Online Learning And Monitoring

Giacomo Boracchi, Francesco Trovò

April 27th, 2022

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

Practical & Recap

Practical Information: Exam

PhD Students: Pass/Fail

- Discussion about the exam topics
- Short oral exam where to discuss
  - A single assignment
  - **one extension** (e.g. something presented in a cited paper / some idea of yours / something you have encountered in your research or in other papers)

MSc Students: Grades (18-30L)

- Oral exam where to discuss the entire course materials
- Oral exam where to present **all** the assignments

PhD students are requested to attend at least 70% of the lectures.

Change-Detection in a Statistical Framework

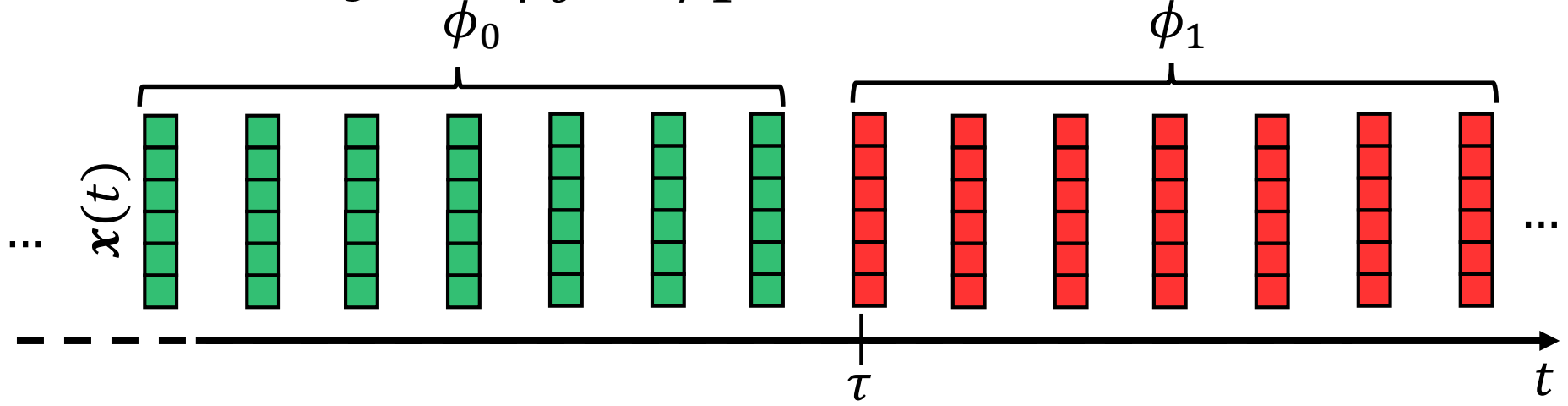
Often, the change-detection problem **boils down to:**

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 $\phi_0 \neq \phi_1$

We denote such change as: $\phi_o \rightarrow \phi_1$



Change Detection by Window Comparison

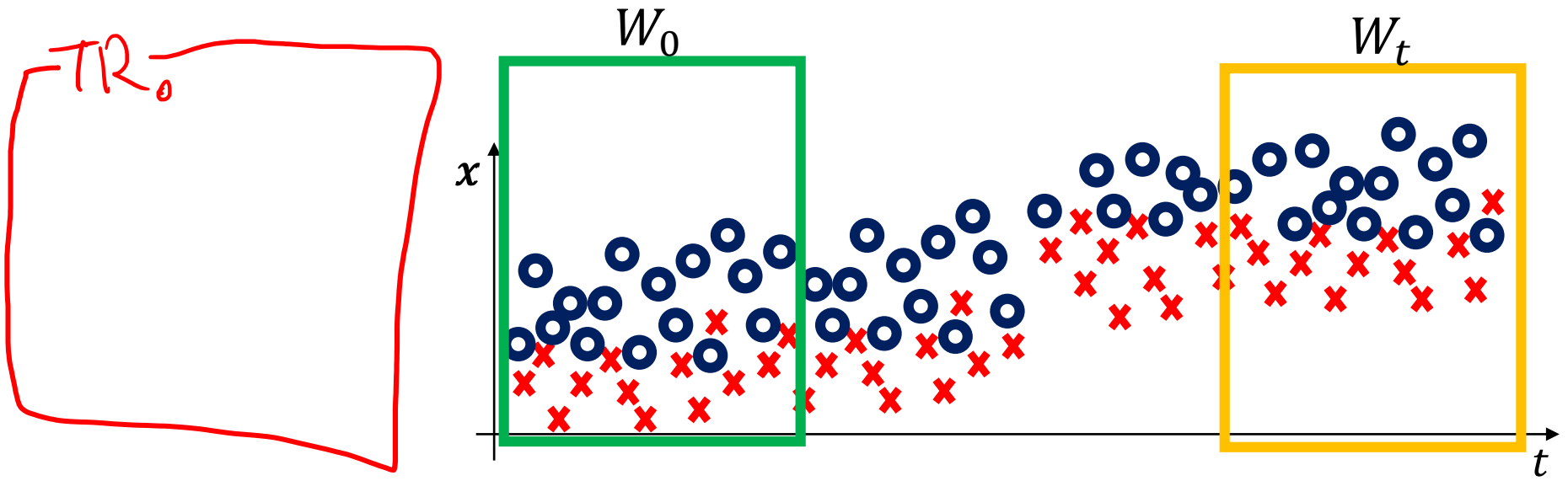
The Motivating Idea

Detect CD at time t by comparing two different windows.

In practice, one computes:

$$\mathcal{S}(W_0, W_t)$$

- W_0: reference window of past (stationary) data
- W_t: sliding window of recent (possibly changed) data
- $\mathcal{S}(\cdot,\cdot)$ is a suitable statistic over the classification error

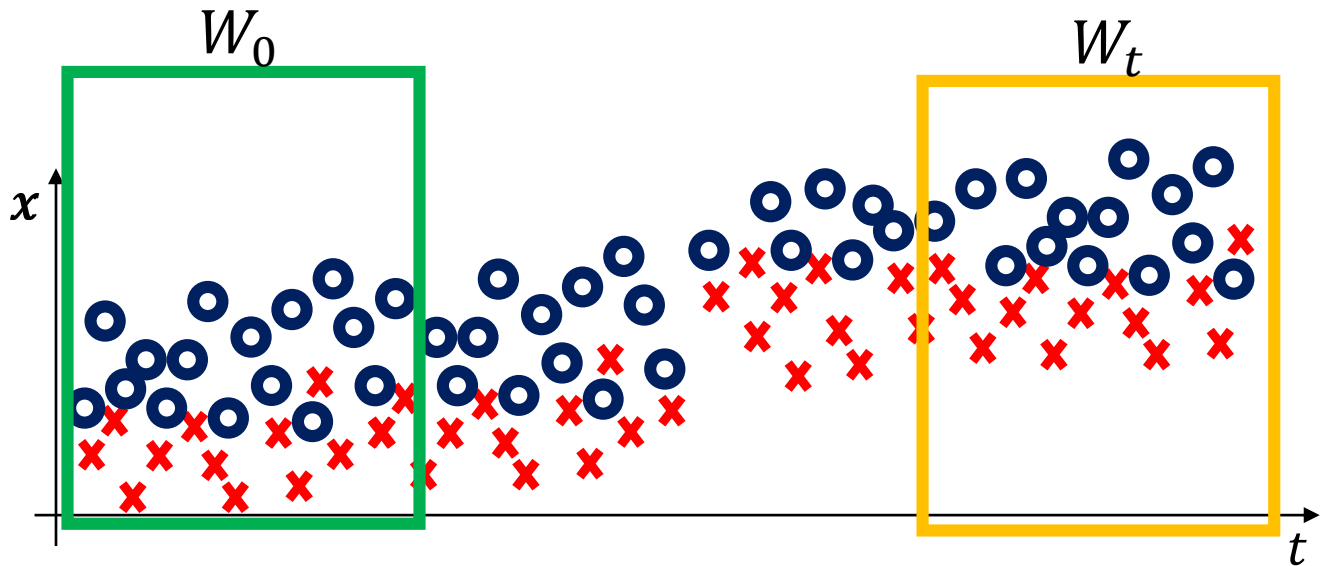


Window Comparison: Major Approaches

Comparing the classification error over W_t and W_0

- The classification error over W_0 is fixed $p_0 = \sum_{W_0} \epsilon_t$
- Compute the classification error over W_t, $p_t = \sum_{W_t} \epsilon_t$, which can be well approximated by a Gaussian distribution

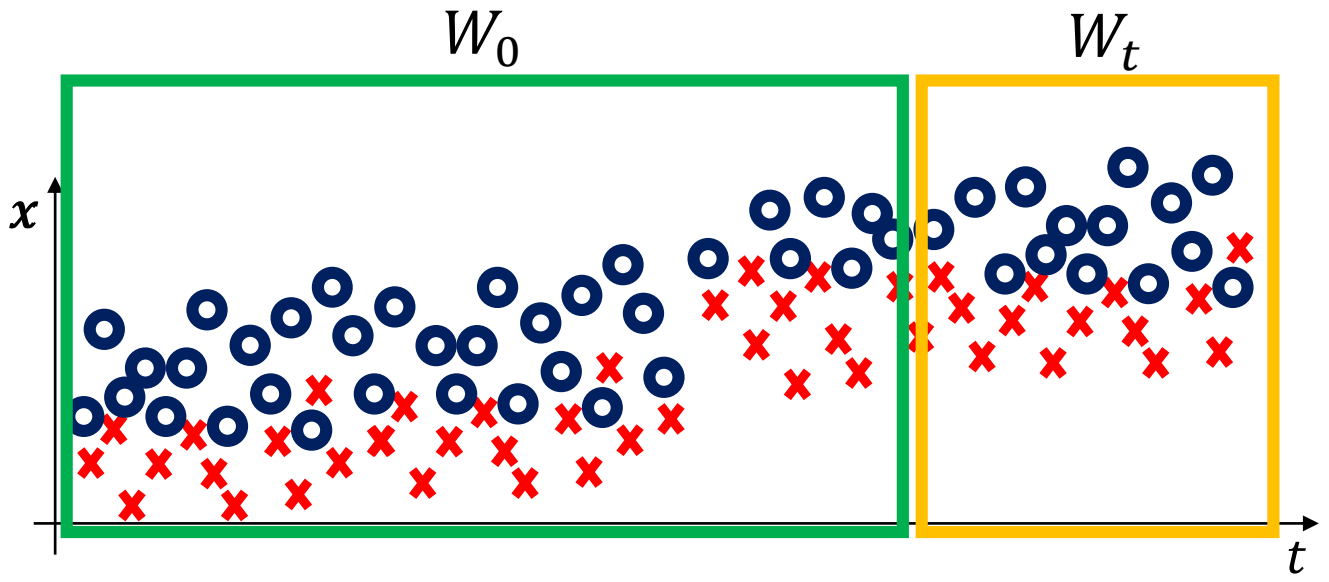
One sided t-test with $H_0 = \{p_t \leq p_0\}$ can detect concept drift



