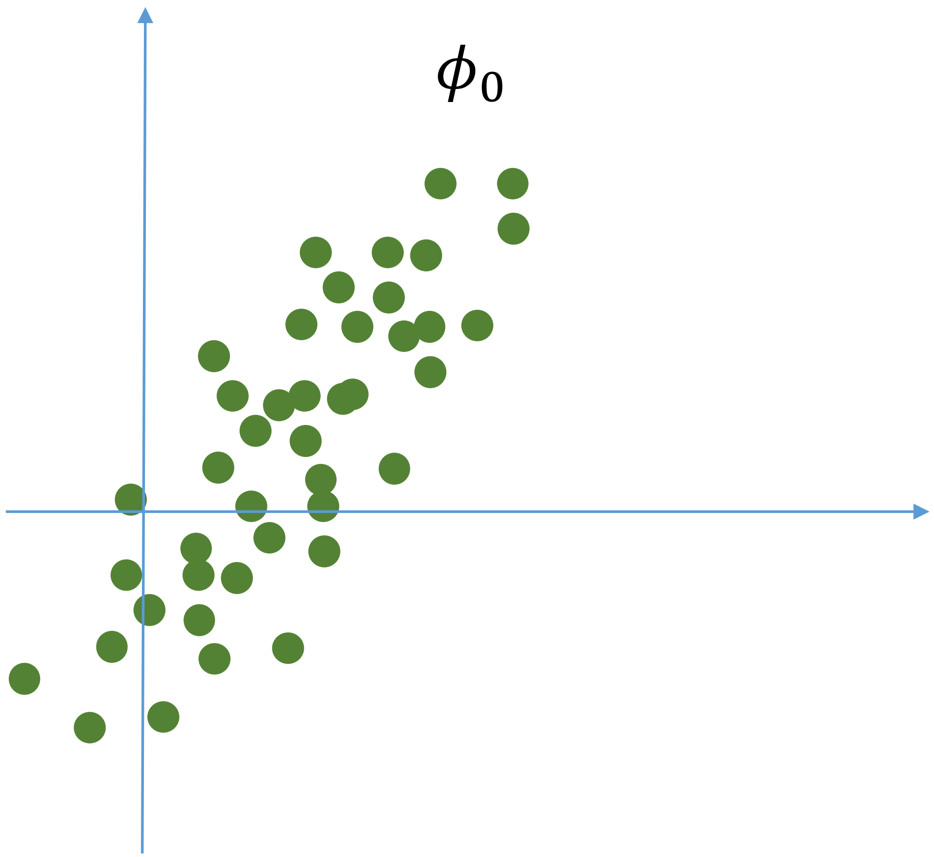
Limitations

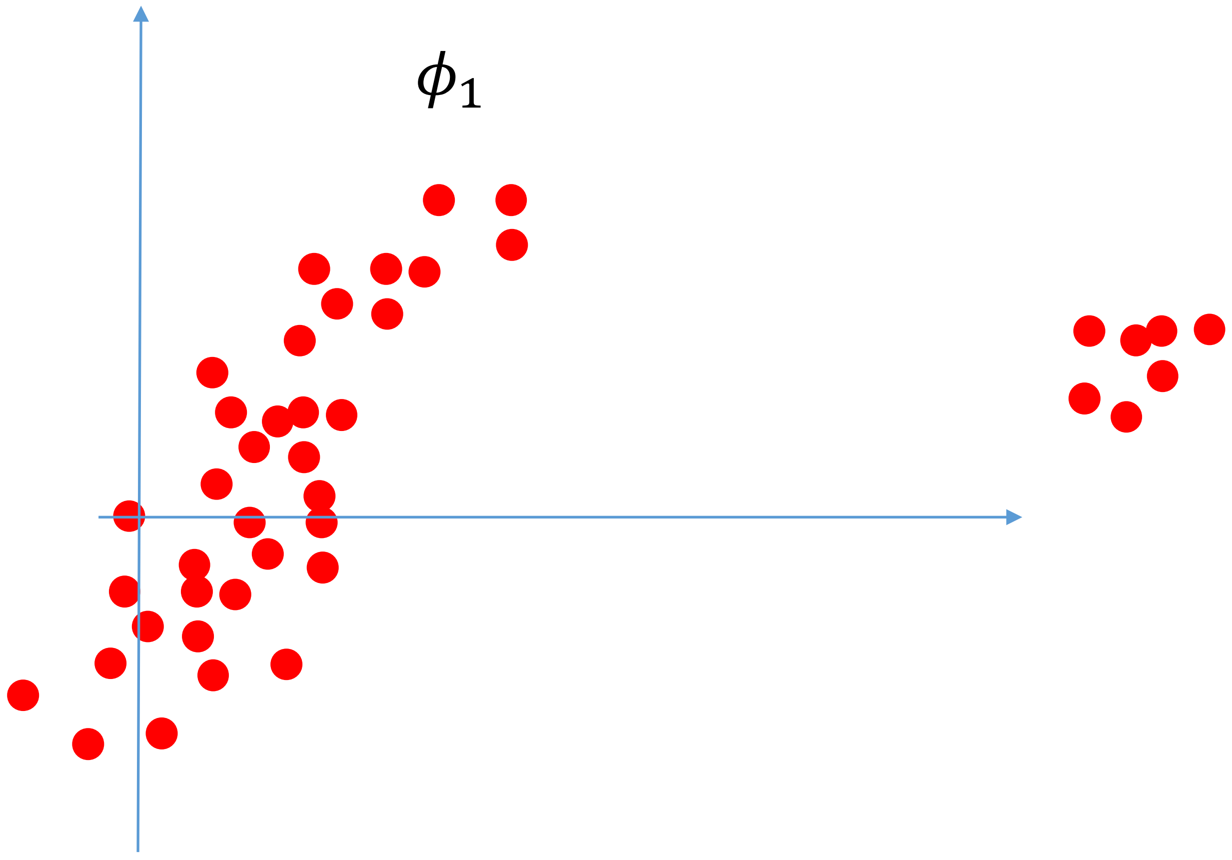
- Like any test based on histograms, QT does not assess distribution changes “within” bins. If you know “what type” of ϕ_0 you’ll have, then likelihood-based statistics are more powerful.



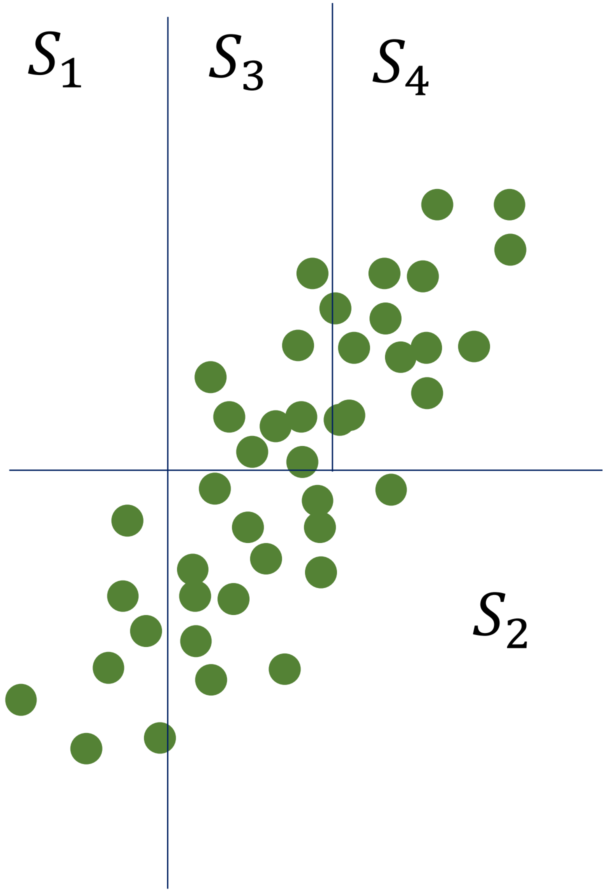
Limitations

- Like any test based on histograms, QT does not assess distribution changes "within" bins. If you know "what type" of ϕ_0 you'll have, then likelihood-based statistics are more powerful.