Window Comparison: Major Approaches

Comparing the classification error over W_t and W_0, using different criteria to select windows and different test statistics

STEPD: compares a recent window W_t against the past window including all the remaining samples

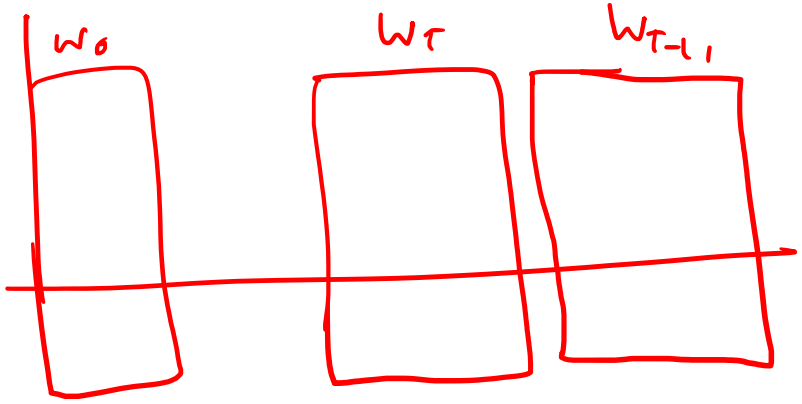


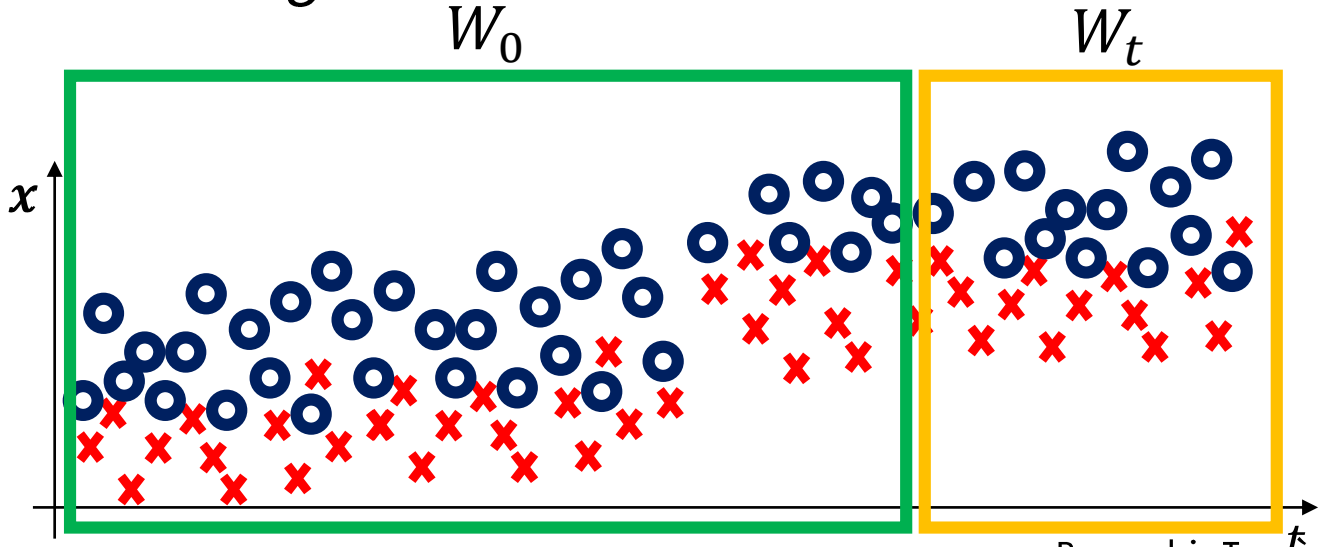
Nishida, K. and Yamauchi, K. *"Detecting concept drift using statistical testing"* In DS, pp. 264–269, 2007

Window Comparison: Major Issues

Issues: Statistical Hypothesis test are "one shot method" and H_0 holds (with control over type I errors), only when W_t are independent and identical realization of ϕ_0.

- Iterating this test even at low α leads to high FPR.
- Testing on overlapping windows W_t violates this i.i.d. assumption.
- W_0 should not include data used for training K
  p_0 might be also computed by CV.





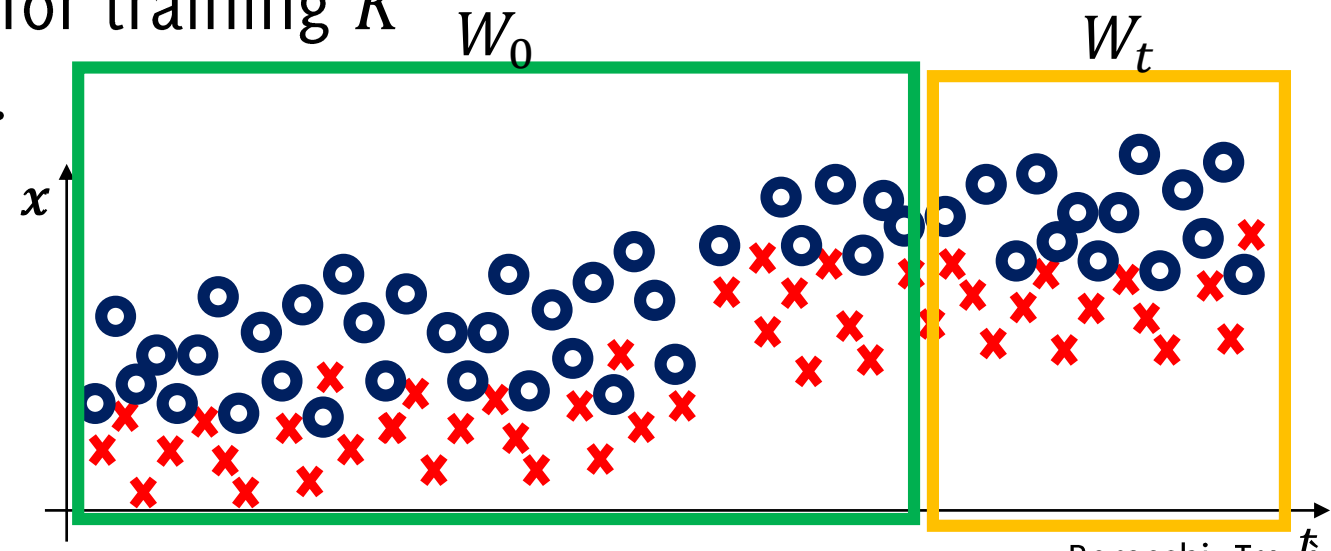
Window Comparison: Major Issues

Issues: Statistical Hypothesis test are "one shot method"

H_0 and control over type I errors hold only when W_t are independent and identical realization of ϕ_0.

- Iterating this test even at low α leads to high FPR
- Testing on overlapping windows W_t violates this i.i.d. assumption.
- W_0 should not include data used for training K p_0 might be also computed by CV.

These issues prevent a sound statistical monitoring and give rise to heuristic schemes like DDM, EDDM, STEPD

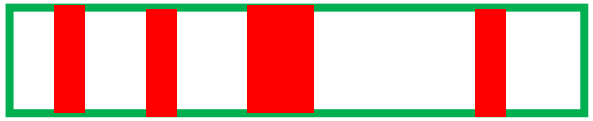


Window Comparison: Testing Exchangeability

In stationary conditions, all data are i.i.d., thus if we

- Select a training set and a test set in a window

- Select another TR and TS pair after reshuffling the two



The empirical error of the two classifiers should be the same

H_0: "equal average error of the two classifiers"

Harel M., Mannor S., El-Yaniv R., Crammer K. *"Concept Drift Detection Through Resampling"*, ICML 2014

Vovk, V., Nouretdinov, I., and Gammerman, A. *"Testing exchangeability on-line"*. In ICML 2003, pp. 768–775, 2003.

The Motivating Idea

Pro:

- There are a lot of test statistics to compare the data distribution on two different windows
- Like any other classification-error based method, these can be simply employed as wrappers to any classification algorithm

Cons:

- The biggest drawback of considering a recent window $\boldsymbol{W_t}$ **having a fixed size**, is that **subtle CD might not be detected** (this is instead the main advantage of sequential techniques)
- Defining the correct window size is very difficult
- Difficult to control False Positive Rate since often it consists of **iterating an hypothesis test over non-independent samples** (overlapping windows)