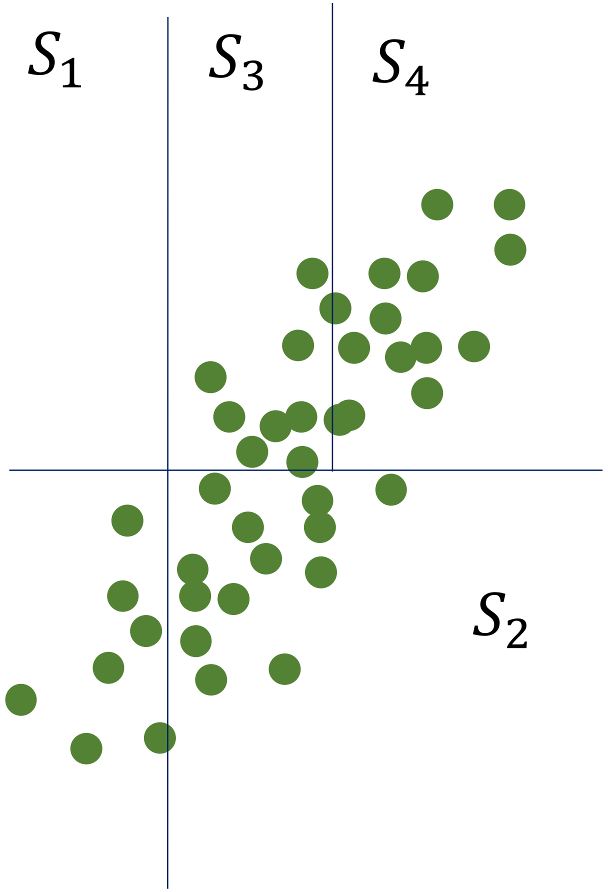


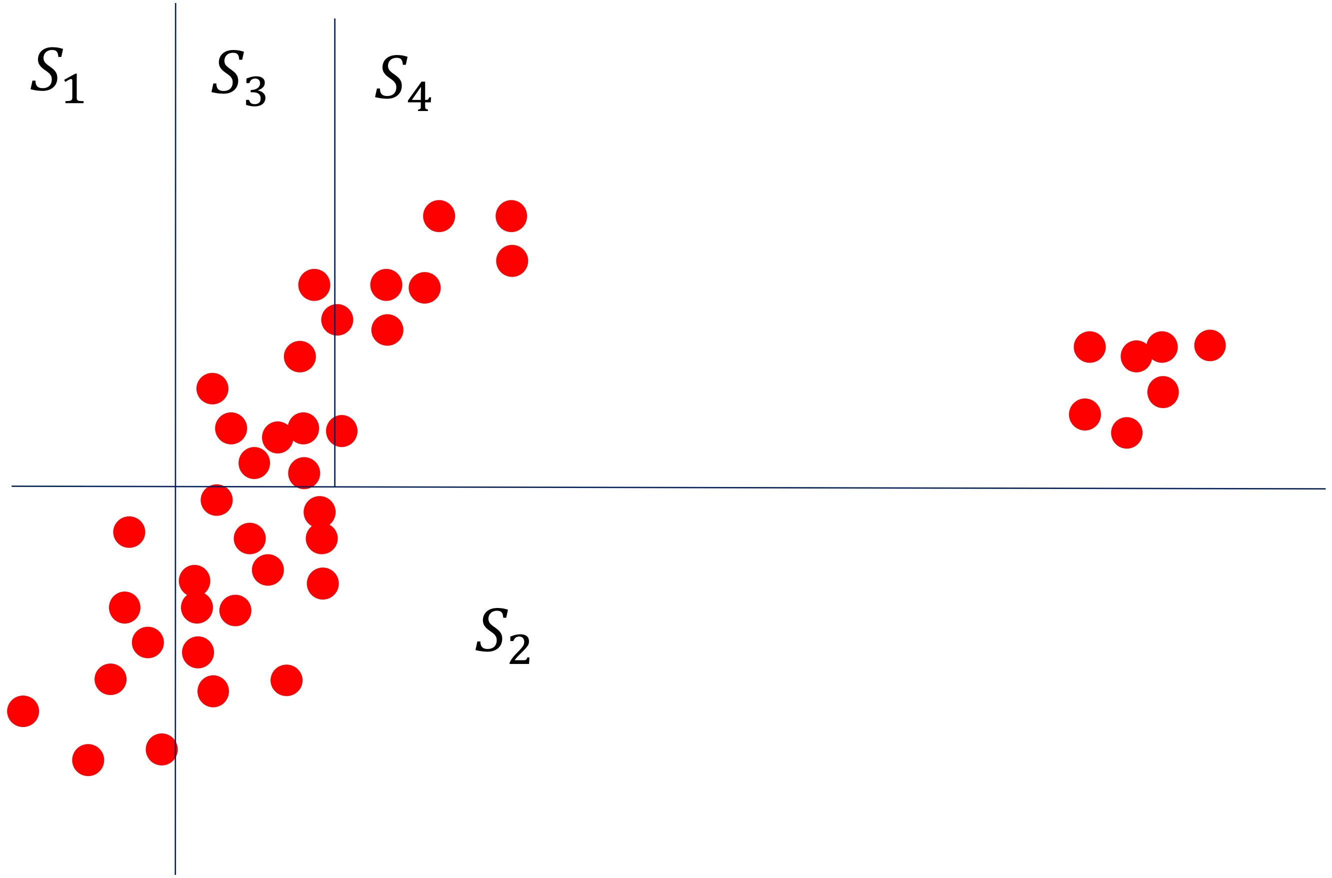


QuanTree Monitoring

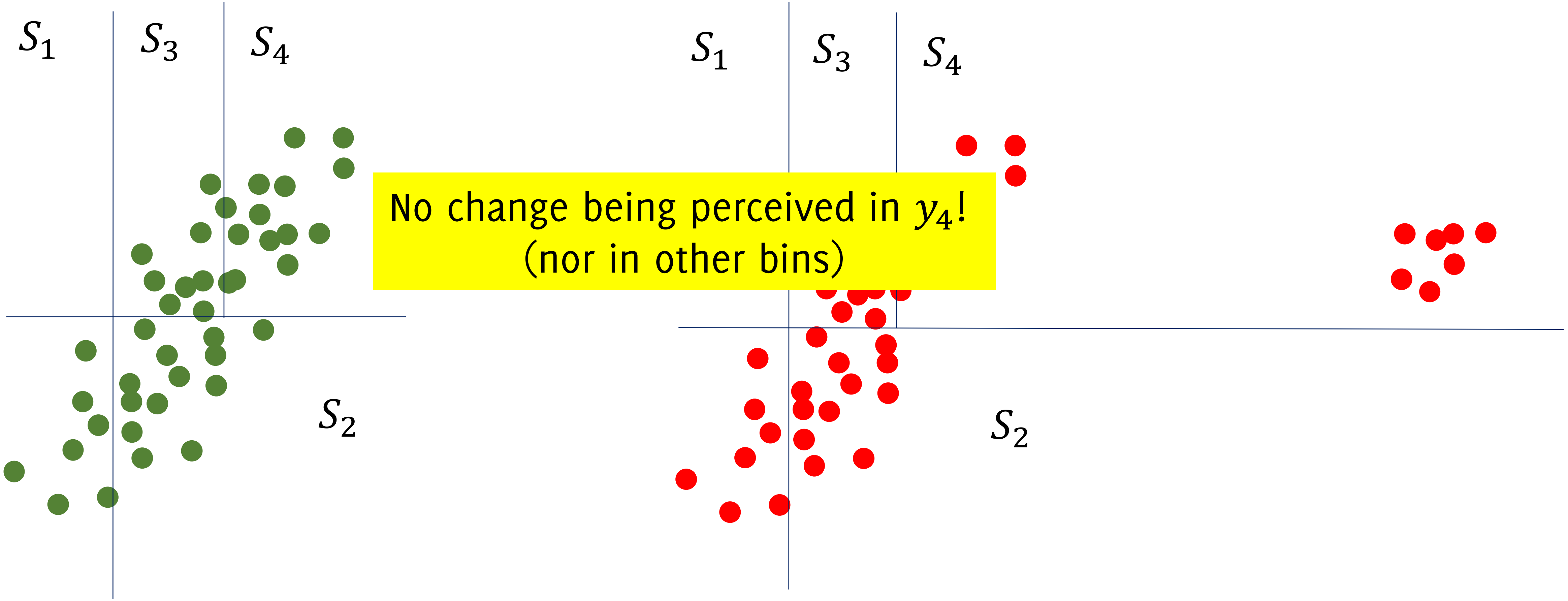


QuanTree Monitoring



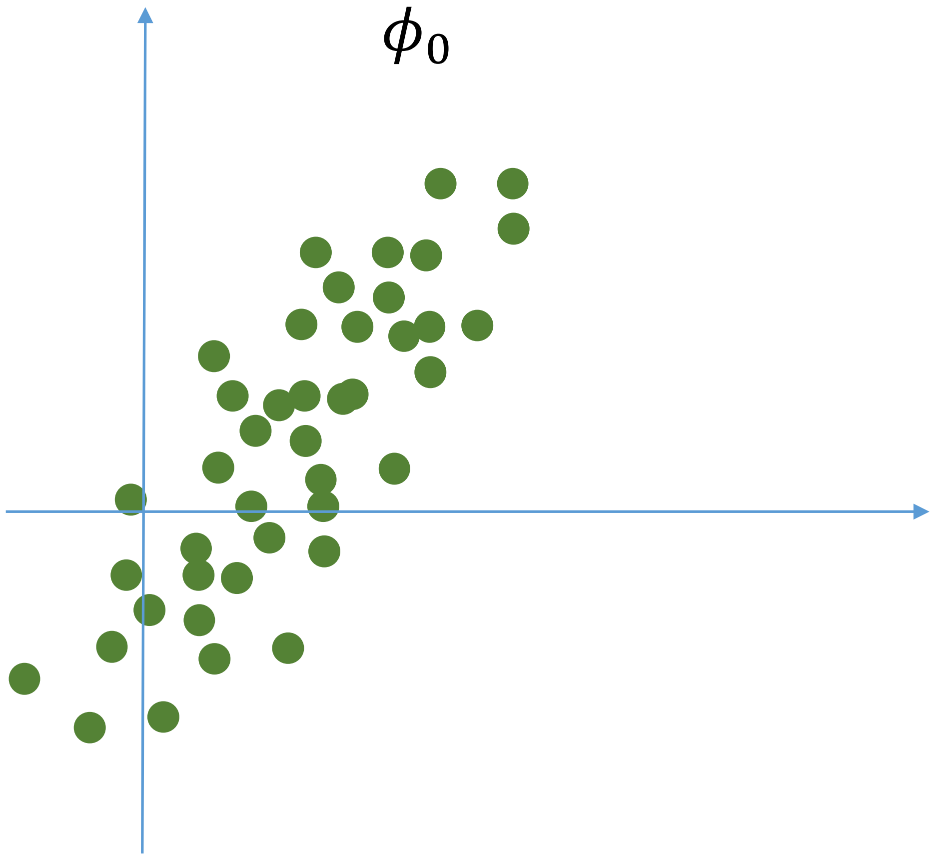


QuanTree Monitoring

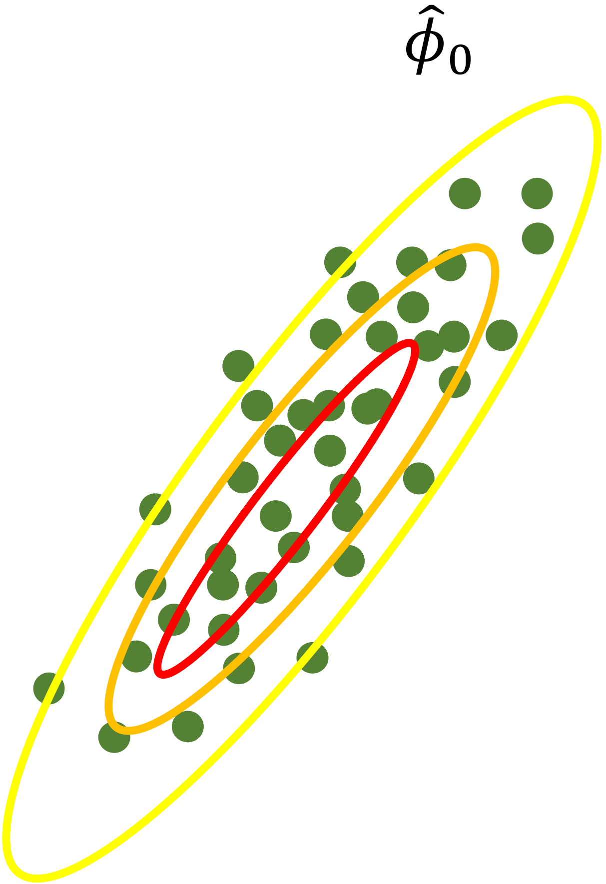


No change being perceived in y_4!
(nor in other bins)

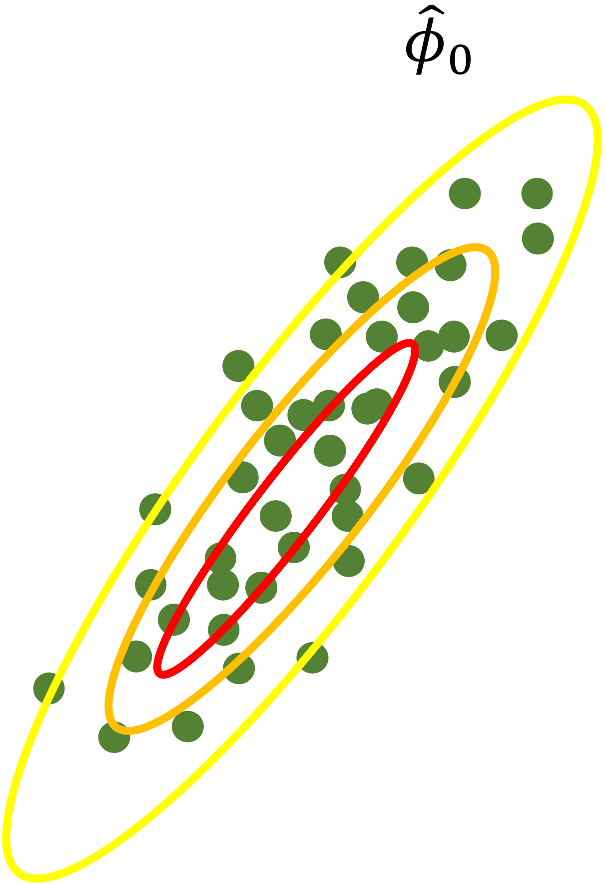
Likelihood- based monitoring

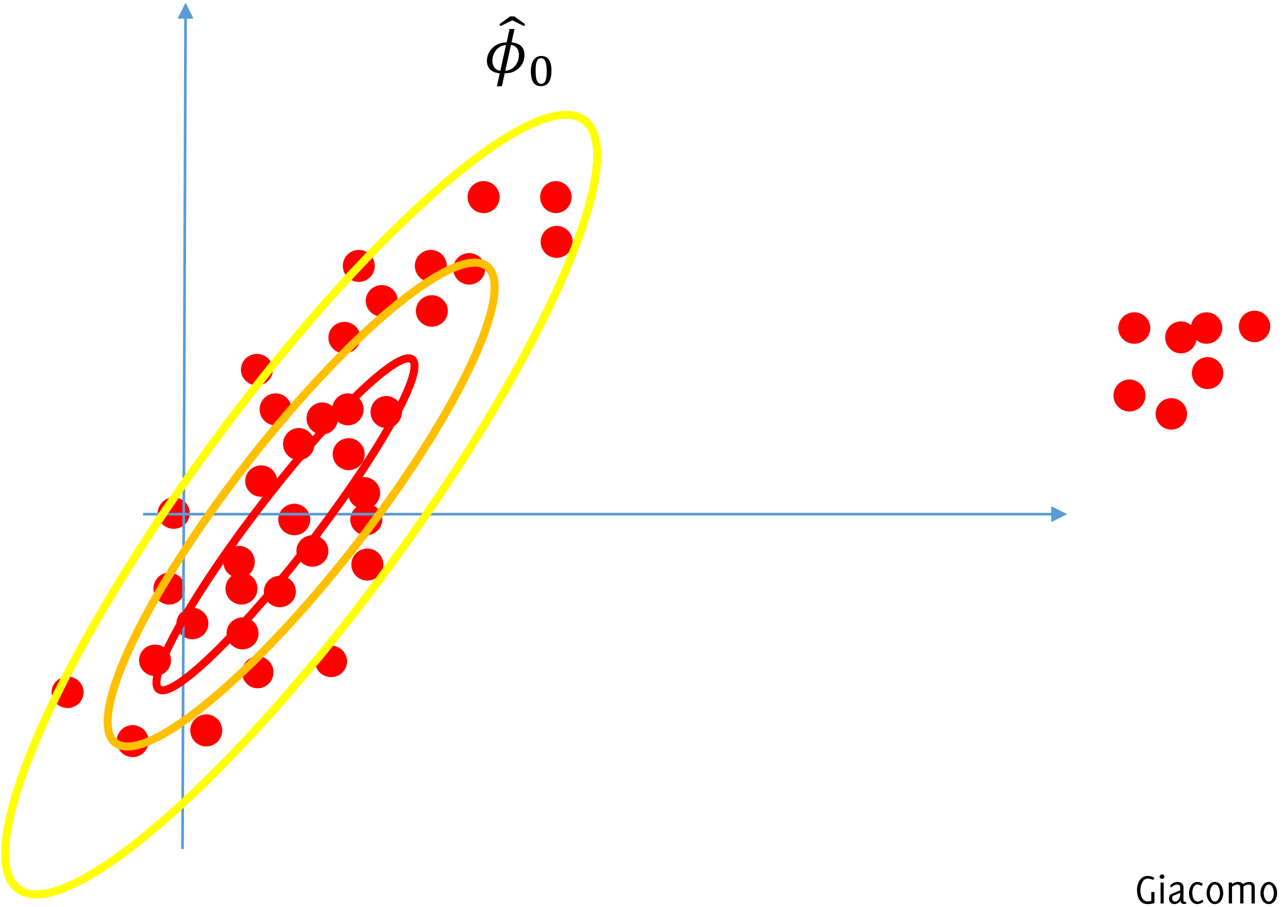


Likelihood- based monitoring



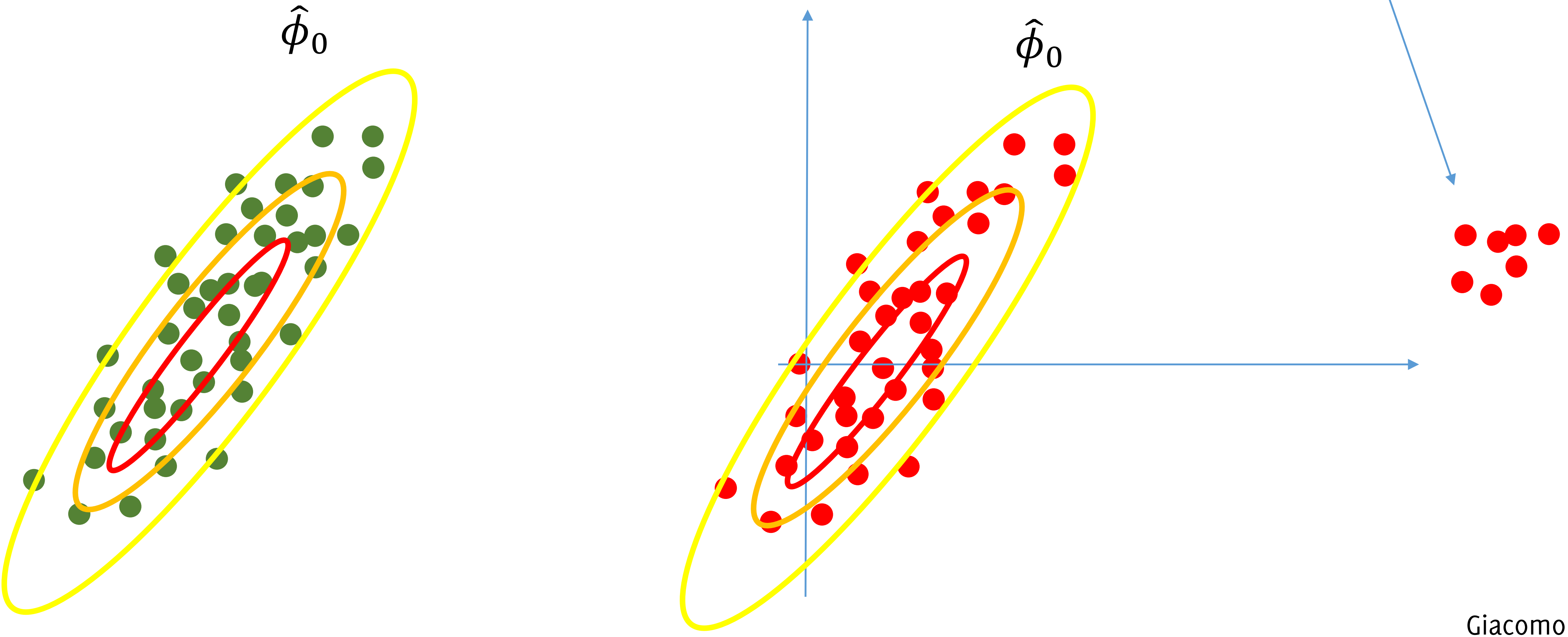
Likelihood- based monitoring





Likelihood- based monitoring

These samples are very unusual w.r.t. $\hat{\phi}_0$
$\hat{\phi}_0(\boldsymbol{x})$ would be very low!



Limitations

- Like any test based on histograms, QT does not assess distribution changes “within” bins. If you know “what type” of ϕ_0 you’ll have, then likelihood-based statistics are more powerful.
- Poor in efficiency compared to other tree structures (e.g., kdTrees that are balanced)
- Just an HT… it does not perform sequential monitoring

QT-Exponential Weighted Moving Average (QT-EWMA)

Sequential Monitoring by QuantTrees

Sequential Monitoring Settings

Online monitoring:

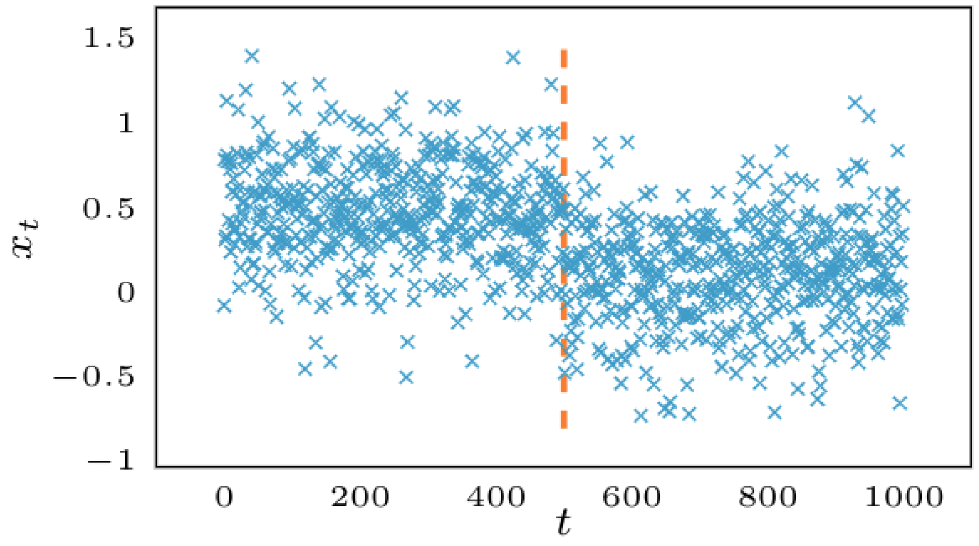
- At time t, a new sample $\boldsymbol{x}(t)$ arrive and a decision must be made

Sequential Monitoring Settings

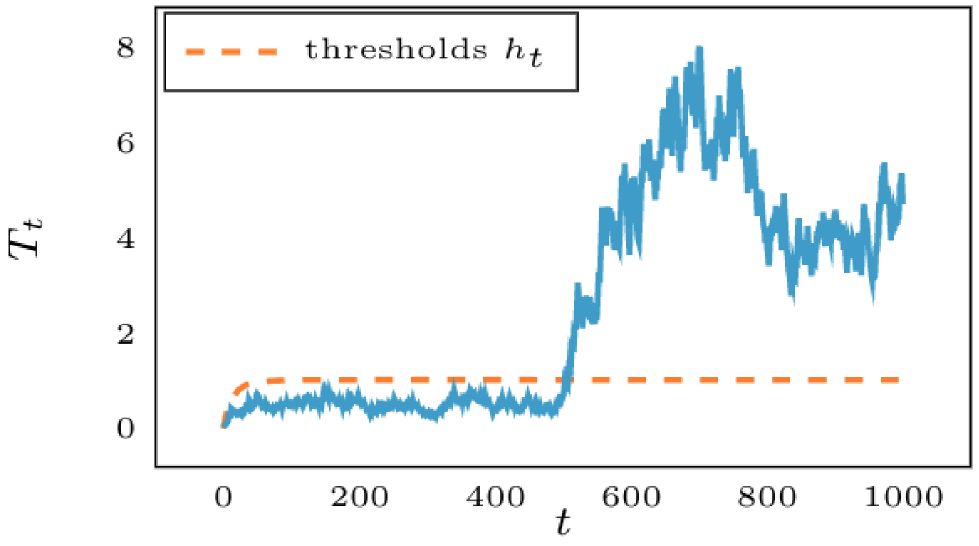
Online monitoring:

- At time t, a new sample $\boldsymbol{x}(t)$ arrive and a decision must be made
- After $\phi_0 \to \phi_1$, the evidence for a change increases and the test is expected to be more powerful

ϕ_0 ϕ_1



"nice" (sequential) test statistic



Sequential Monitoring Settings

Online monitoring:

- At time t, a new sample $\boldsymbol{x}(t)$ arrive and a decision must be made
- After $\phi_0 \to \phi_1$, the evidence for a change increases and the test is expected to be more powerful
- There is no clear notion of false alarm, rather measure **the expected time between false positive**, Average Run Length ARL_0

$$ARL_0 = \mathrm{E}_{\boldsymbol{x}}[\hat{\tau} | \boldsymbol{x} \sim \phi_0]$$

- Similarly, rather than the test power (TPR or AUC), rather measure **the expected detection delay**

$$ARL_1 = \mathrm{E}_{\boldsymbol{x}}[\hat{\tau} | \boldsymbol{x} \sim \phi_1]$$

Sequential Monitoring Challenges

Computational:

- Each decision should be made in constant time
- Impossible to store previously observed data as a reference

Theoretical:

- Difficult to define sequential statistics with multivariate data
- Difficult to define, for a target value of ARL_0, the corresponding threshold $\gamma = \gamma(ARL_0)$ which do not depend on ϕ_0
- Bootstrap is often not a viable alternative since we need to **consider temporal evolution** of the analysis

EWMA: Exponential Weighted Moving Average

EWMA is a standard sequential monitoring scheme for **1D** datastreams

We take inspiration from **ECDD for concept-drift** monitoring

$$Z_0 = 0, \qquad Z_t = (1-\lambda) Z_{t-1} + \lambda\, e_t$$

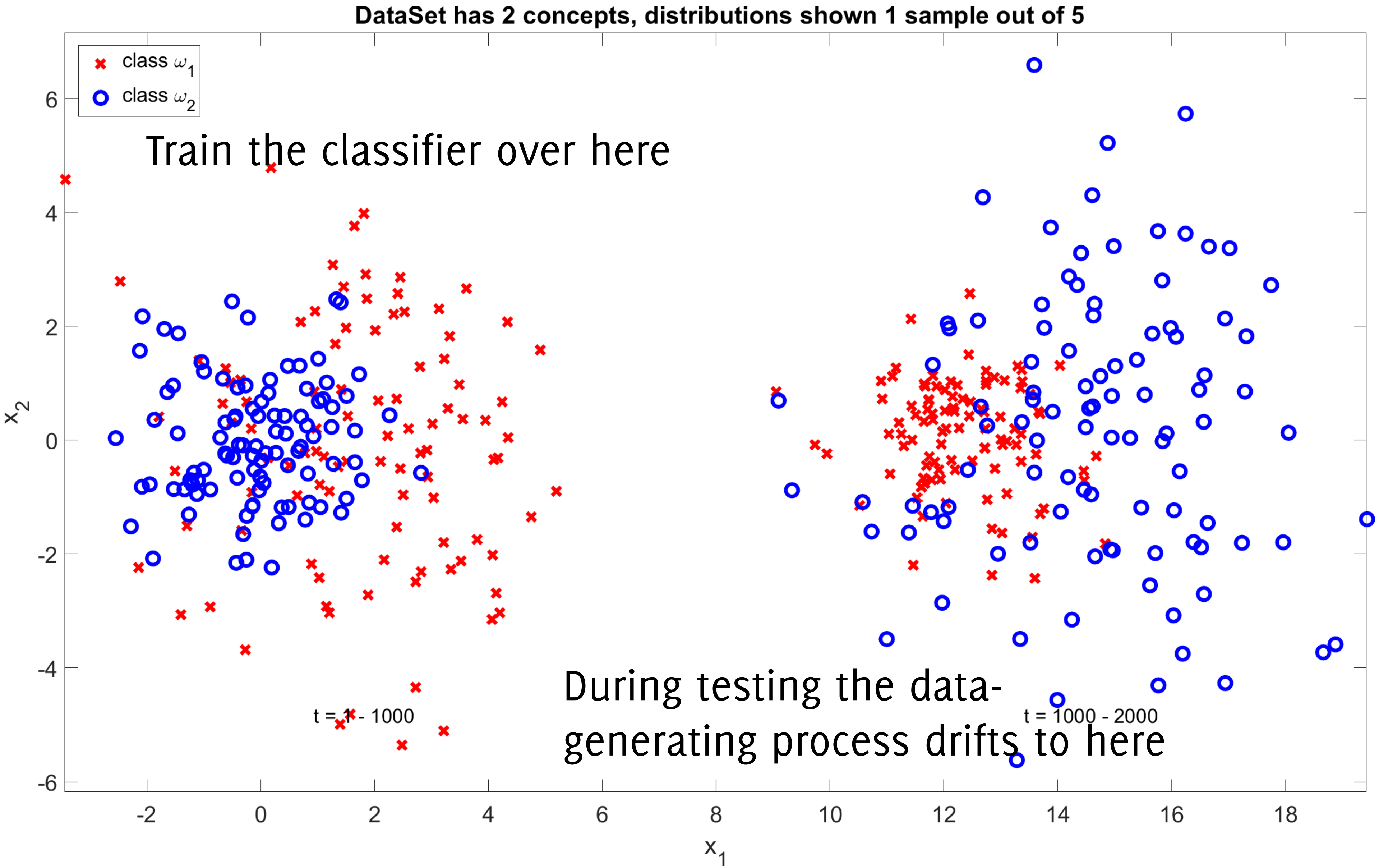
- $e_t \in \{0,1\}$ is the **classification error** of a classifier at time t
- $\lambda \in [0,1]$ is a parameter regulating test “reactiveness”

As a matter of fact

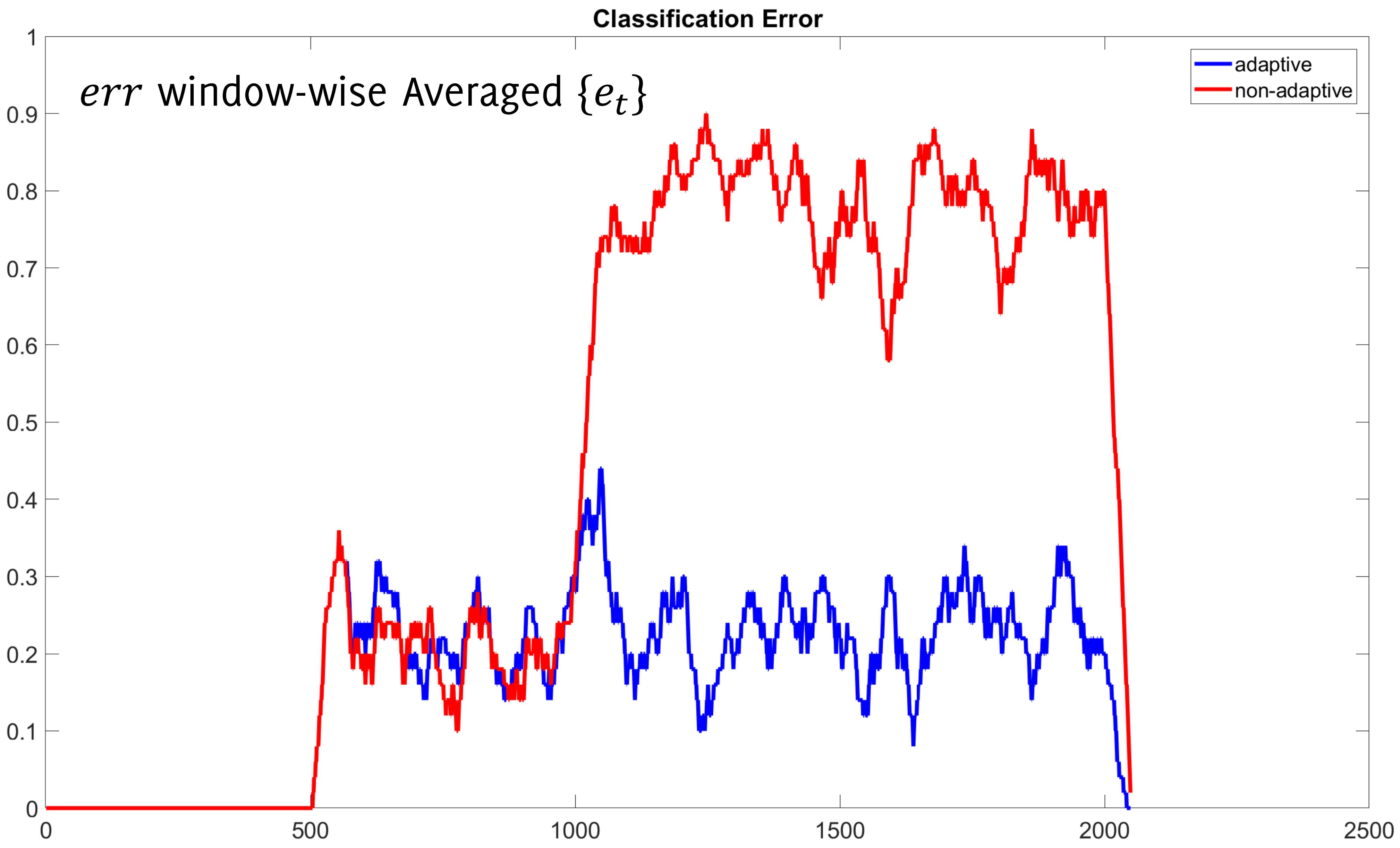
- Z_t is in stationary conditions tends to the average classification error
- After a change, Z_t moves towards the post-change classification error

G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand *"Exponentially Weighted Moving Average Charts for Detecting Concept Drift"* Pattern Recogn. Lett. 33, 2 (Jan. 2012), 191–198 2012

ECDD Example



ECDD Example (classes' swap)



ECDD Detection Scheme

In ECDD it is possible to set a detection rule controlling ARL_0

$$Z_t > p_0 + L_t \sigma_{Z_t}$$

Defining the sequence $\{L_t\}_t$ is very complicated as these depend on $\hat{p}_{0,t}$ (the estimated classification error)

A «simple» problem to address via MonteCarlo simulation is, given a value L and p_0, to estimate the corresponding ARL_0