Monitoring the Input Distribution

Giacomo Boracchi, Francesco Trovò

April 27th, 2022

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

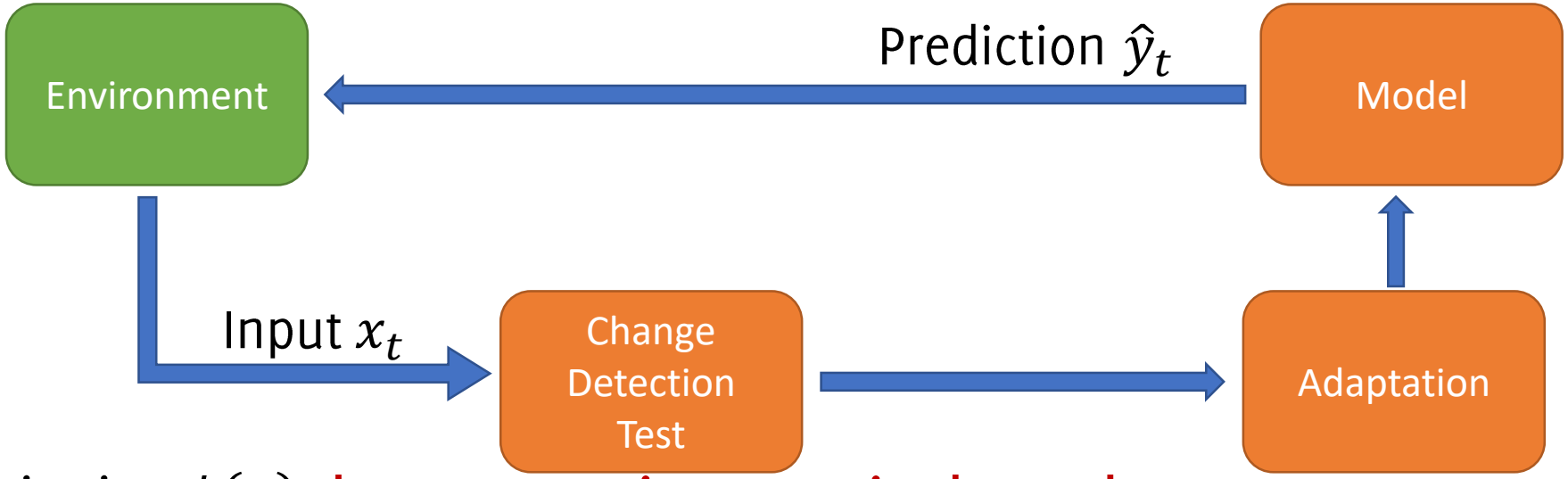
Monitoring the Input Distribution

… when ϕ_0 and ϕ_1 are both unknown

Change Detection Test

Boracchi, Trovò

Monitoring Input Distribution



Pros:

- Monitoring $\phi(\boldsymbol{x})$ **does not require supervised samples**
- Enables the detection of both **real and virtual concept drift**
- **Detection before prediction**

Cons:

- CD that does not affect $\phi(\boldsymbol{x})$ are not perceivable (e.g. classes' swap)
- In principle, changes not affecting $\phi(y|\boldsymbol{x})$ do not require reconfiguration.
- Difficult to design **sequential detection tools** when streams are **multivariate** and drawn from an **unknown distribution**

Monitoring Input by Comparing Windows

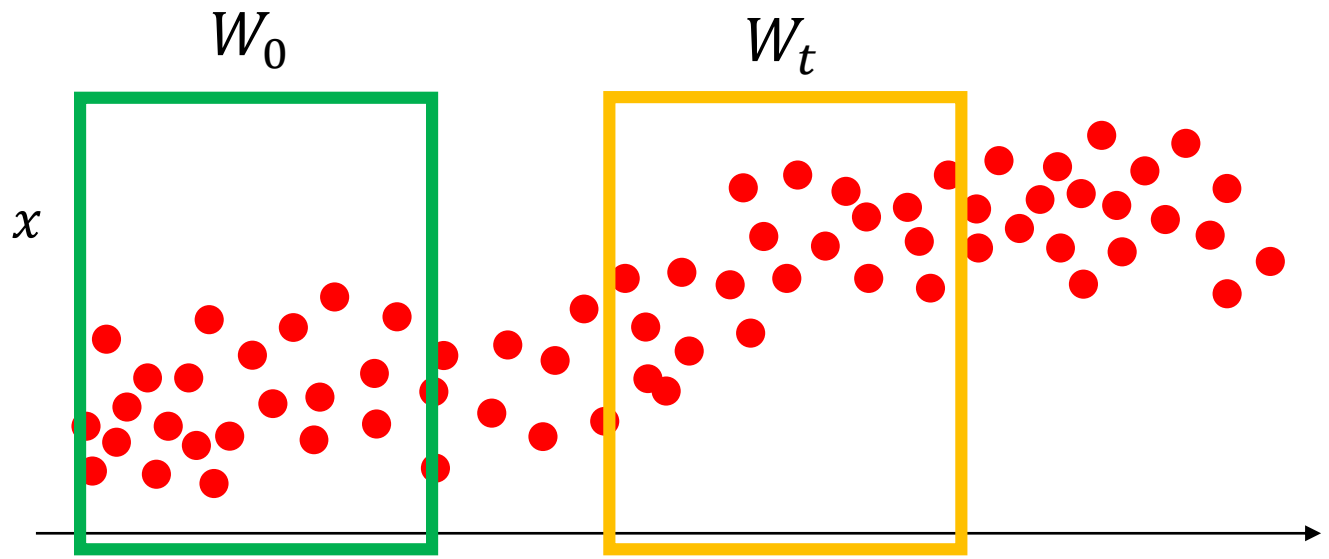
The Motivating Idea

Detect CD at time t by comparing two different windows.

In practice, one computes:

$$\mathcal{S}(W_0, W_t)$$

- W_0: reference window of past (stationary) data
- W_t: sliding window of recent (possibly changed) data
- $\mathcal{S}$ is a suitable statistic over the classification error

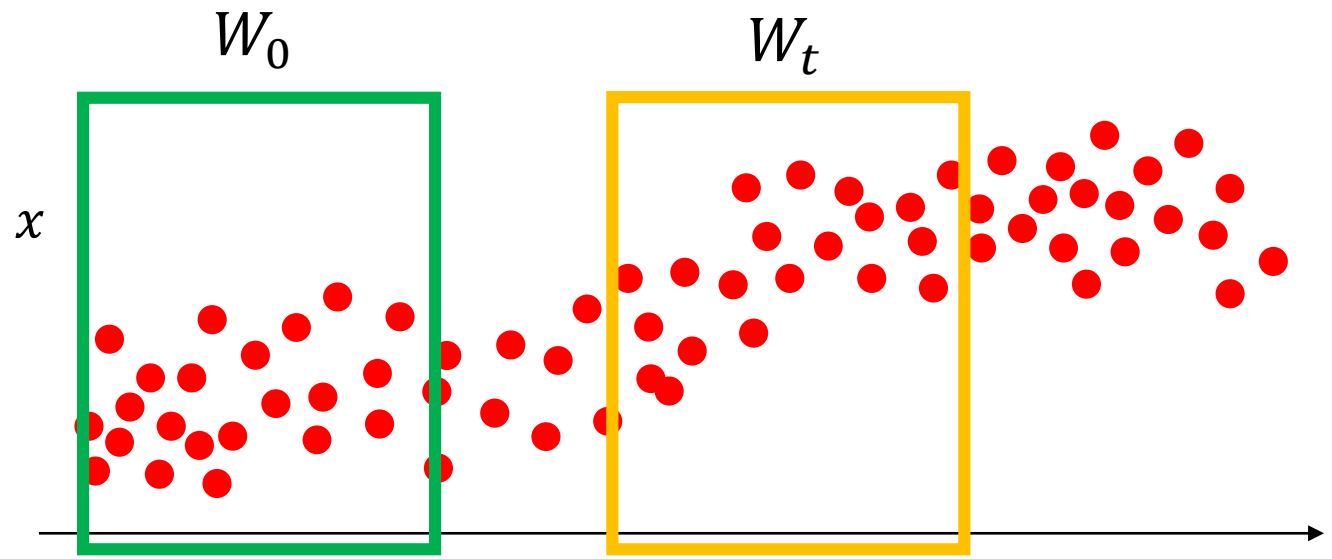


Window Comparison: Major Approaches

Hypothesis testing:

- Select W_0, a reference window from the initial concept: $W_0 \subset TR$
- As data arrives, crop a window W_t from the latest samples
- Detect concept drift by comparing an appropriate test statistic with γ

$$S(W_0, W_t) \lessgtr \gamma$$



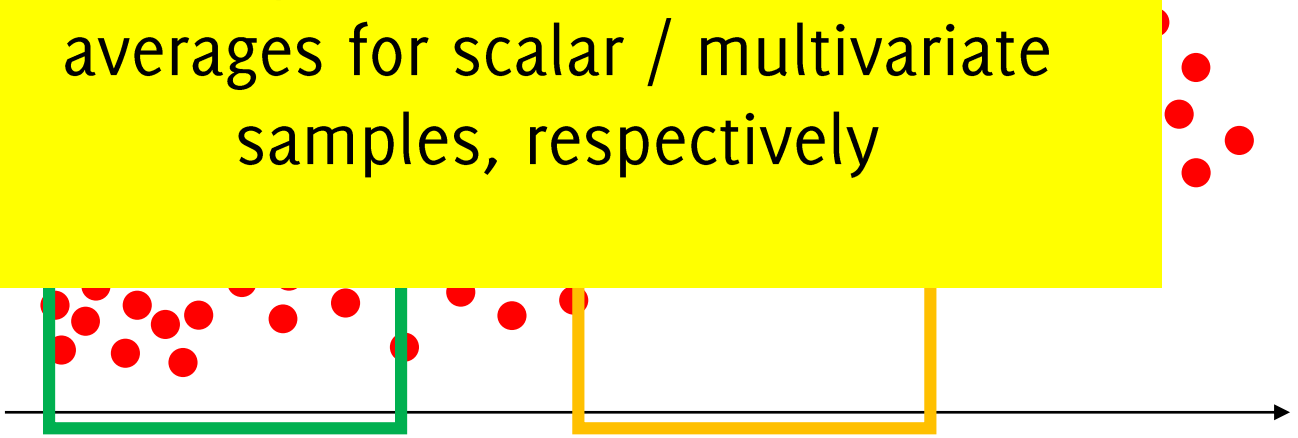
Window Comparison: Major Approaches

Hypothesis testing:

- Select W_0, a reference window from the initial concept: $W_0 \subset TR$
- As data arrives, [illegible] ples
- Detect concept [illegible] tatistic with γ

Tackle the change-detection problem as anomaly detection

e.g., t-test / Hotelling t-square for detecting shifts in the window-wise averages for scalar / multivariate samples, respectively



Window Comparison: Major Approaches

ADWIN: **Compare the averages of scalar inputs** over two adjacent windows having increasing size.

Whenever two *"large enough"* subwindows of the stream exhibit *"distinct enough"* averages, detect a change and drop the old samples in W.