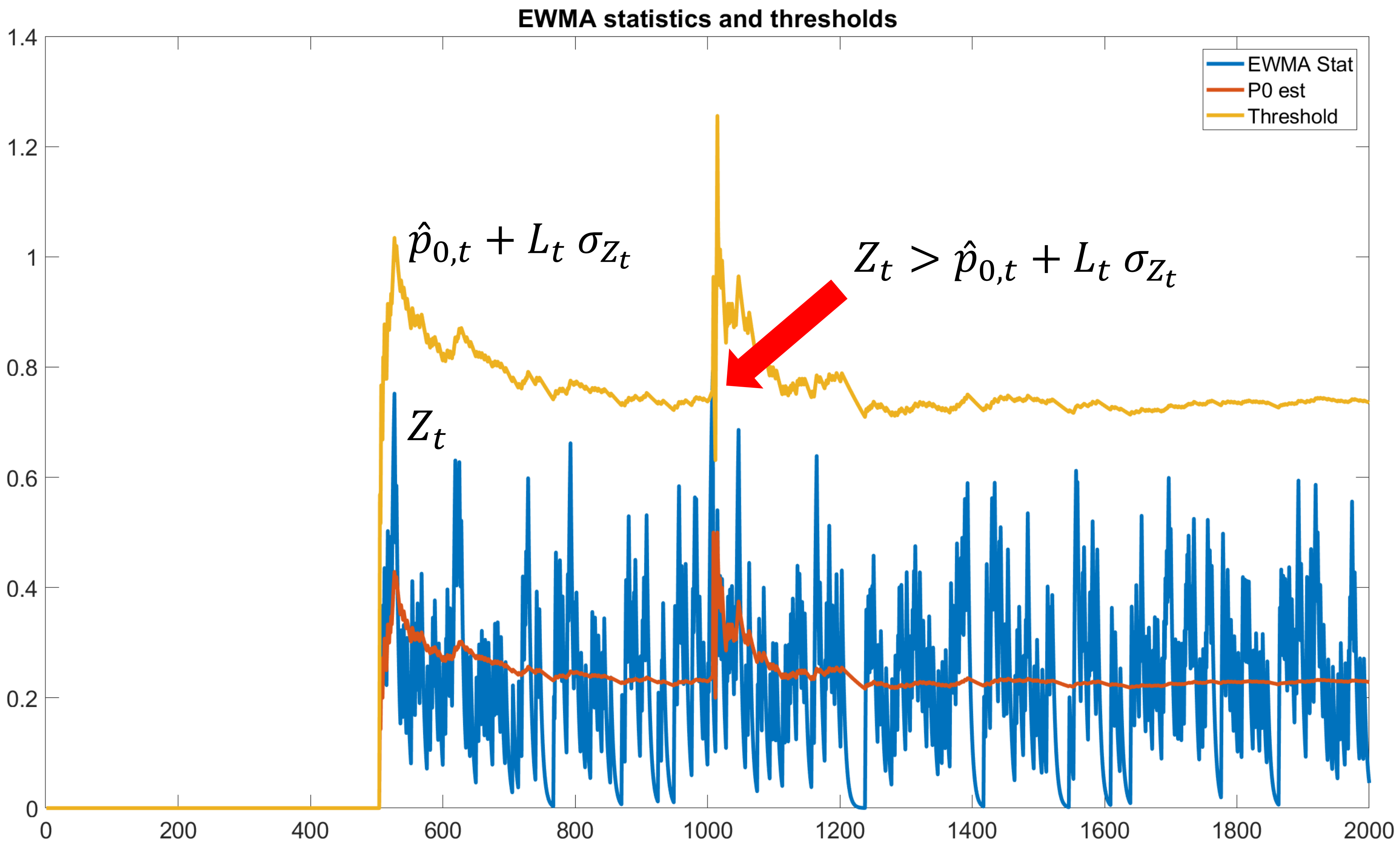
$$Montecarlo(L, p_0) \rightarrow ARL_0$$

It is also possible «to revert» this by setting up a suitable Montecarlo scheme such that, provided ARL_0 and p_0 one estimates L

This holds because e_t follows a Bernoulli distribution

G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand *"Exponentially Weighted Moving Average Charts for Detecting Concept Drift"* Pattern Recogn. Lett. 33, 2 (Jan. 2012), 191–198 2012

ECDD Statistics



Sequential Monitoring by QT: Idea

Idea:

- Compute a single ***«bin-wise»*** **EWMA statistic** on the proportion of samples falling in each bin. This is exactly the same of the classification error
- **Aggregate all the EWMA statistics** in a *Pearson-like* statistic
- Compute Detection thresholds **via the MonteCarlo process** in [Ross 2012], **but leveraging QT properties** to speed up simulations

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

QT-EWMA: QuantTree EWMA

Define an online statistic $\mathcal{T}_t$ to monitor the proportion of samples falling in each bin S_k of a QuantTree $h = \{S_k, \hat{\pi}_k\}$. For each new sample define

$$y_{k,t} = \mathbb{I}(x_t \in S_k) = \begin{cases} 0 & x_t \notin S_k \\ 1 & x_t \in S_k \end{cases}$$

where, $E_{\phi_0}\left[y_{k,t}\right] = p_k$ (the probability for a sample to fall in S_k)

We define K *"bin-wise"* EWMA

$$Z_{k,0} = 0, \qquad Z_{k,t} = (1-\lambda) Z_{k,t-1} + \lambda\, y_{k,t} \quad \forall k = 1, \dots, K$$

and a Change Detection Statistic

$$\mathcal{T}_t = \sum_{k=1}^{K} \frac{\left(Z_{k,t} - \hat{\pi}_k\right)^2}{\hat{\pi}_k}$$

Which is the Pearson Statistics monitoring how much the bin-wise EWMA departs from $\hat{\pi}_k$

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

The QT-EWMA Algorithm

find in which bin each sample falls

Algorithm 1: QT-EWMA

input : datastream $x_1, x_2, \ldots$, target $\{\pi_j\}_{j=1}^K$, thresholds $\{h_t\}_t$, TR

output : detection flag `ChangeDetected`, detection time t^*

`ChangeDetected` $\leftarrow$ False, $\quad t^* \leftarrow \infty$;

estimate QT histogram $\{(S_j, \pi_j)\}_{j=1}^K$ from TR and define $\{\hat{\pi}_j\}_{j=1}^K$ as in (4);

$Z_{j,0} \leftarrow \hat{\pi}_j \; \forall j = 1, \ldots, K$;

for $t = 1, \ldots$ **do**

    $y_{j,t} \leftarrow \mathbb{1}(x_t \in S_j)$;

    $Z_{j,t} \leftarrow (1-\lambda) Z_{j,t-1} + \lambda y_{j,t}, \quad j = 1 \ldots, K$;

    $T_t \leftarrow \sum_{j=1}^K (Z_{j,t} - \hat{\pi}_j)^2 / \hat{\pi}_j$;

    if $T_t > h_t$ **then**

        `ChangeDetected` $\leftarrow$ True, $\quad t^* \leftarrow t$;

        break;

    end

end

return `ChangeDetected`, t^*

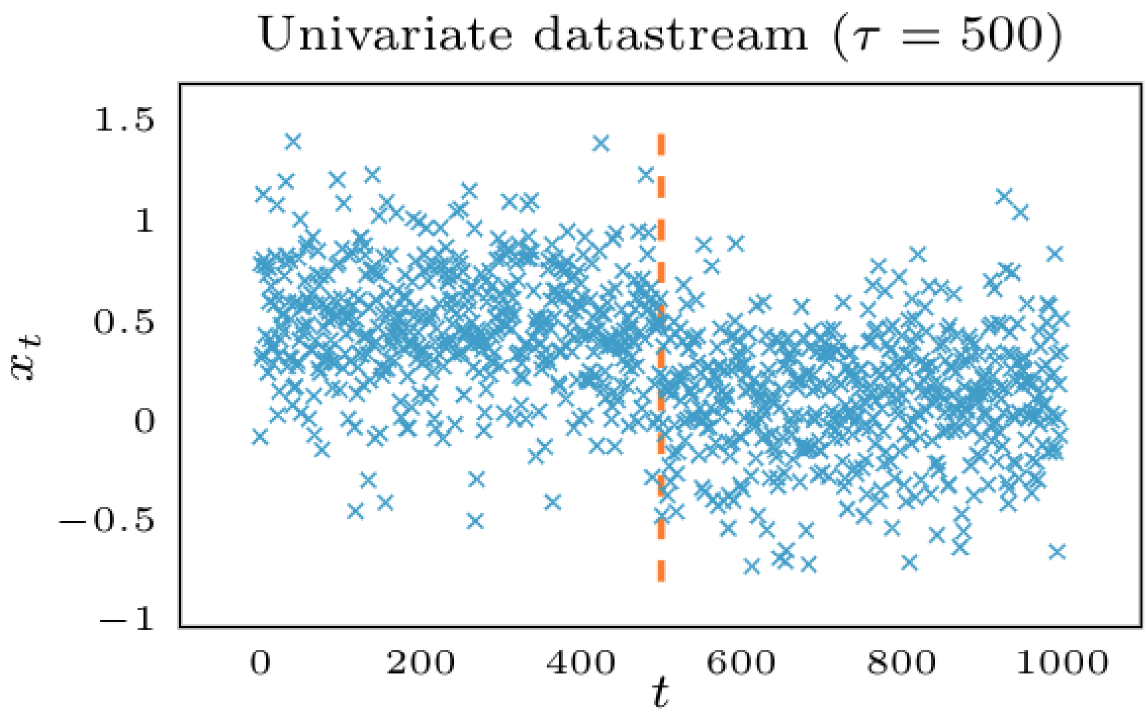
monitor the empirical bin probabilities by EWMA statistics $Z_{j,t}$

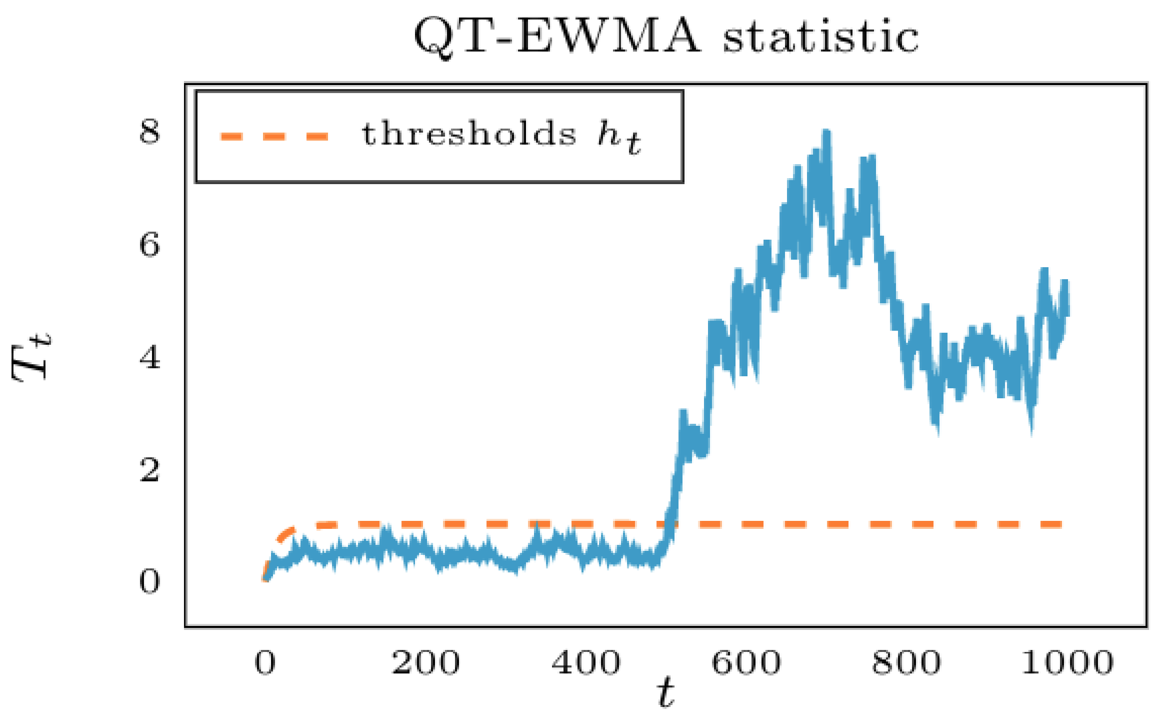
measure the deviation from the expected probabilities by $\mathcal{T}_t$

detect a change when $\mathcal{T}_t$ exceeds a threshold h_t

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

Example





The **deviation** of the **bin probabilities** from their expected values measured by T_t **increases** after a **distribution change**

QT-EWMA: Thresholds $\{h_t\}$ computation

The theoretical properties of QuantTree **guarantee** that our statistics are **independent** from ϕ_0, and d.

Test statistics depends on N, the target ARL_0, the parameter λ and $\{\pi_k\}$

We set h_t to keep a constant probability of a false alarm at each time t

$$P(\mathcal{T}_t > h_t | \mathcal{T}_\tau < h_\tau, \forall \tau < t) = \alpha = \frac{1}{ARL_0}$$

We design an efficient **Monte Carlo scheme** to compute these thresholds using theoretical results from QT.

We regularize $\{\boldsymbol{h_t}\}$ by fitting a polynomial in t^{-1} to the empirical estimates

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

QT-EWMA-update

When TR is very small, $\hat{\pi}_k$ are very far from the true probabilities, and

$$\mathcal{T}_t = \sum_{k=1}^{K} \frac{\left(Z_{k,t} - \hat{\pi}_k\right)^2}{\hat{\pi}_k}$$

Is not very powerful as a test statistic

Idea: update bin probabilities $\hat{\pi}_k$ as long as no change is detected

$$\hat{p}_{k,0} = \hat{\pi}_k, \qquad \text{and } \hat{p}_{k,t} = (1 - \omega_t)\hat{p}_{k,t-1} + \omega y_{k,t}$$

Where

$$\omega_t = \frac{1}{\beta(N + t)}$$

regulates the updating speed, and tends to 0 as t increases.

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

QT-EWMA-update Updating Speed

The updating speed is regulated by β

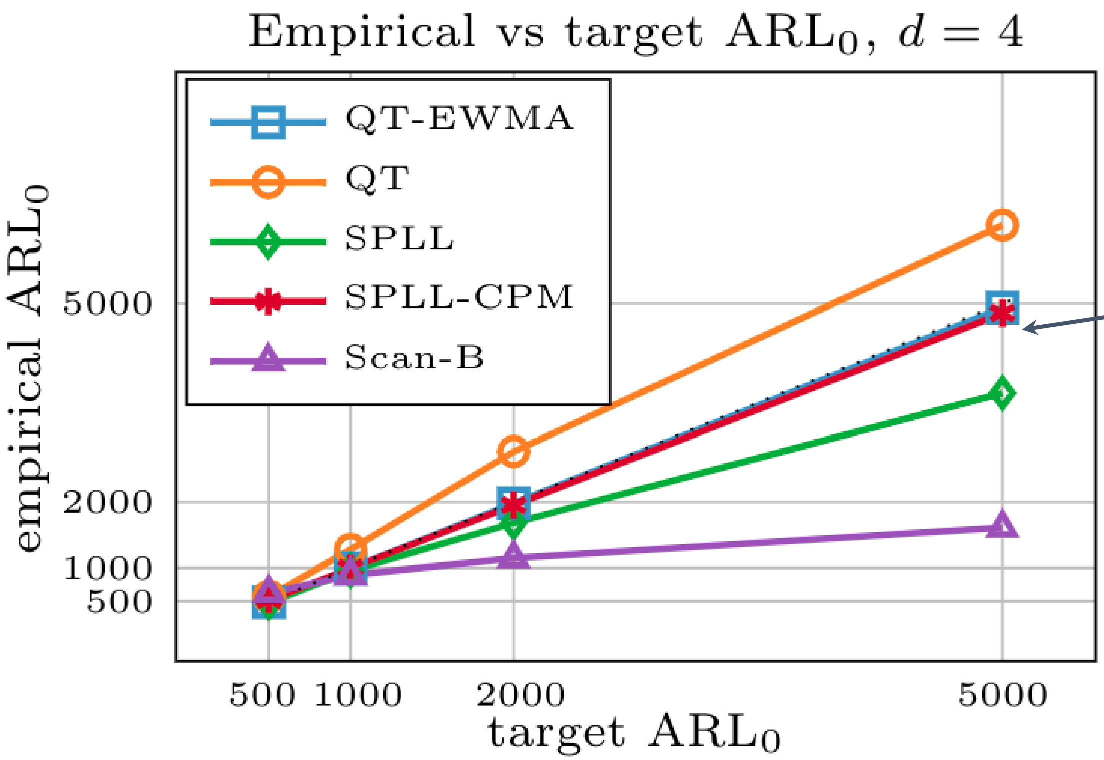
$$\omega_t = \frac{1}{\beta(N+t)}$$

- High values or β are meant to prevent updating the bin probabilities after the change
- The updating speed β is a parameter of QT-EWMA-update. Detection thredsholds depend on β as well

L. Frittoli, D. Carrera, G. Boracchi *"Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree"* IEEE TKDE 2022

Experiments: synthetic Gaussian data

We set different ARL_0 values and measure the **empirical** ARL_0 of QT-EWMA and the other considered methods

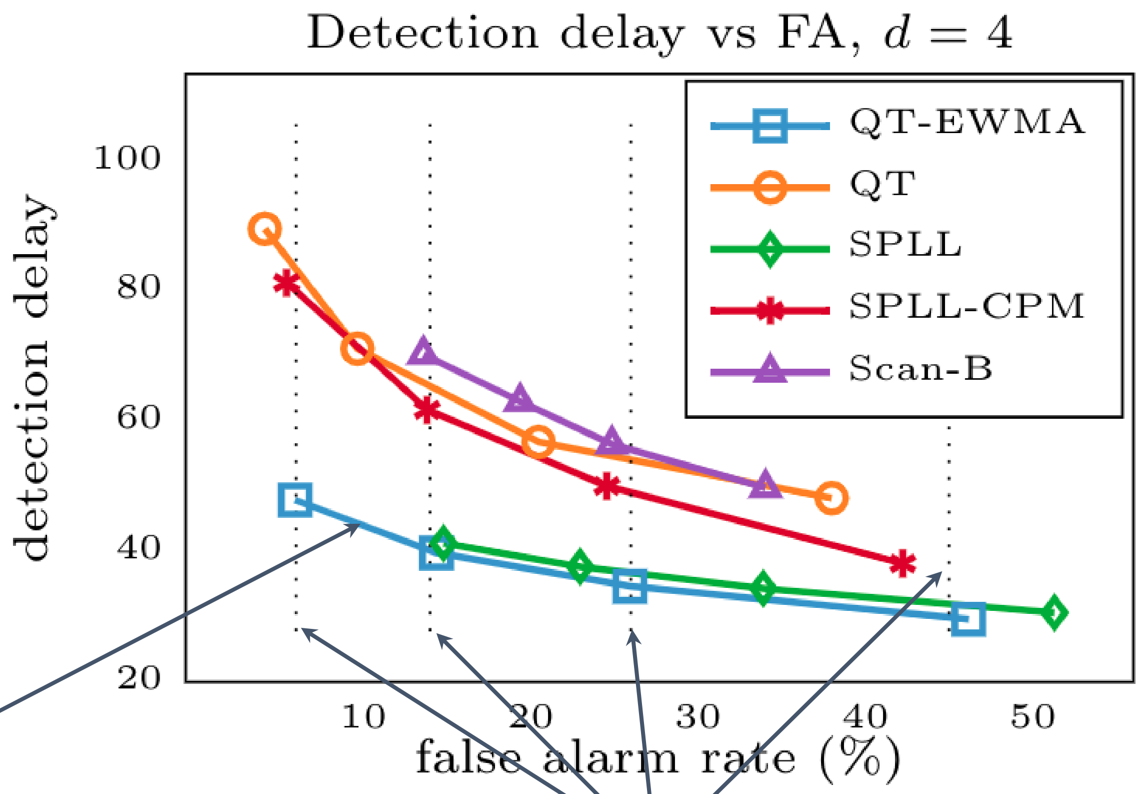


Goal: preserve the target ARL_0

[SPLL] L. Kuncheva "Change Detection in Streaming Multivariate Data Using Likelihood Detectors", IEEE TKDE, 2011
[Scan-B] S. Li et al. "M-Statistic for Kernel Change-Point Detection", Advances in Neural Information Processing Systems, 2015

Experiments: synthetic Gaussian data

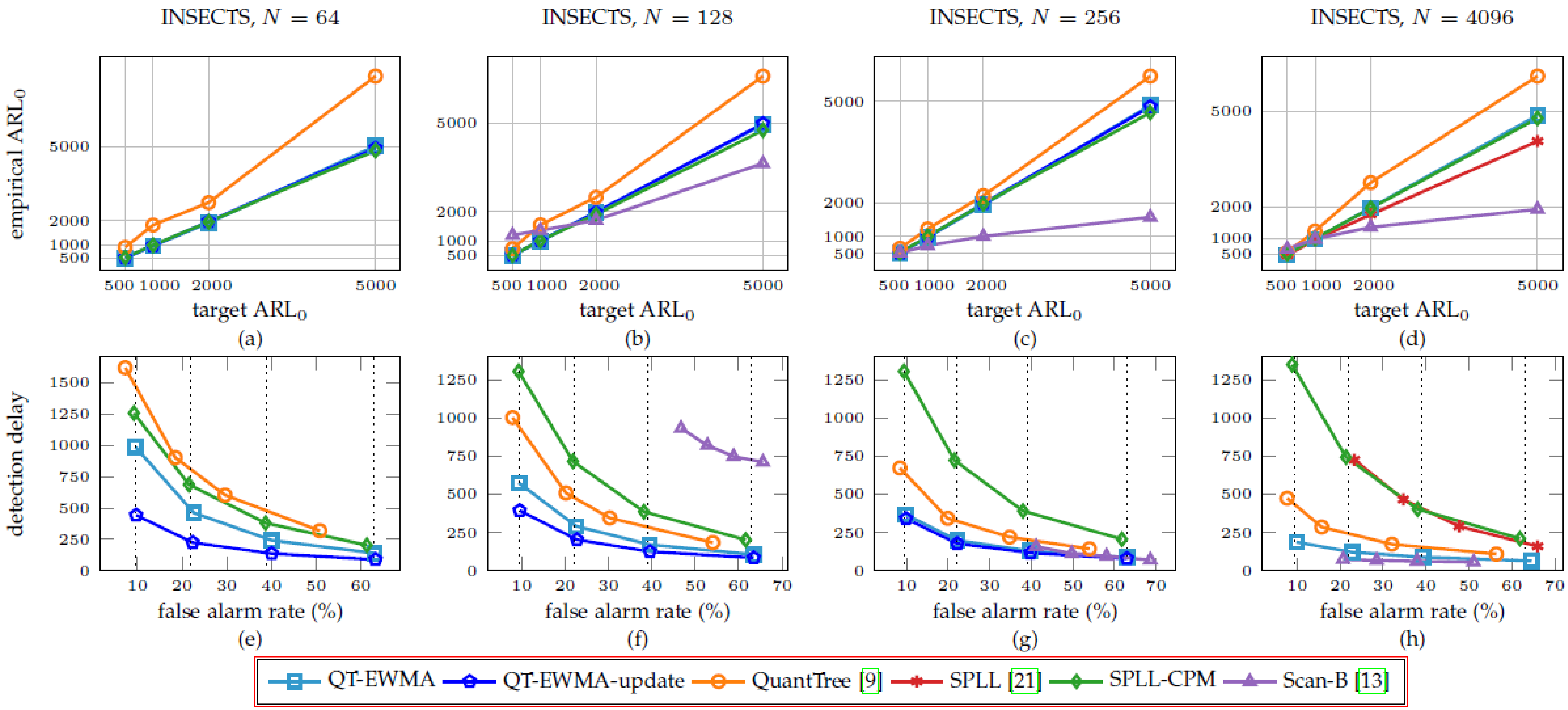
We set different ARL_0 values and observe the **trade-off** between **detection delay** and **false alarm rate**



Goal 1: minimize the detection delay

Goal 2: maintain the target false alarm rates depending on the target ARL_0

Experiments: Real data



Giacomo Boracchi

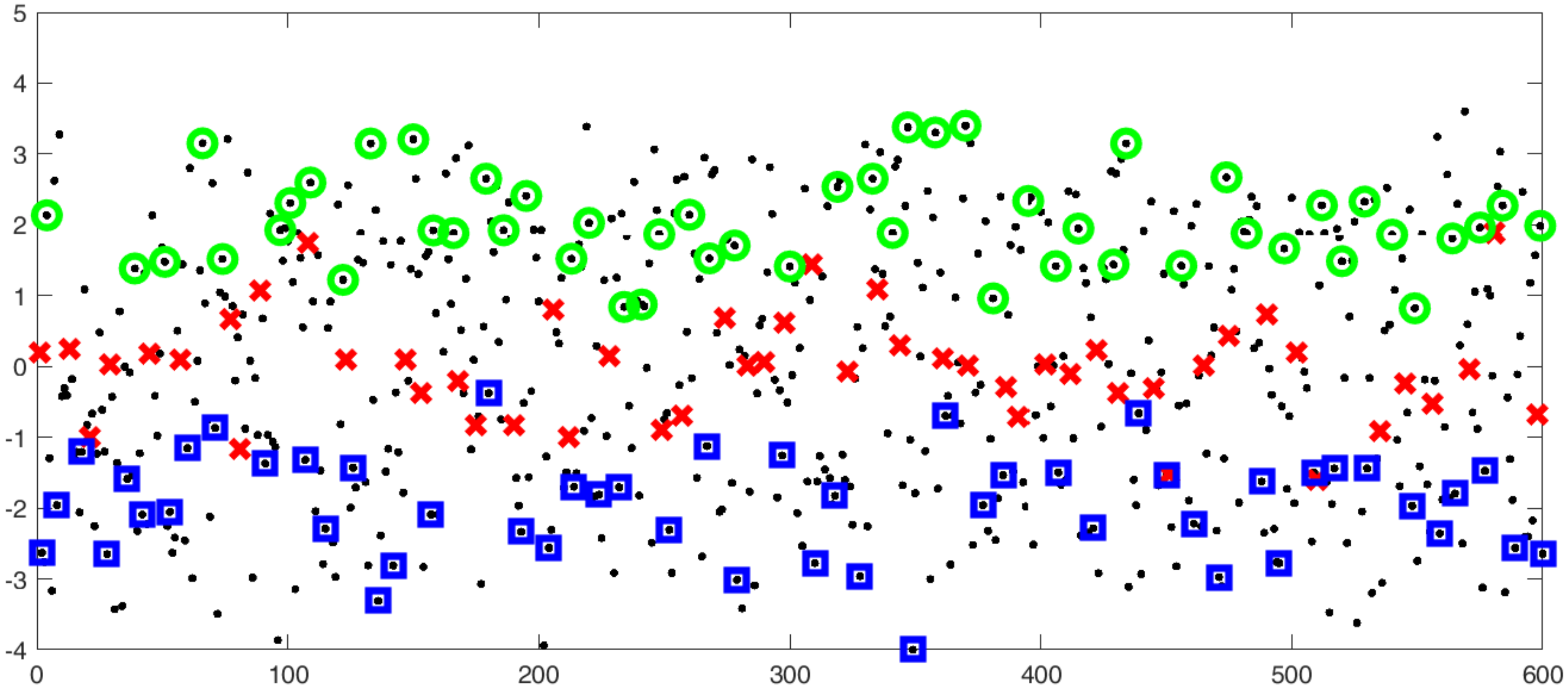
Class Distribution Monitoring for Concept Drift Detection

A partitioning scheme specifically
designed for change detection

Concept Drift Detection Approaches

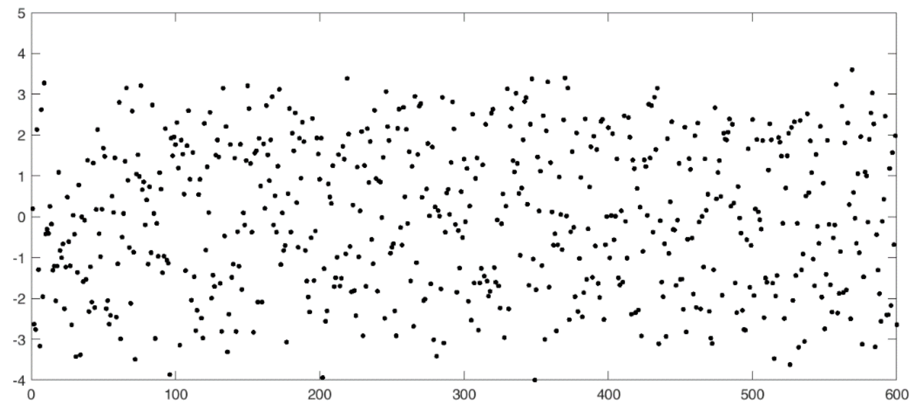
There are two mainstream approaches in concept drift detection:

- **Monitoring** $\{\boldsymbol{x}_t\}$, the stream of raw inputs
- **Monitoring** $\{e_t\}$, the stream of classification error



Monitoring the Raw Inputs

Raw inputs



Monitor by
QT-EWMA

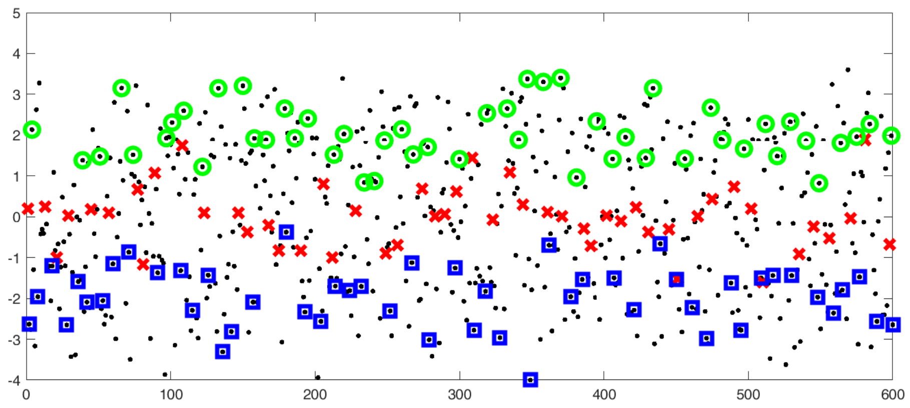
Concept Drift Detection

Monitoring $\{\boldsymbol{x_t}\}$, the stream of raw inputs

- PRO: does not require supervision
- PRO: informative in terms of data-generating process
- PRO: can detect new classes when these changes the distribution
- CONS: multivariate, difficult to monitor
- CONS: ignore labels when provided