$W = 101010110111111$

ADWIN: ADAPTIVE WINDOWING

- Initialize Window W
- **for each** $t > 0$ **do**
  $W \leftarrow W \cup \{x_t\}$ (add x_t to the head of W)
  - **repeat** Drop elements from the tail of W
    until $|\mu_0 - \mu_1| < \epsilon$ holds for every split
  of W into $W = [W0, W1]$
- Output μ_W

Bifet A., Gavaldà R. "Learning from time-changing data with adaptive windowing" SIAM Int. Conference on Data Mining 2007

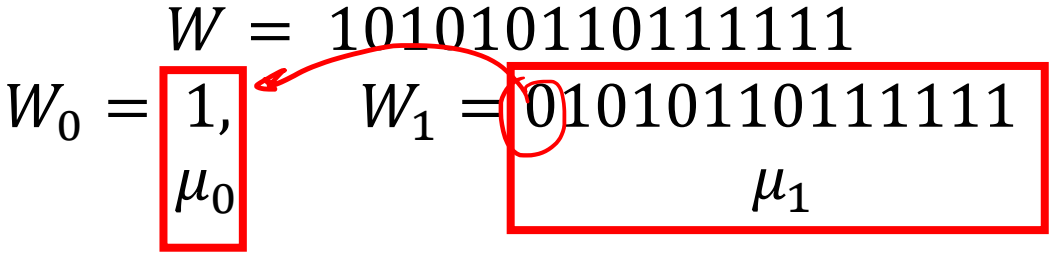
Window Comparison: Major Approaches

ADWIN: Compare the averages of scalar inputs over two adjacent windows having increasing size.

Whenever two *"large enough"* subwindows of the stream exhibit *"distinct enough"* averages, detect a change and drop the old samples in W.

ADWIN: ADAPTIVE WINDOWING

- Initialize Window W
- **for each** $t > 0$ **do**
  $W \leftarrow W \cup \{x_t\}$ (add x_t to the head of W)
  - **repeat** Drop elements from the tail of W
    until $|\mu_0 - \mu_1| < \epsilon$ holds for every split of W into $W = [W0, W1]$
- Output μ_W



Bifet A., Gavaldà R. "Learning from time-changing data with adaptive windowing" In Proc. of SIAM International Conference on Data Mining 2007

Window Comparison: Major Approaches

ADWIN: Compare the averages of scalar inputs over two adjacent windows having increasing size.

Whenever two *"large enough"* subwindows of the stream exhibit *"distinct enough"* averages, detect a change and drop the old samples in W.

ADWIN: ADAPTIVE WINDOWING

- Initialize Window W
- **for each** $t > 0$ **do**
  $W \leftarrow W \cup \{x_t\}$ (add x_t to the head of W)
  - **repeat** Drop elements from the tail of W
    until $|\mu_0 - \mu_1| < \epsilon$ holds for every split of W into $W = [W0, W1]$
- Output μ_W

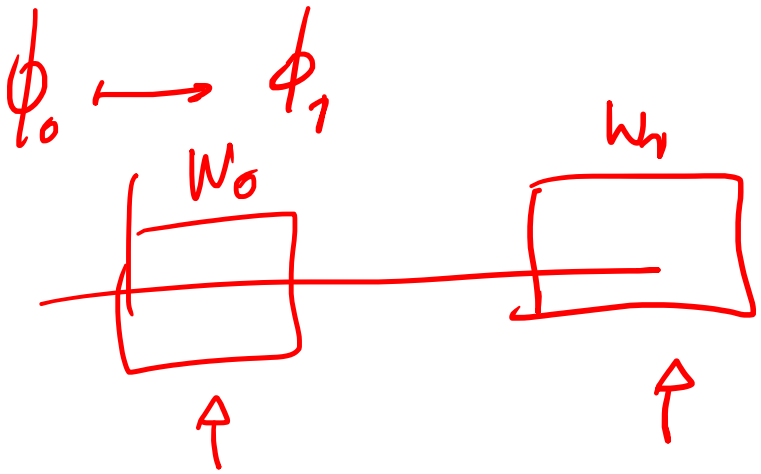
$$W = 101010110111111$$
$$W_0 = 1, \quad W_1 = 01010110111111$$
$$W_0 = 10, \quad W_1 = 1010110111111$$
$$\cdots$$
$$W_0 = 101010110, \quad W_1 = 111111$$
$$|\mu_0 - \mu_1| \geq \epsilon$$

ADWIN2: efficient variant reducing computation and memory footprint

Bifet A., Gavaldà R. "Learning from time-changing data with adaptive windowing" SIAM Int. Conference on Data Mining 2007

Window Comparison: Major Approaches

1. **Hypothesis testing**
2. **ADWIN**: Compare the averages of scalar inputs over two adjacent windows
3. Compute **empirical distributions** of raw data over W_0 and W_t and compare
   - The Kullback-Leibler divergence



T. Dasu, Sh. Krishnan, S. Venkatasubramanian, and K. Yi. *"An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams"*. In Proc. of the 38th Symp. on the Interface of Statistics, Computing Science, and Applications, 2006

Window Comparison: Major Approaches

1. **Hypothesis testing**
2. **ADWIN**: Compare the averages of scalar inputs over two adjacent windows
3. Compute **empirical distributions** of raw data over W_0 and W_t and compare
   - The Kullback-Leibler divergence
   - The Hellinger distance

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

Window Comparison: Major Approaches

1. **Hypothesis testing**
2. **ADWIN**: Compare the averages of scalar inputs over two adjacent windows
3. Compute **empirical distributions** of raw data over W_0 and W_t and compare
   - The Kullback-Leibler divergence
   - The Hellinger distance
   - The density ratio over the two windows using kernel methods (to overcome curse of dimensionality problems when computing empirical distributions)

$$\frac{\phi_1(x)}{\phi_0(x)}$$

Kawahara, Y. and Sugiyama, M. *"Sequential change-point detection based on direct density-ratio estimation"*. Statistical Analysis and Data Mining, 5(2):114–127, 2012.

Other Monitoring Schemes for Input Distribution

Change Detection Approaches

- The Change-Point Formulation
  - Parametric
  - Non-parametric
- Change-Detection by Monitoring Features / the Log-likelihood
- Change-Detection by Histograms

Change Detection in Parametric Settings: CPM

Change-Point Methods (CPM) are **sequential** monitoring schemes that **extend** traditional **parametric hypothesis tests.**

Parametric settings: ϕ_0 and ϕ_1 are known up to their parameters (θ_0 and θ_1), thus the change $\phi_0 \rightarrow \phi_1$ corresponds to a change $\theta_0 \rightarrow \theta_1$

Non-Parametric settings: Both ϕ_0 and ϕ_1 are unknown, the change $\phi_0 \rightarrow \phi_1$ is completely unpredictable

Pro: CPMs do not require training samples

Pro: They provide fixed ARL_0

Con: None of these statistics can be used on multivariate data.

Hawkins, D. M., and Zamba, K. D. *"Statistical process control for shifts in mean or variance using a changepoint formulation"* Technometrics 2005

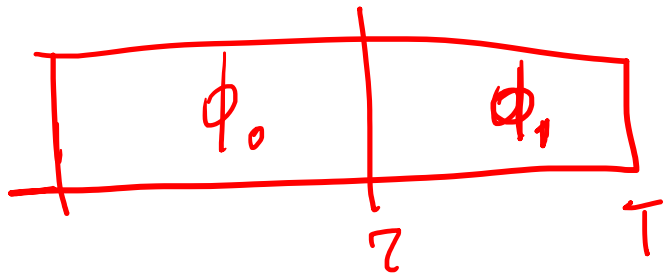
Ross, G. J. "Sequential change detection in the presence of unknown parameters". Statistics and Computing, 24(6), 1017-1030, 2014

Change Detection in Parametric settings: CPM

In Statistical Process Control, monitoring is divided in two phases:

- **Offline / Phase I:** Given a sequence $\{x_t\}$, determine whether it contains a change point τ or not. This is "one-shot test"
- **Online / Phase II:** data arrive steadily, and decision has to be taken as data flows (online).

We illustrate first the basic CPM scheme in offline monitoring, then we show how this mechanism can be **iterated to perform online change detection** (sequential monitoring).



Hawkins, D. M., and Zamba, K. D. *"Statistical process control for shifts in mean or variance using a changepoint formulation"* Technometrics 2005

Ross, G. J. "Sequential change detection in the presence of unknown parameters". Statistics and Computing, 24(6), 1017-1030, 2014

The Changepoint Model for Statistical Process Control

DOUGLAS M. HAWKINS and PEIHUA QIU

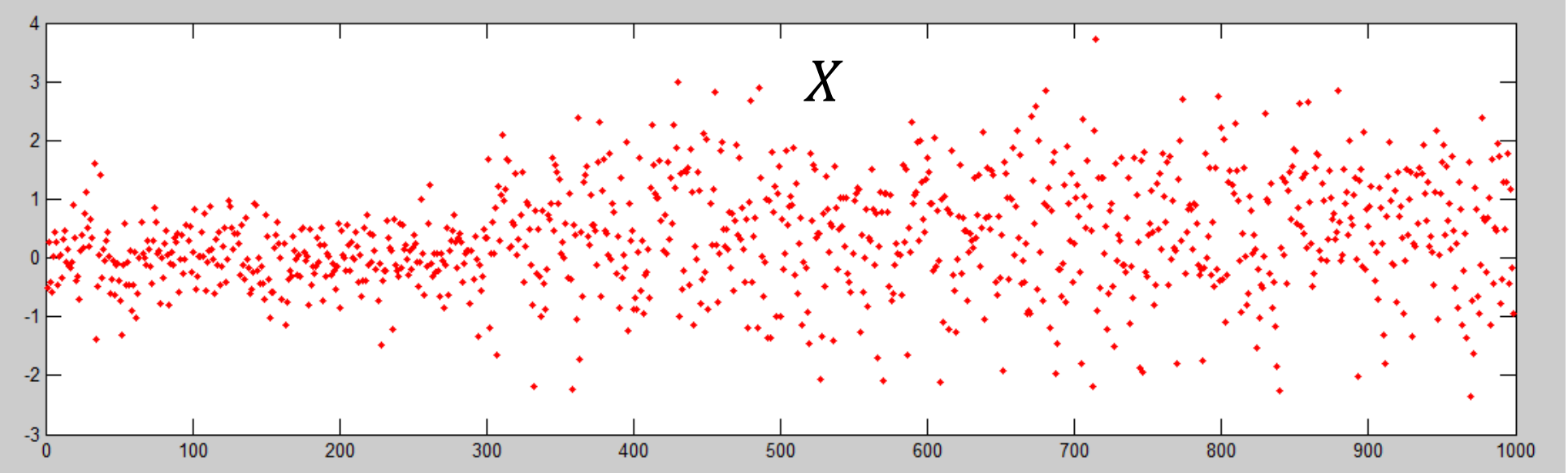
University of Minnesota, Minneapolis, MN 55455

CHANG WOOK KANG

Hanyang University, Seoul, Korea

D. M. Hawkins, P. Qiu, and C. W. Kang, "The changepoint model for statistical process control," Journal of Quality Technology, vol. 35, No. 4, pp. 355–366, 2003.

The Change Point Method (CPM)

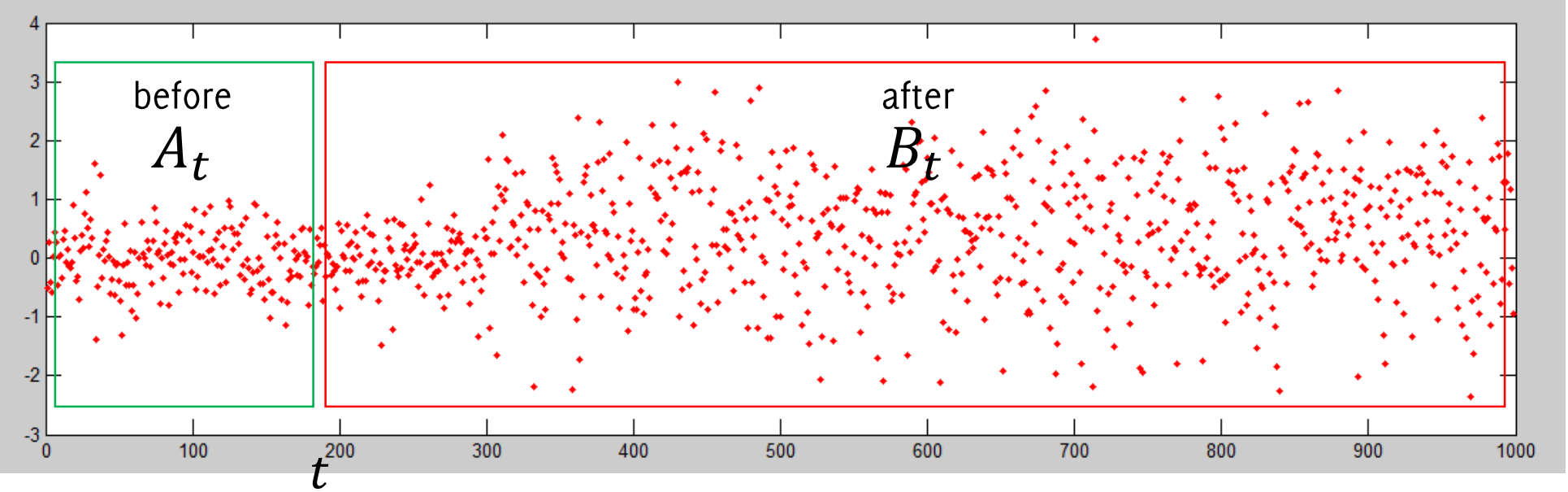


Assume a sequence X of 1000 points is given and we want to find the change point τ inside (offline analysis)

$$\boldsymbol{x}(t) \sim \begin{cases} \phi_0 & t < \tau \\ \phi_1 & t \geq \tau \end{cases}$$

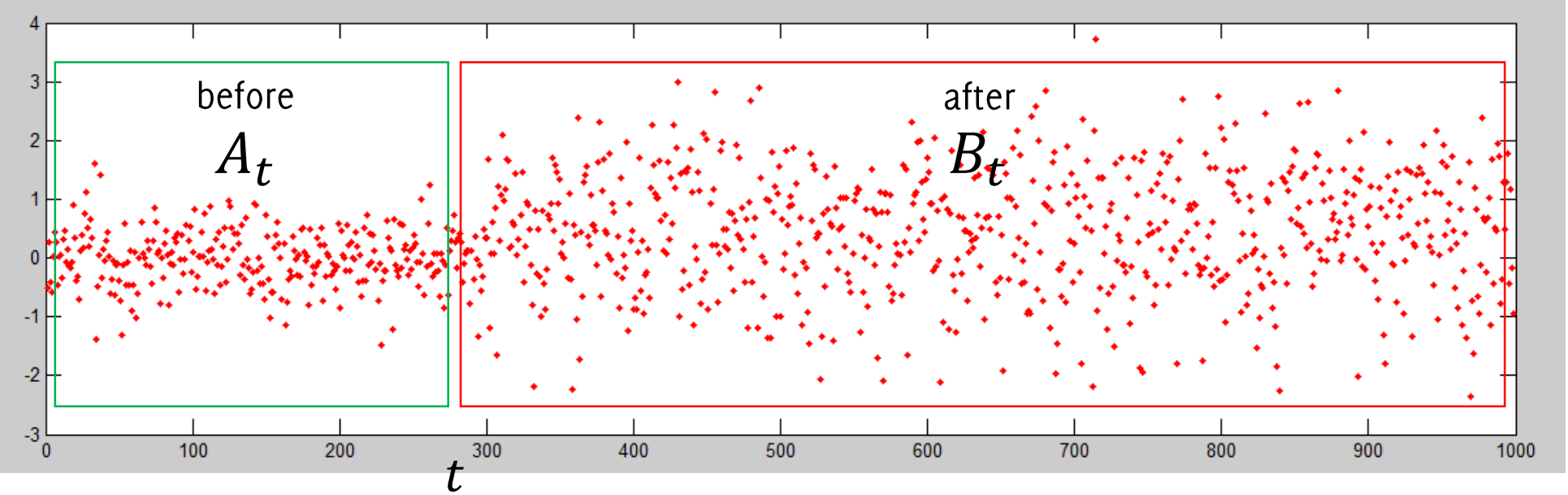
Assume we are given a statistic S_t to compare two datasets $A_t, B_t \subset X$ coming before and after t

The Change Point Method (CPM)



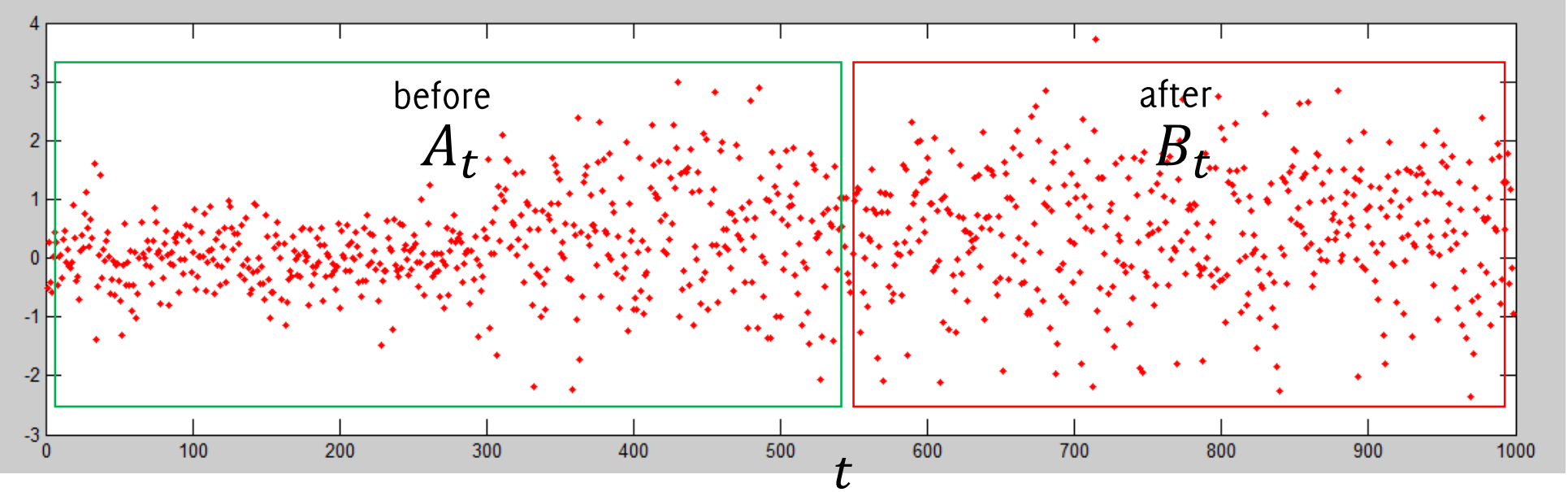
- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** $\mathcal{S}_t = \mathcal{S}(A_t, B_t)$ to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure** $t = \delta, \ldots, 1000 - \delta$ and store the value of the statistic

The Change Point Method (CPM)



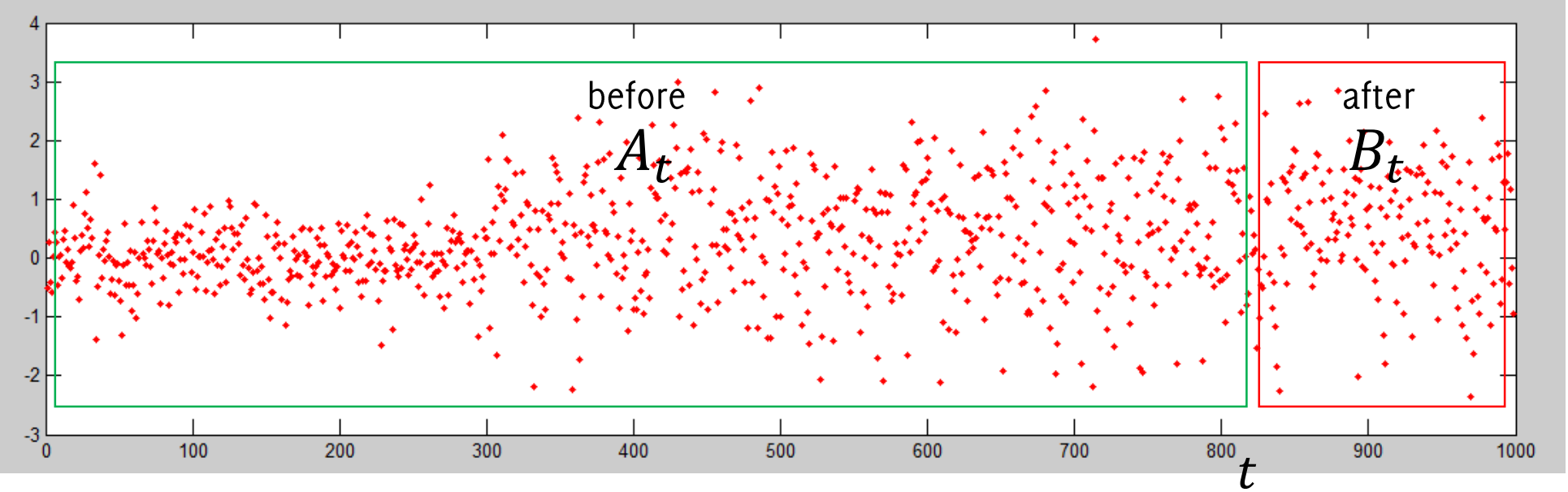
- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** $\mathcal{S}_t = \mathcal{S}(A_t, B_t)$ to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure** $t = \delta, \ldots, 1000 - \delta$ and store the value of the statistic