Monitoring Classification Error

Annotated Labels



Compute Classification Error

Monitor by ECDD

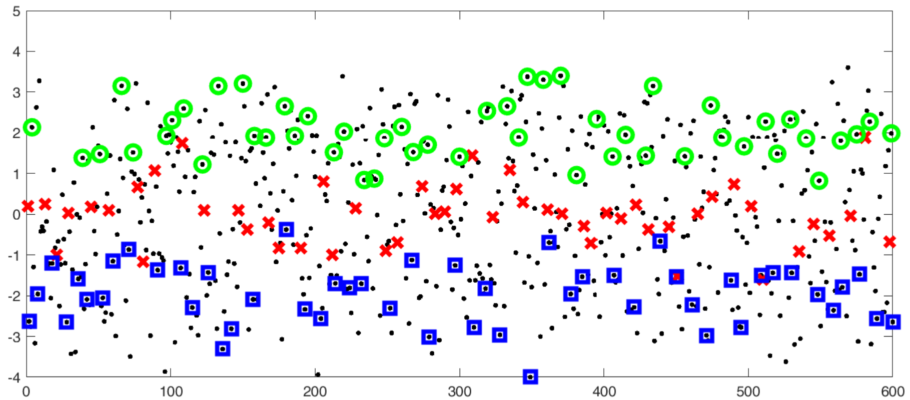
Concept Drift Detection

Monitoring $\{e_t\}$, the stream of classification error

- PRO: detect changes affecting the classification performance
- PRO: simple univariate monitoring schemes are required
- CONS: requires supervision
- CONS: new classes cannot be detected
- CONS: ignores *virtual drifts,* that do not affect the classification error
- CONS: little informative to monitor the data-generating process

Class-Distribution Monitoring

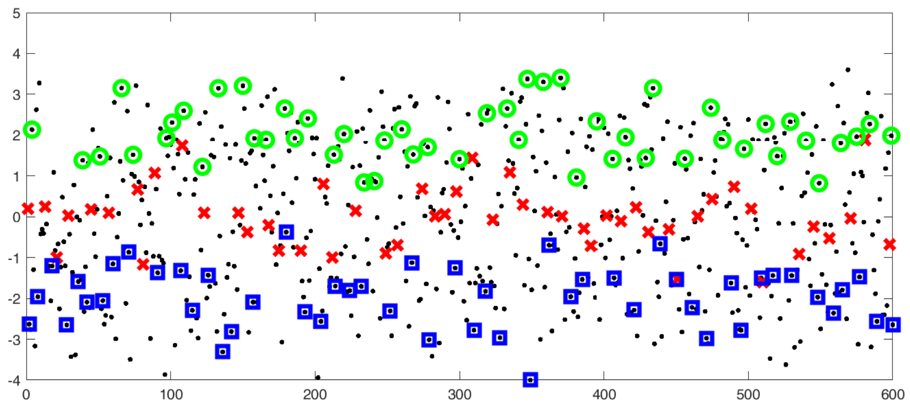
Annotated Labels

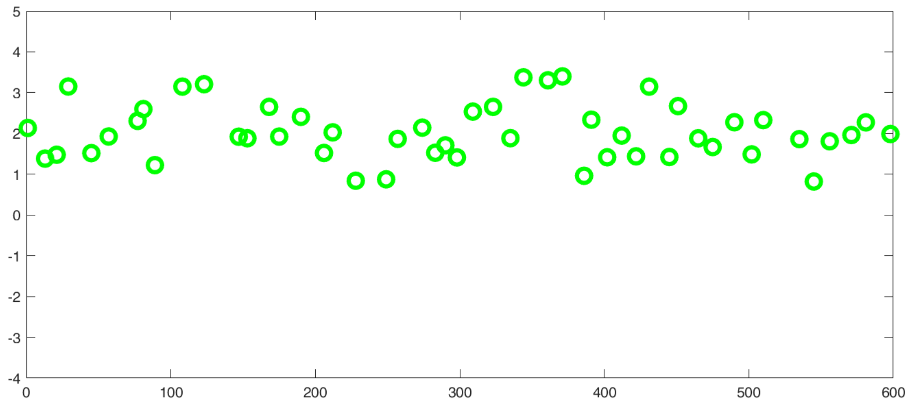


Divide the stream in class-wise labels

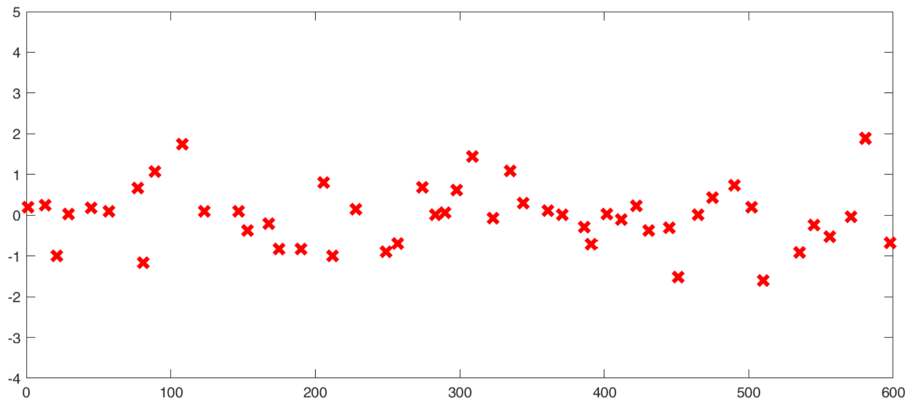
Class-Distribution Mon

Annotated Labels

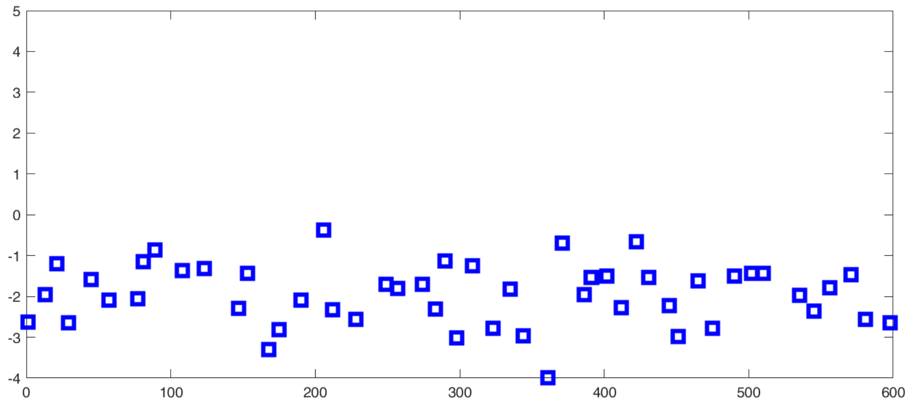




Monitor by QT-EWMA



Monitor by QT-EWMA



Monitor by QT-EWMA

Class-Distribution Monitoring

Idea: monitor the class-conditional distributions ϕ_0^m defined by

$$\mathbb{P}_{\phi_0^m}(x_t) = \mathbb{P}_{\phi_0}(x_t \mid y_t = m)$$

- **model** each class-conditional distribution by a **QuantTree histogram** [1]
- **monitor** samples from each class by **QT-EWMA** [2], an **online** and **nonparametric** change-detection algorithm

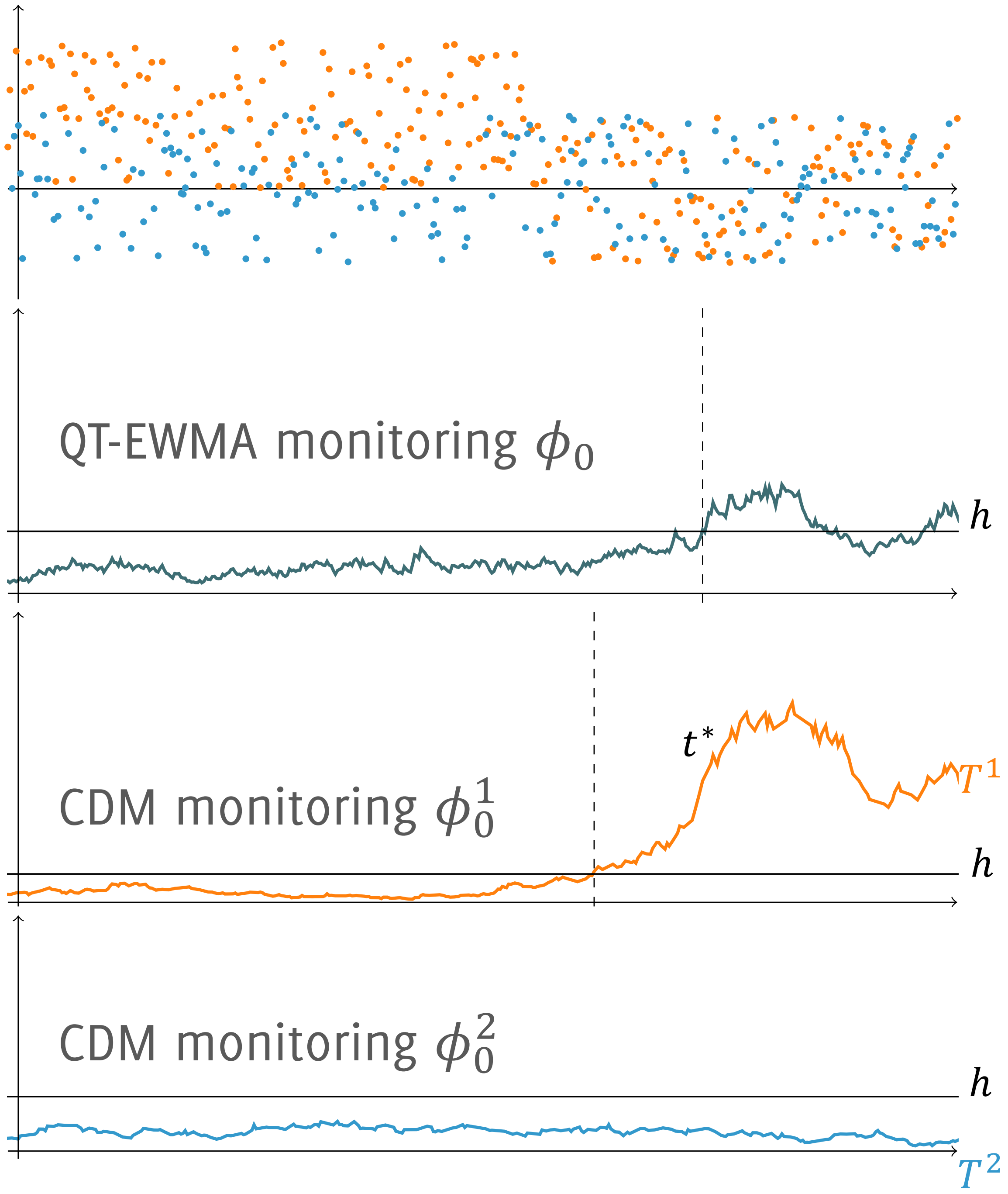
It is now possible to fairly compare:

- **ECDD** monitoring the classification error
- **QT-EWMA** monitoring each class

Since we can operate at **the same** ARL_0 **and the same amount of labels**

D. Stucchi, L. Frittoli, G. Boracchi *"Class Distribution Monitoring for Concept Drift Detection"* IEEE IJCNN 2022

Class-wise distribution monitoring!



Monitoring Raw-data distribution

Monitoring distribution class 1

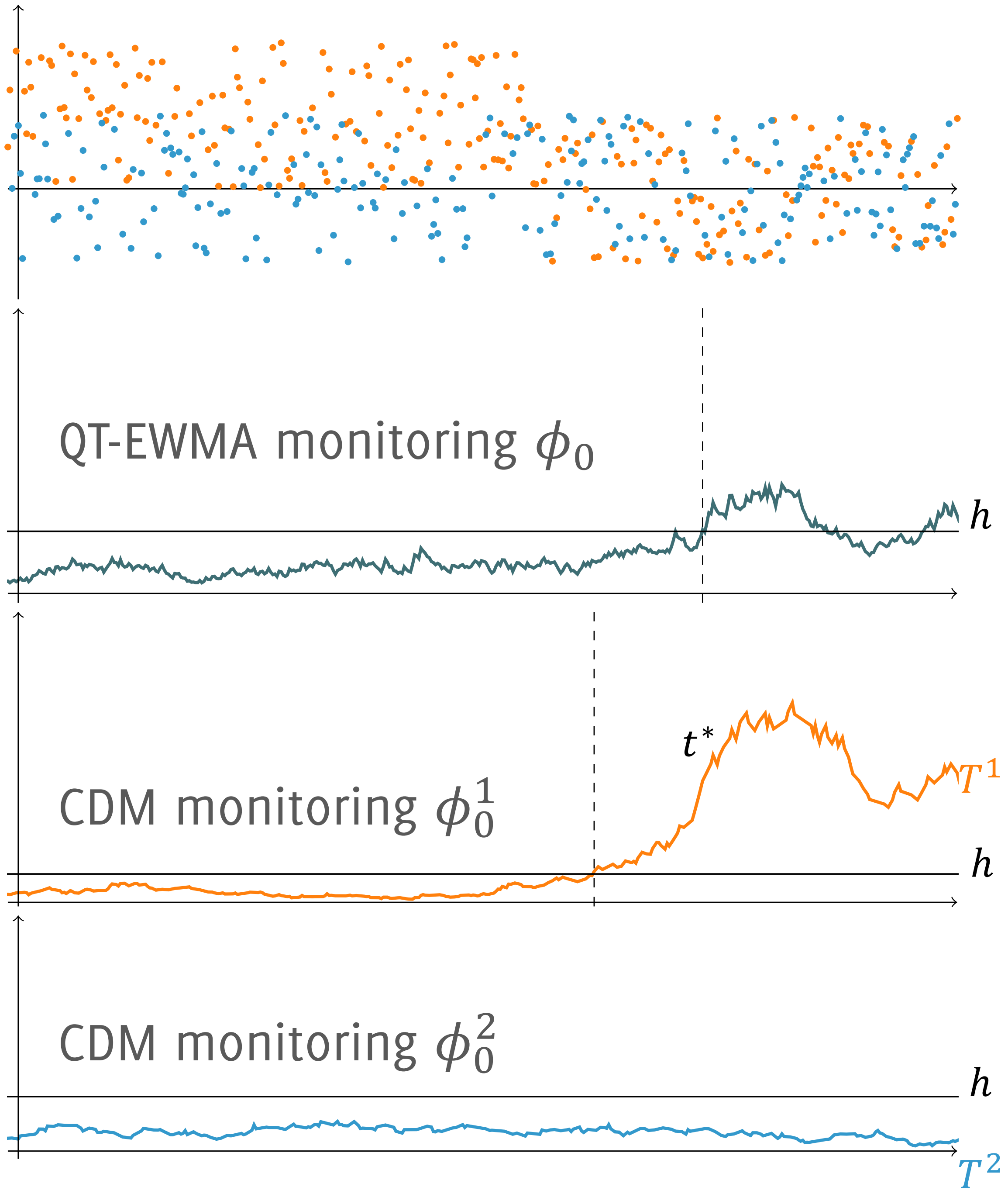
Monitoring distribution class 2

Giacomo Boracchi

Class-wise distribution monitoring!