The Change Point Method (CPM)



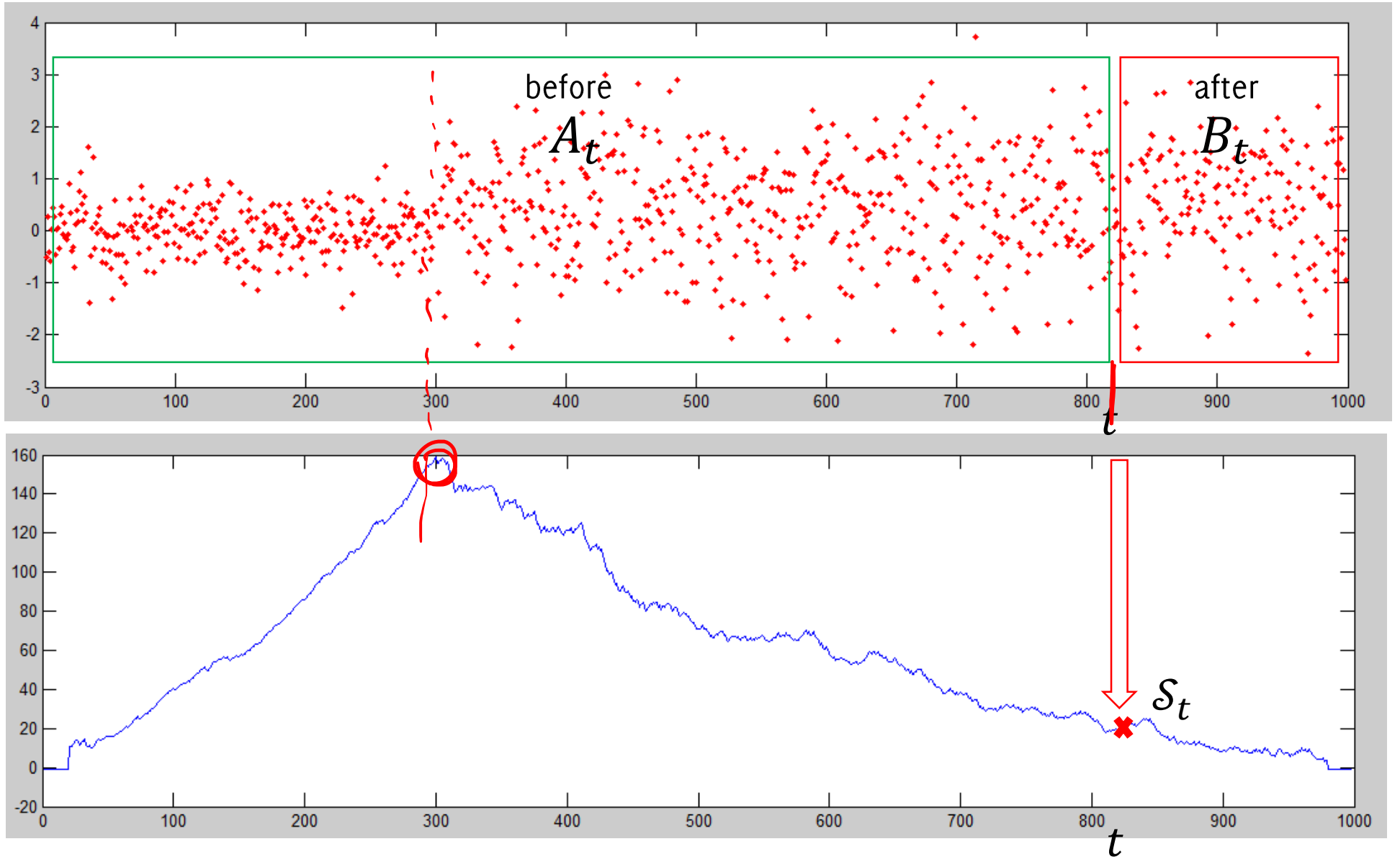
- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** $\mathcal{S}_t = \mathcal{S}(A_t, B_t)$ to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure** $t = \delta, \ldots, 1000 - \delta$ and store the value of the statistic

The Change Point Method (CPM)



- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** $\mathcal{S}_t = \mathcal{S}(A_t, B_t)$ to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure** $t = \delta, \ldots, 1000 - \delta$ and store the value of the statistic

The Change Point Method (CPM)



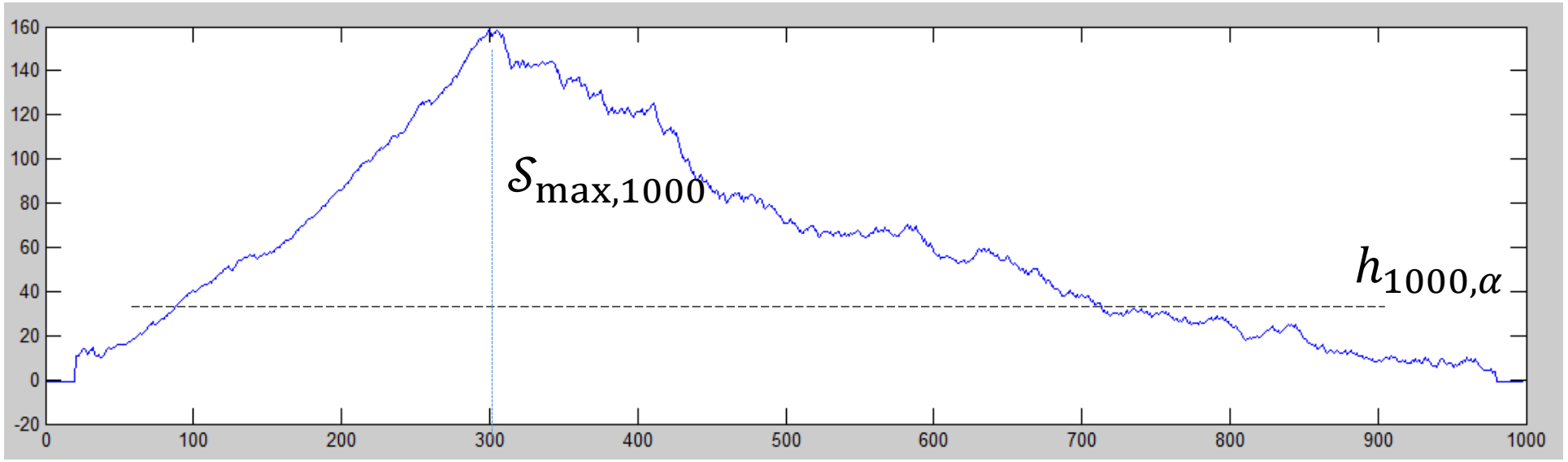
Boracchi, Trovò

The Change Point Method (CPM)

The point where the **statistic achieves its maximum** is the most likely position of the change-point

As in hypothesis testing, it possible to **set a threshold** $h_{1000,\alpha}$ for $\mathcal{S}_{\max,1000}$ by setting to α the **probability of type I errors.**

The CPM framework can be extended to online monitoring, and in this case it is possible to control the ARL_0



The CPM Formulation

Offline analysis: test all the possible splits $\forall\, t \in [1, N]$, being $N = \#X$

- Define $A_t = \{x(u), 0 \leq u < t\}$ and $B_t = \{x(u), t \leq u \leq N\}$
- Compute the test statistic

$$\mathcal{S}_t = \mathcal{S}(A_t, B_t)$$

- We claim that $\{x(t)\}_t$ contains a change point when

$$\mathcal{S}_{\max,N} = \max_t(\mathcal{S}_t) > \gamma_N$$

The threshold γ_N has to be set to control type I errors under $H_0 : X \sim \phi_0$

- The estimated change point location is

$$\hat{\tau} = \operatorname*{argmax}_t(\mathcal{S}_t) > \gamma_N$$

Threshold Computation (Offline Analysis)

Finding a threshold γ_N guaranteeing control over type I error is not trivial, as this depends on **the distribution of $\boldsymbol{S_{max,N}}$** under $X \sim \phi_0$

Rmk: the distribution of $S_{\max,N}$ is very complicated due to the **high correlation between the $\{\boldsymbol{S_{t,N}}\}$ statistics**.