Advantages:

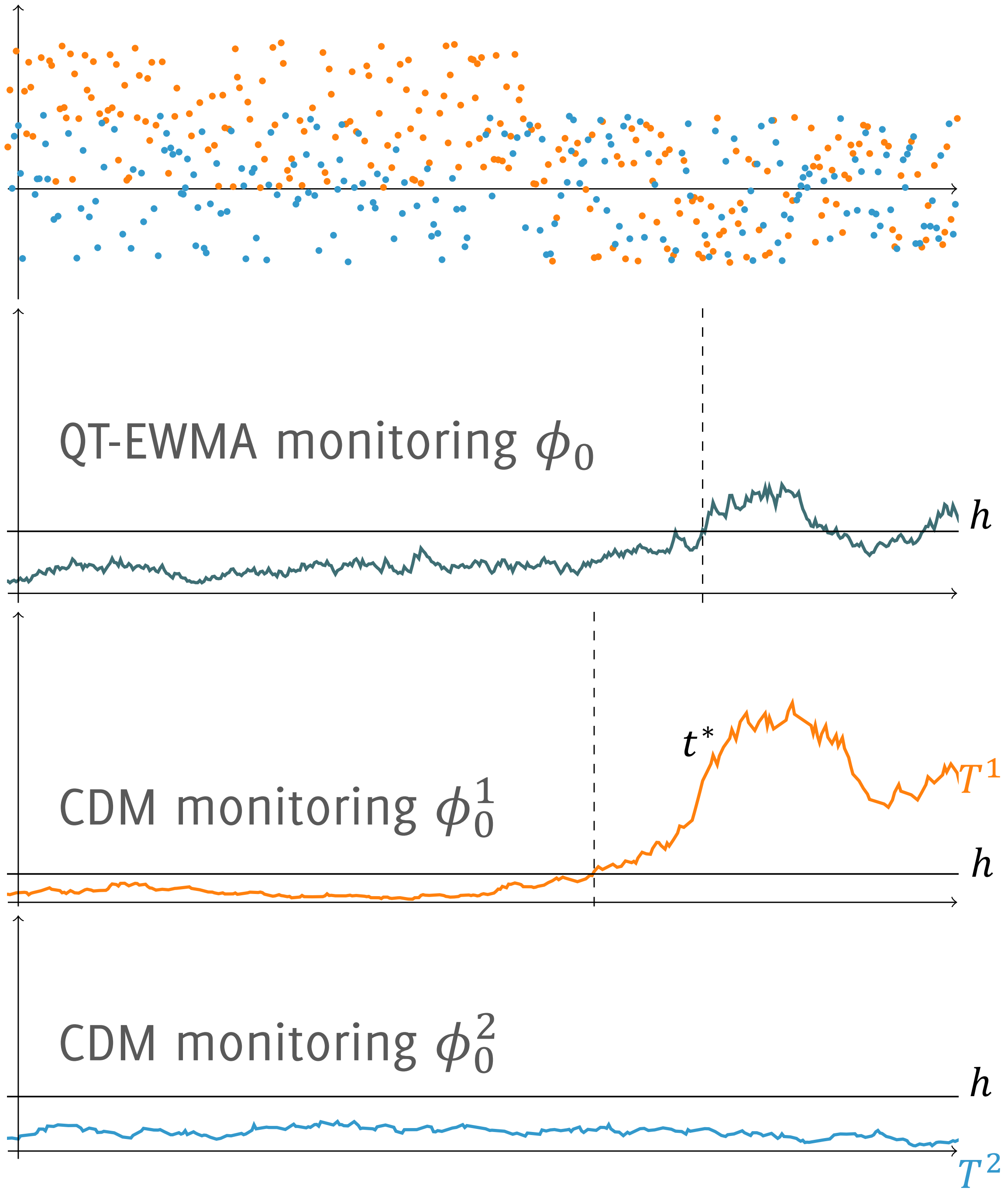
- **CDM is faster** than QT-EWMA in detecting the drift, especially when it affects a subset of classes
- CDM indicates **which class** triggered the detection



Class-wise distribution monitoring!

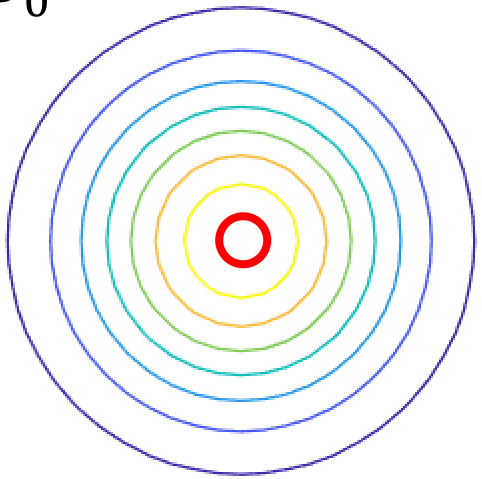
Advantages:

- **CDM is faster** than QT-EWMA in detecting the drift, especially when it affects a subset of classes
- CDM indicates **which class** triggered the detection

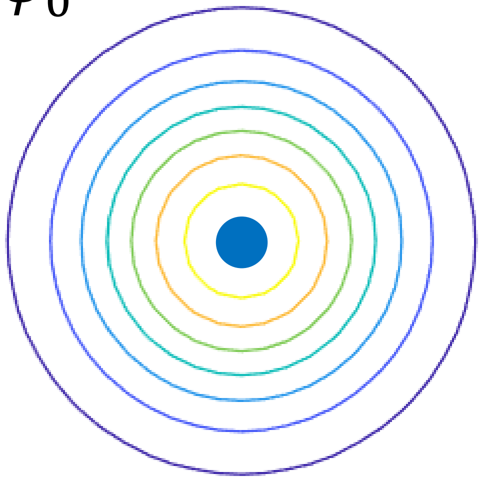


Experiments on Synthetic Data

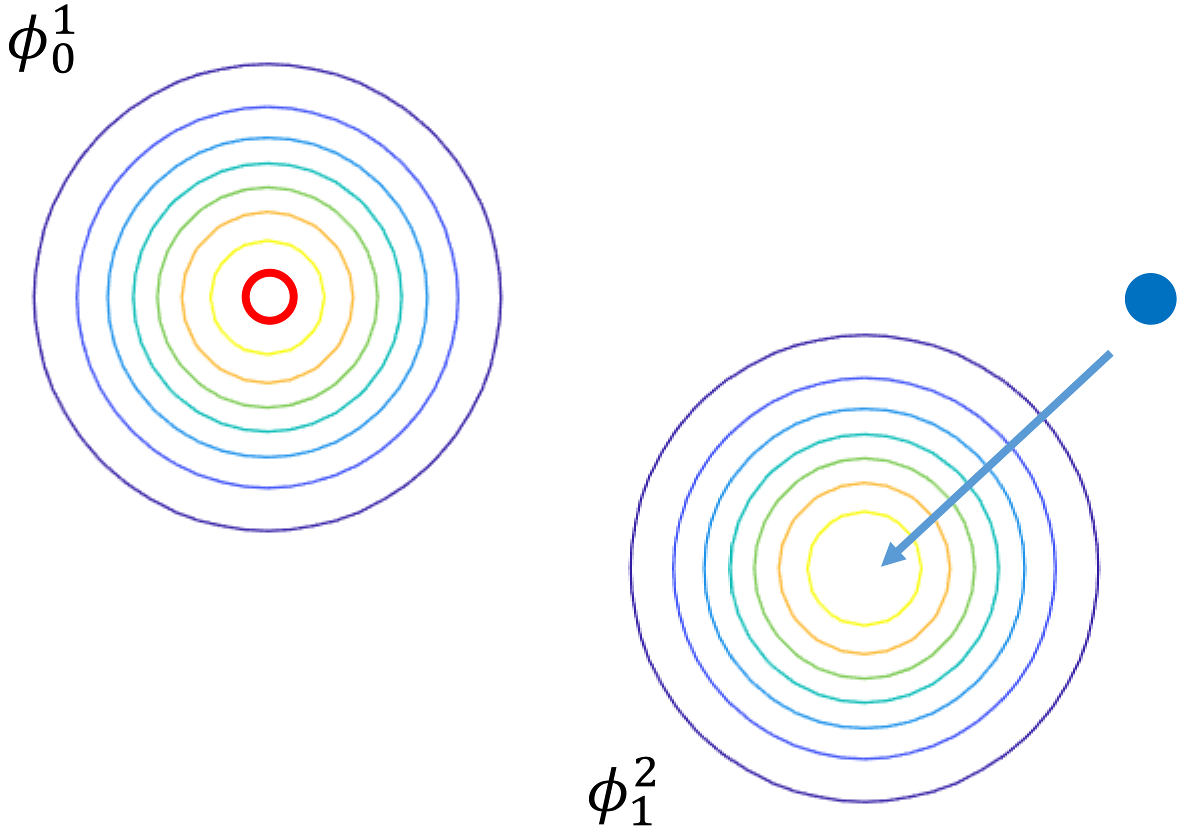
ϕ_0^1



ϕ_0^2



Experiments on Synthetic Data



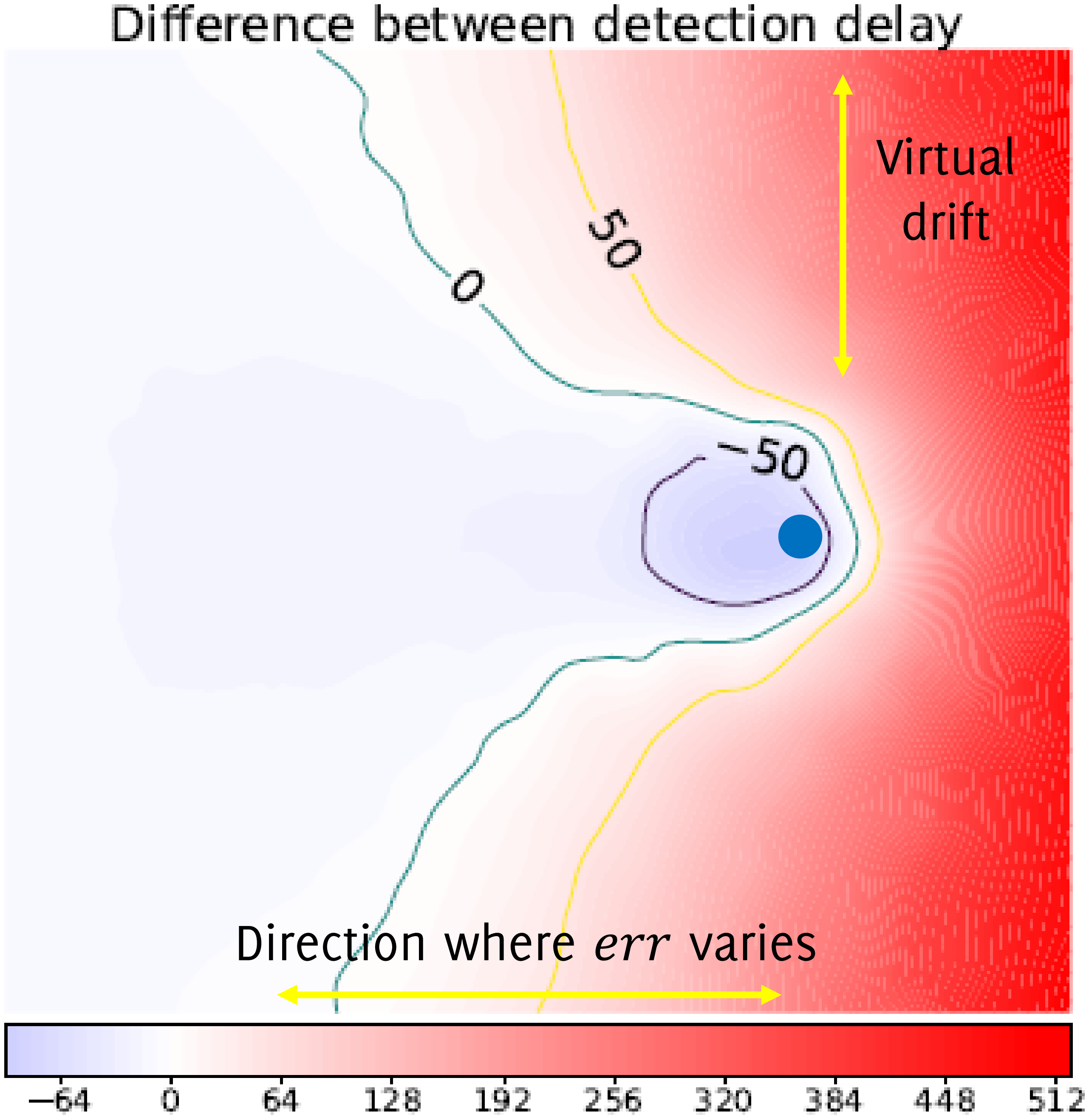
Comments

ECCD is better (blue) when

- The classification error (err) varies but $sKL(\phi_0^2, \phi_1^2)$ is low
- The difference is substantial only for small changes in $sKL(\phi_0^2, \phi_1^2)$

CDM is better (red)

- In all virtual drifts
- When drifts poorly affects the classification performance

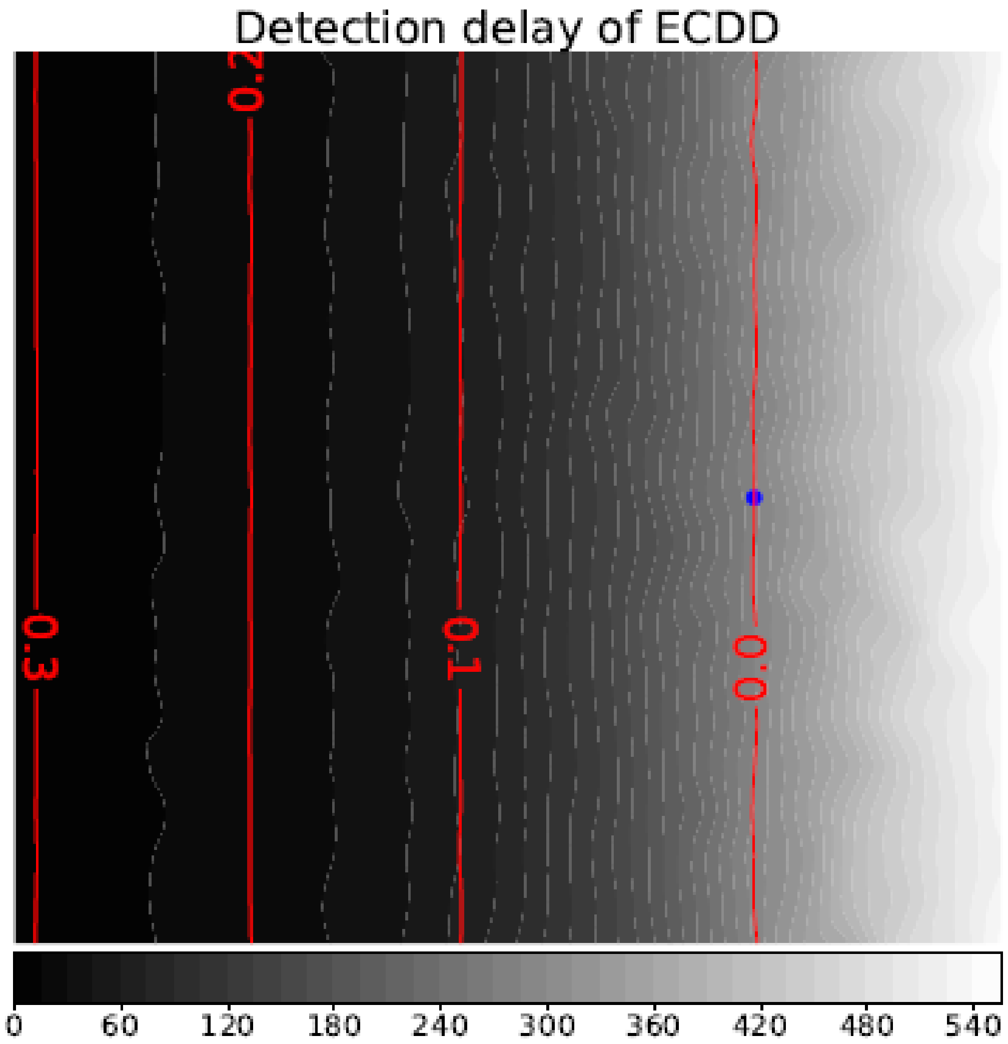


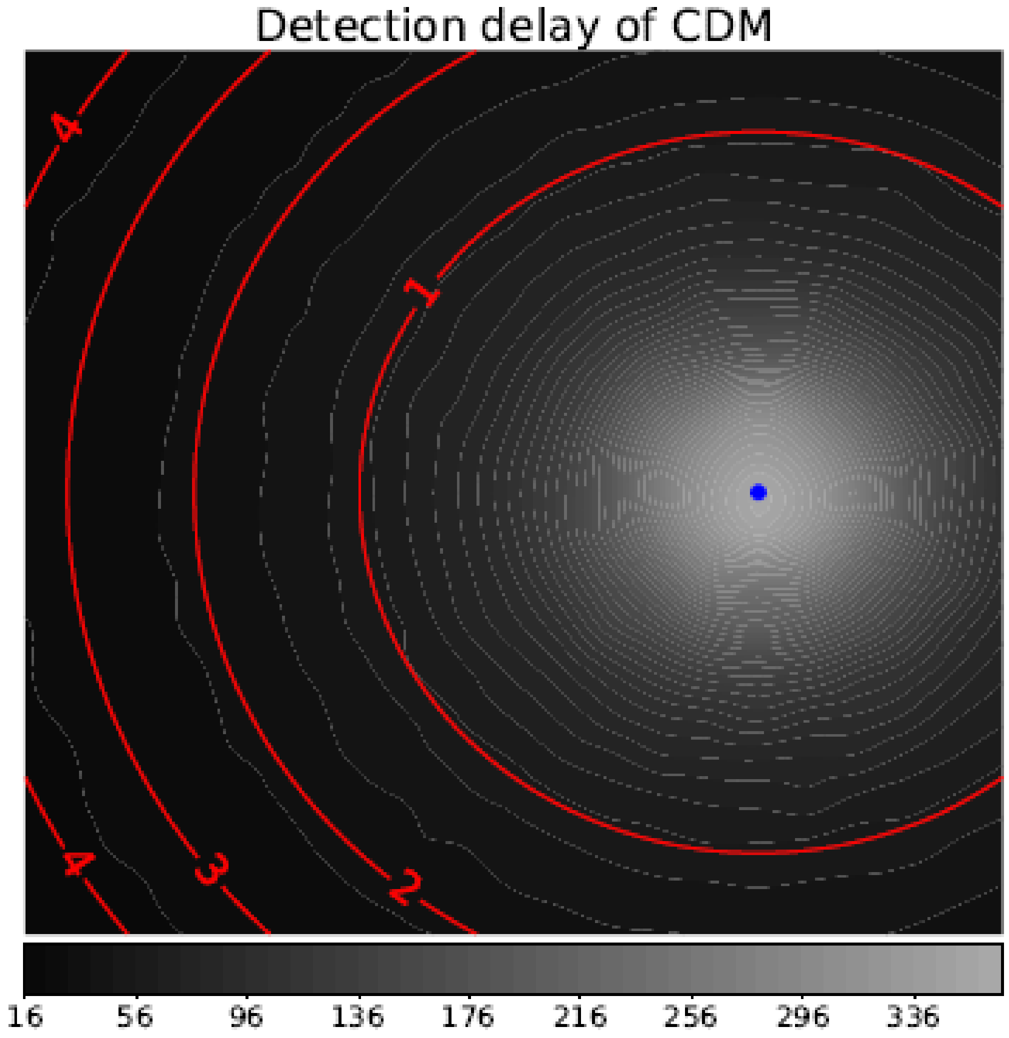
D. Stucchi, L. Frittoli, G. Boracchi *"Class Distribution Monitoring for Concept Drift Detection"* IEEE IJCNN 2022

More on Detection Delays

Contour plot is $err_0 - err_1$

Contour plot is $sKL(\phi_0^2, \phi_1^2)$





D. Stucchi, L. Frittoli, G. Boracchi *"Class Distribution Monitoring for Concept Drift Detection"* IEEE IJCNN 2022

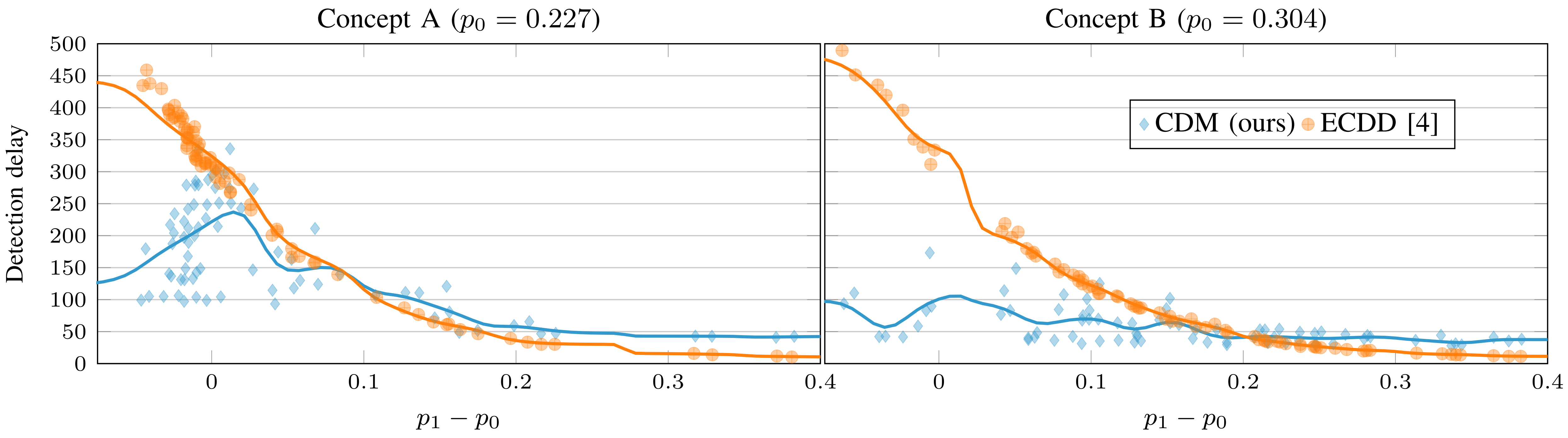
Detection Delay on Insects Datasets

Drifted Classes	ECDD [4]	Scan-B [5]	QT-EWMA [2]	CDM (ours)
1	207.98	212.53	267.73	**195.45**
2	245.85	162.58	195.44	**124.92**
3	264.27	224.99	278.57	**204.00**
4	224.91	235.87	265.96	**196.74**
1, 2	198.17	131.71	174.80	**114.44**
1, 3	172.62	169.87	223.50	**160.98**
1, 4	165.77	163.63	221.66	**145.82**
2, 3	163.66	126.56	167.55	**112.18**
2, 4	176.53	119.41	154.95	**106.49**
3, 4	210.04	169.88	218.90	**153.51**
1, 2, 3	139.29	115.01	152.91	**103.60**
1, 2, 4	148.03	103.24	141.09	**98.89**
1, 3, 4	144.81	134.83	183.41	**131.38**
2, 3, 4	132.36	**96.92**	136.90	98.57
1, 2, 3, 4	122.38	**88.86**	128.04	91.44
Avg. rank	2.416	2.356	3.524	**1.704**

Detection Delay on Insects Datasets

CDM achieves **excellent detection delays**, outperforming ECDD when the drift has **little impact on classification error** (which happens quite often!).

The difference in classification error is computed empirically on the entire dataset

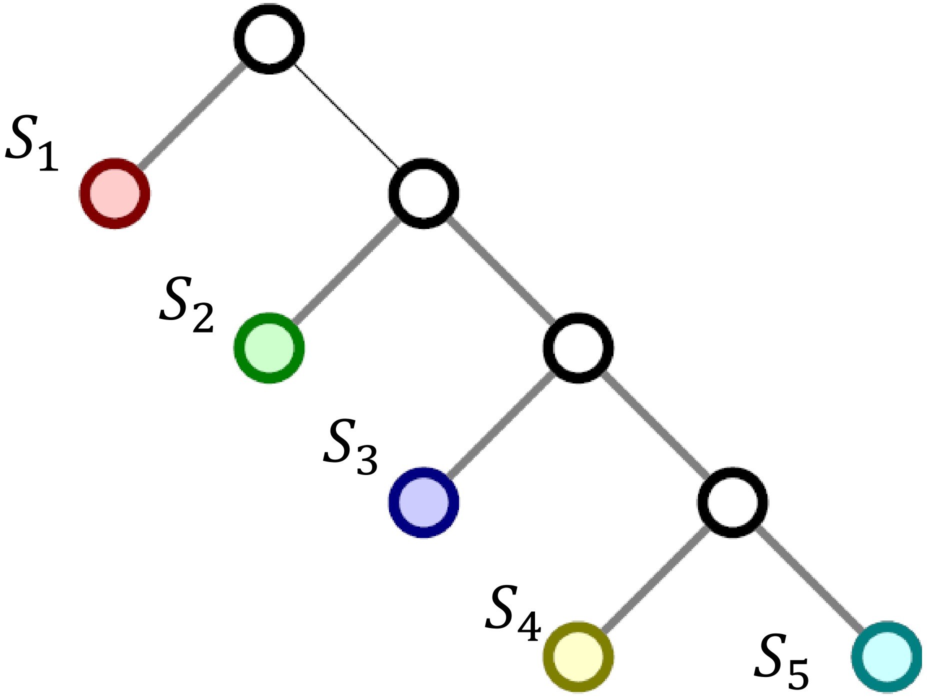


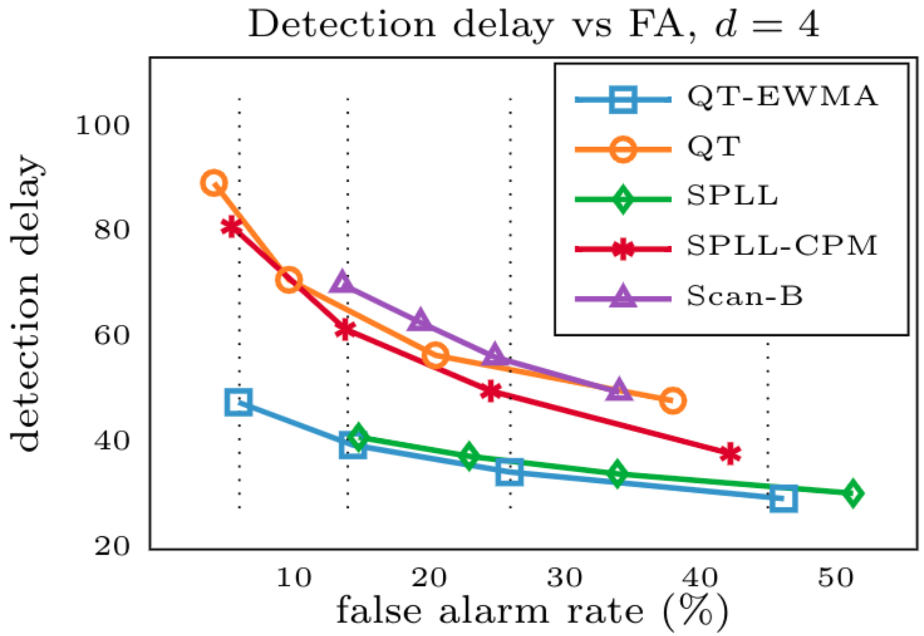
D. Stucchi, L. Frittoli, G. Boracchi *"Class Distribution Monitoring for Concept Drift Detection"* IEEE IJCNN 2022

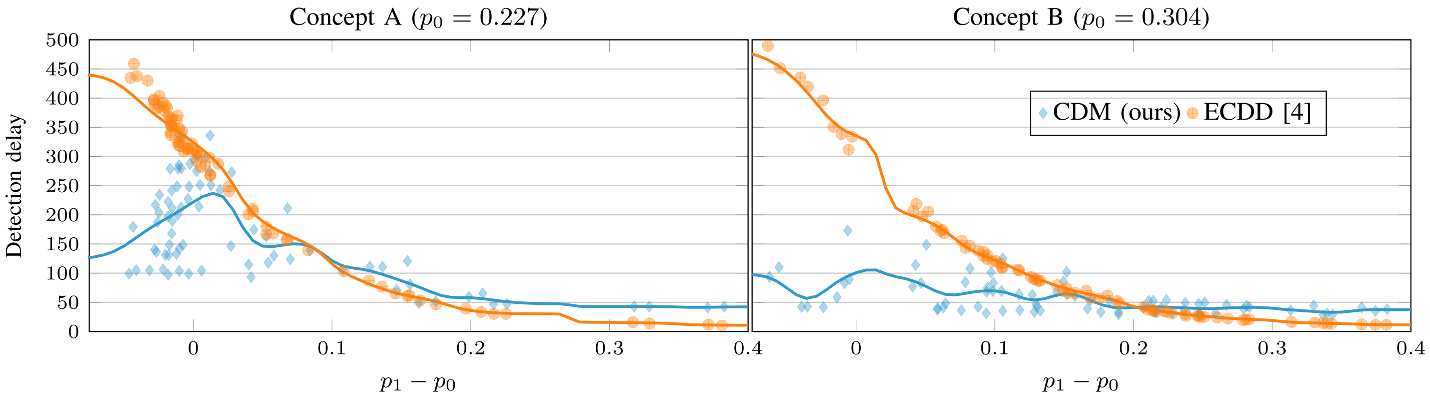
Conclusions

- QuantTree is an effective, theoretically grounded monitoring scheme for multivariate datastreams.
- Our focus is to be Nonparametric and Control “False Alarms”.
- *Histograms are flexible, design them to be comfortable for monitoring*
- QT can be customized to perform Sequential Monitoring
- …. and to other settings (W.I.P.)
- Enables new type of investigation (like CDM)

Thank you! Questions?







Giacomo Boracchi