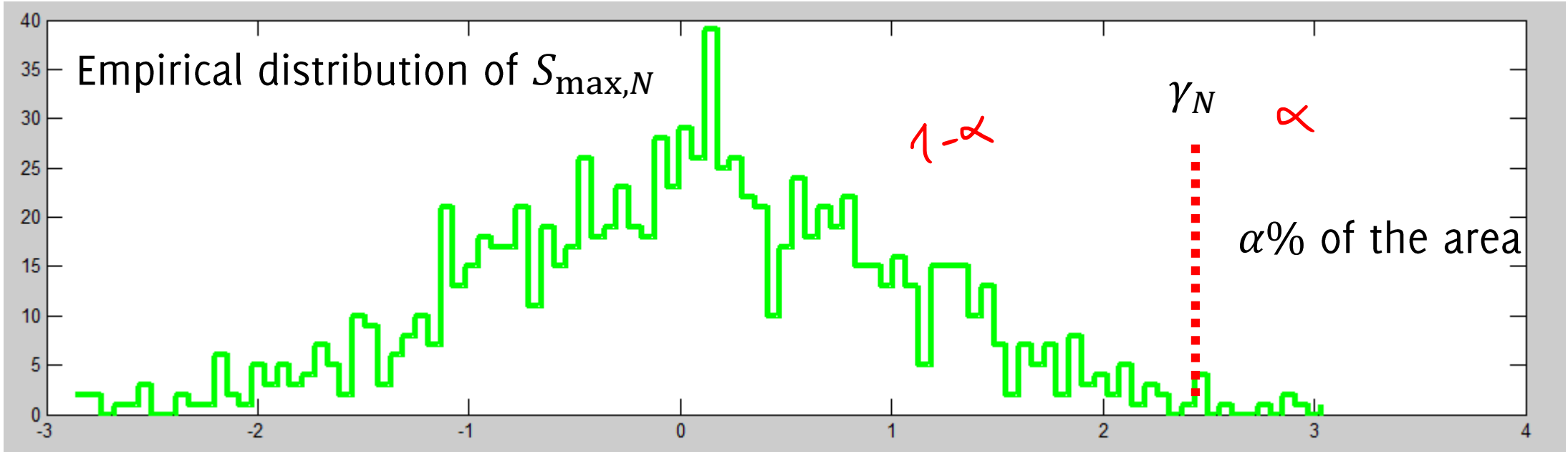
Other options:

- Bonferroni approximations are too loose (many comparisons, one per sample of the stream)
- Asymptotic bounds are only available for certain statistics S, thus wouldn't apply to all distribution ϕ_0 (and would possibly yield a coarse approximation at early monitoring stages)
- **Resort to MonteCarlo simulations**

Threshold Computation (offline analysis)

Therefore we resort to bootstrap.

- Draw many (?!) sequences $X \sim \phi_0$
- Compute the statistic $S_{\max,N}$ for each sequence and store their values
- Set the threshold as the quantile of this empirical distribution



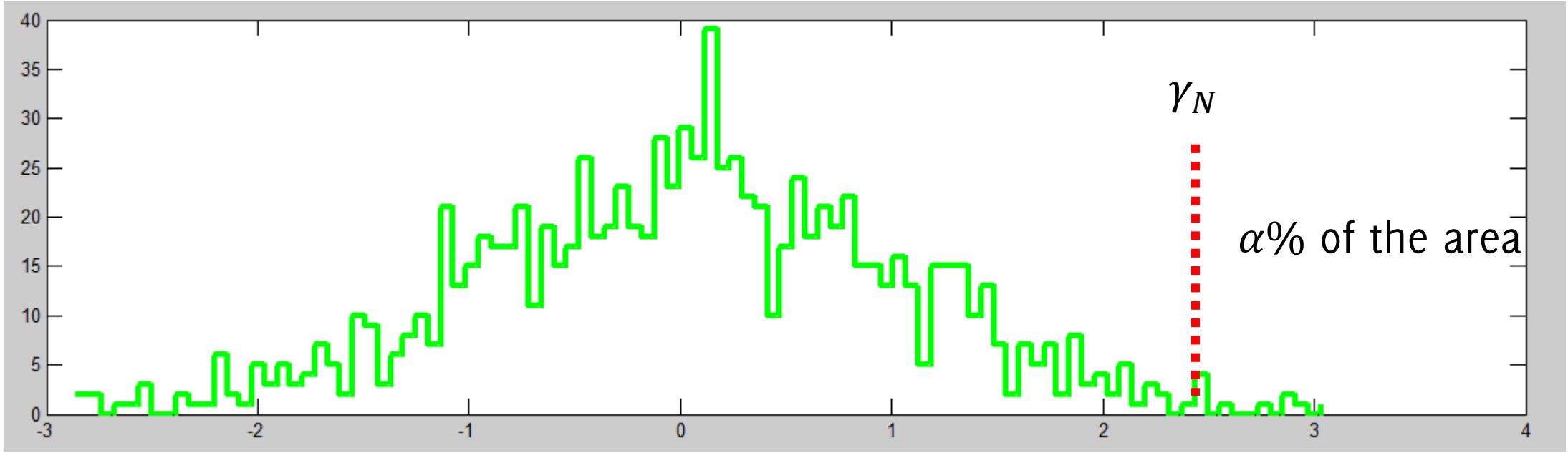
Threshold Computation

The computed thresholds depends on many factors:

- The distribution of input data ϕ_0
- The length of the sequence N
- The target FPR α

Therefore we resort to bootstrap.

- Draw many (?!) sequences $X \sim \phi_0$
- Compute the statistic $\mathcal{S}_{\max,N}$ for each sequence and store their values
- Set the threshold as the quantile of this empirical distribution



Threshold Computation

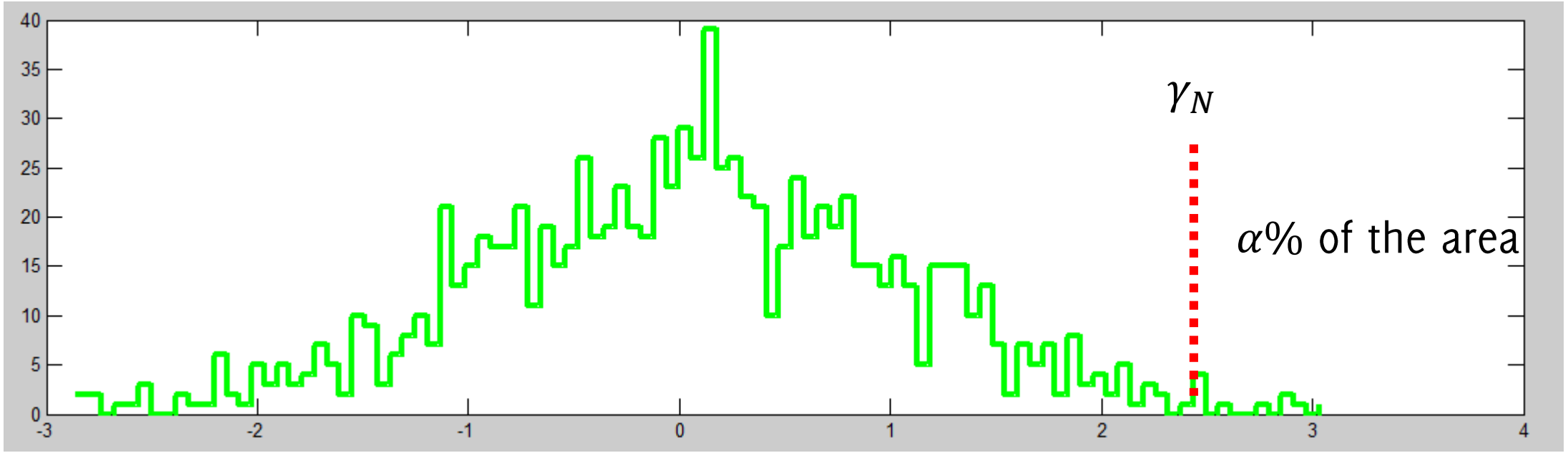
The computed thresholds depends on many factors:

- The distribution of input data ϕ_0
- The length of the sequence N
- The target FPR α

The same bootstrap procedure therefore has to be repeated for each ϕ_0 and N

Therefore we resort to bootstrap.

- Draw many (?!) sequences $X \sim$
- Compute the statistic $\mathcal{S}_{\max,N}$ for
- Set the threshold as the quantile of this empirical distribution



Nonparametric Monitoring of Data Streams for Changes in Location and Scale

Gordon J. Ross, Dimitris K. Tasoulis, and Niall M. Adams

Department of Mathematics
Imperial College London
London, SW7 2AZ, U.K.
(*gordon.ross03@imperial.ac.uk*; *d.tasoulis@imperial.ac.uk*;
n.adams@imperial.ac.uk)

G. J. Ross, D. K. Tasoulis, and N. M. Adams, "Nonparametric monitoring of data streams for changes in location and scale," Technometrics, vol. 53, no. 4, pp. 379–389, 2011.

CPM in non-parametric settings

Any statistics for HT could be used in both online and offline change-point methods. A better option would be to adopt **nonparametric statistics**, like:

- Mann-Whitney,
- Mood,
- Lepage,
- Two sample Kolmogorov-Smirnov,
- Cramer von Mises,

$S_{MK,N} = S(A_t, B_t)$ does not depend on ϕ_0

which do not require **any information** about ϕ_0 or ϕ_1.

A relevant advantage: sequences for computing the threshold can be generated by an arbitrarily distribution ψ, as the test statistic S does not depend on ϕ_0

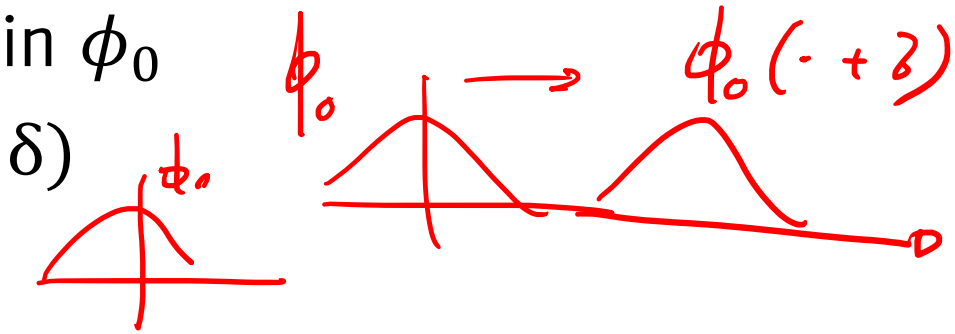
Ross, G. J., Tasoulis, D. K., Adams, N. M. *"Nonparametric monitoring of data streams for changes in location and scale"* Technometrics, 53(4), 379-389, 2012.

CPM in non-parametric settings

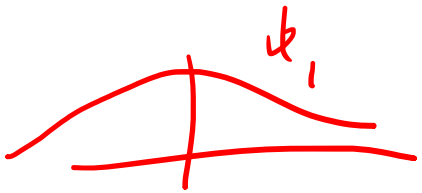
The (two samples) **Kolmogorov Smirnov and Cramer Von Mises** are very general test statistics, as they assess variations in the empirical distribution of data.

However, these "omnibus" tests have **low power**, and it is better to focus on statistics detecting specific types changes in ϕ_0

- *Location Changes: i.e.,* $\phi_1(x) = \phi_0(x + \delta)$
- *Scale Changes: i.e.,* $\phi_1(x) = \phi_0(\delta x)$



In practice it is very unlikely that $\phi_1$and $\phi_0$would differ while having the same expectation and variance.



Nonparametric Statistics for Scale and Location

Most of nonparametric statistics ranks the observations

$$rk\big(x(i)\big) = \sum_{i \neq j} I(x(i) > x(j))$$

The **Mann-Whitney** statitic to assess location changes between two sets

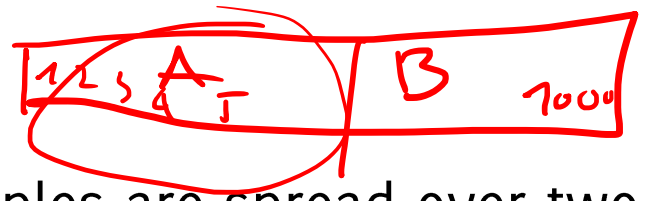
The **Mood** statistic to assess scale changes between two sets

Both Mann-Withney and Mood statistics:

- Can be used to compare two sets A, B
- **Are independent from ϕ_0 the distribution of the observations** $x(t)$

Mann-Whitney Statistic for two sets A and B

The idea



When N i.i.d. samples are spread over two sets A and B, the expectation of the sum in in A of ranks computed over $[A, B]$, should be like "the average rank" over $[A, B]$

$$E\left[\sum_{x(t)\in A} r(x(t))\right] = \#A * \frac{(\#[A,B]+1)}{2}$$

The U statistic measures how much the sum in A of ranks over $[A, B]$

$$\sum_{x(t)\in A} r(x(t))$$

deviates from $\#A * \frac{(\#[A,B]+1)}{2}$

```
m = length(A(:)); n = length(B(:));
N = m + n;

% row vector containing both dataset
D = [A(:); B(:)]';
% labels,
L = [ones(1, m) , zeros(1, n)];
[~, indx] = sort(D);
V = L(indx);
xx =[1 : size(D, 2)] ;

% U: Wilcoxon / Mann-Whitney statistic
U = xx * V';

%% compute normalization terms
mu = m * (N + 1) / 2;
sigma =  m * n * (N + 1)  / 12;

%% compute the normalized test statistic
U = abs(U - mu) / sqrt(sigma);
```

Mann-Whitney Statistic for two sets A and B

The idea

When N i.i.d. samples are spread over two sets A and B, the expectation of the sum in in A of ranks computed over $[A, B]$, should be like "the average rank" over $[A, B]$

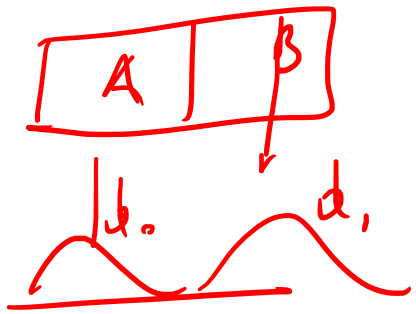
$$E\left[\sum_{x(t)\in A} r(x(t\right.$$

When $A \sim \phi_0$ and $B \sim \phi_0(\cdot - \delta)$, the ranks of elements in B will be larger ($\delta > 0$) or smaller ($\delta < 0$) than those in A

The U statistic measures how much the sum in A of ranks over $[A, B]$

$$\sum_{x(t)\in A} r(x(t))$$

deviates from $\#A * \frac{(\#[A,B]+1)}{2}$




```
m = length(A(:)); n = length(B(:));
N = m + n;

% row vector containing both dataset
D = [A(:); B(:)]';
% labels,
L = [ones(1, m) , zeros(1, n)];
                                    ;

                                    ] ;
% U: Wilcoxon / Mann-Whitney statistic
U = xx * V';

%% compute normalization terms
mu = m * (N + 1) / 2;
sigma =  m * n * (N + 1)  / 12;

%% compute the normalized test statistic
U = abs(U - mu) / sqrt(sigma);
```

Mood Statistic for two sets A and B

The idea

When N i.i.d. samples are divided in two sets A and B, then

$$E[r(x(t))] = \frac{N+1}{2}$$

the expected rank of each point under H_0 = "both sets are identically distributed" is $(N+1)/2$

M measures the deviation of ranks from this expectation

```
m = length(A(:)); n = length(B(:));
N = m + n;

% row vector containing both dataset
D = [A(:); B(:)]';

% compute the rank
[vs, vi] = sort(D);
[x, r] = sort(vi);

% Mood Statistic,
M = sum((r(1 : m) - (N + 1) / 2).^2);

% Expectation of Mood Stats
mu = m * (N^2 - 1) / 12;

% Standard deviation of Mood Stats
sigma =  m*n*(N + 1)*(N - 2)* (N+2) / 180;

%% compute the normalized test statistic
M = abs((M - mu)) / sqrt(sigma);
```

Mood Statistic for two sets A and B

The idea

When N i.i.d. samples are divided in two sets A and B, then

$$E[r(\quad)] = \frac{N+1}{2}$$

the expected r

H_0 that both s

distributed is (

M measures the deviation of ranks from this expectation

When $A \sim \phi_0(\cdot)$ and $B \sim \phi_0(\delta\,\cdot)$, the ranks of elements in B will be more extreme ($\delta > 1$) or condensed ($\delta < 1$) than those in A.
This results in a larger/smaller variance of ranks, which corresponds to larger values of M statistics

```
m = length(A(:)); n = length(B(:));
N = m + n;

% row vector containing both dataset
D = [A(:); B(:)]';

% compute the rank

                                    (N + 1) / 2).^2);

                                    d Stats
mu = m * (N^2 - 1) / 12;

% Standard deviation of Mood Stats
sigma =  m*n*(N + 1)*(N - 2)* (N+2) / 180;

%% compute the normalized test statistic
M = abs((M - mu)) / sqrt(sigma);
```

And both Location and Scale Changes?

In practice we don't know if ϕ_0 and ϕ_1 would differ because of location or scale changes

Using Mood and Mann-Whitney in parallel makes difficult to control the ARL_0 (or type I error in the offline scenario)

Better to monitor location and scale jointly: use the **Lepage Test statistic**

$$L = U^2 + W^2$$

CPM for Online Monitoring

Observations arrive steadily,

$$x(1), \ldots, x(N), \ldots$$

possibly forming an infinite stream

At each new arrival, a Change-Point Method (CPM) assesses if the distribution of the observations differs from the previous samples.

The primary issue is the **detection**, but the CPM monitoring scheme performs also the **estimation of change point location**, once the detection is signalled.

In fact any online CPM returns

- $\hat{T}$, the time instant when the change is detected,
- $\hat{\tau}$, the estimate of the change time-instant

Two Issues in CPMs for Online Monitoring

In principle, one may **iterate the offline approach** presented before – at each new arrival.

Two issues:

- How to compute the thresholds?
- Iterating CPM becomes time and resources demanding..

Even if we compute γ_N for the offline analysis, these thresholds would not be appropriate for online analysis

Threshold Computation: Online CPM

Quantiles of test statistic $\mathcal{S}_{\max,t}$ used for offline analysis cannot be used, since γ_t **has to be set controlling the conditional probability** that

$$P\left(\mathcal{S}_{\max,t} > \gamma_t \mid \mathcal{S}_{\max,t-1} < \gamma_{t-1}, \dots, \mathcal{S}_{\max,1} < \gamma_1\right) < \alpha$$

Still, one may resort to numerical simulations to compute them **in a sequential manner**.

A few methods can set the false alarm probability (FAP) to be the same in each point, that is, $P(\mathcal{S}_{\max,t} > \gamma_t \mid x \sim \phi_0) = \alpha$ for all t,

then the ARL_0 relates to α as

$$\alpha = \frac{1}{ARL_0}$$

Threshold Computation: Online CPM

For each desired value of α:

Generate a dataset D of one million streams containing 5000 points drawn from an **arbitrary distribution** $\boldsymbol{\psi}$ (e.g. $N(0,1)$). This is feasible when $\mathcal{S}$ is a distribution-free statistic, as this does not depend on ϕ_0.

- For $t = 1, \ldots$
  - Evaluate the statistics over each stream in D and compute $\mathcal{S}_{\max,t}$
  - Compute γ_t over sequences in D such that
    $P\left(\mathcal{S}_{\max,t} > \gamma_t \mid \mathcal{S}_{\max,t-1} < \gamma_{t-1}, \ldots, \mathcal{S}_{\max,1} < \gamma_1\right) < \alpha$
  - Remove from D streams where $\mathcal{S}_{\max,t} > \gamma_t$
- Interpolate the values of γ_t by some parametric function of t, to "fill in possible gaps" and to smooth all the estimates.

Threshold Computation

Here is an example of polynomial cofficients modeling γ_t

Table 1. Polynomial approximation of h_t as a function of $\gamma = 1/t$

		Coefficient polynomial approximation of h_t				
ARL_0	CPM type	Constant	t^{-1}	t^{-3}	t^{-5}	t^{-7}
370	Mood	3.27×10^0	-1.69×10^0	2.03×10^2	-2.85×10^6	2.70×10^9
	LP	1.53×10^1	-4.42×10^1	-3.26×10^3	-1.32×10^7	1.47×10^{10}
500	Mood	3.37×10^0	-1.59×10^0	-3.64×10^3	5.16×10^6	-3.04×10^9
	LP	1.62×10^1	-5.03×10^1	-1.32×10^4	2.21×10^6	7.62×10^9
1000	Mood	3.60×10^0	-3.55×10^0	5.79×10^2	-2.33×10^6	8.69×10^8
	LP	1.82×10^1	-6.43×10^1	-7.72×10^4	1.50×10^8	-1.01×10^{11}
10,000	Mood	4.25×10^0	-8.28×10^0	5.90×10^2	-6.77×10^6	4.98×10^9
	LP	2.45×10^1	-1.50×10^2	-1.20×10^3	-1.85×10^7	6.66×10^9
20,000	Mood	4.44×10^0	-1.21×10^1	8.05×10^3	-1.53×10^7	8.46×10^9
	LP	2.64×10^1	-1.87×10^2	7.35×10^4	-1.70×10^8	1.04×10^{11}

Boracchi, Trovò

Online Monitoring: Ranks Computation

Ranks computation requires storing all the data in memory

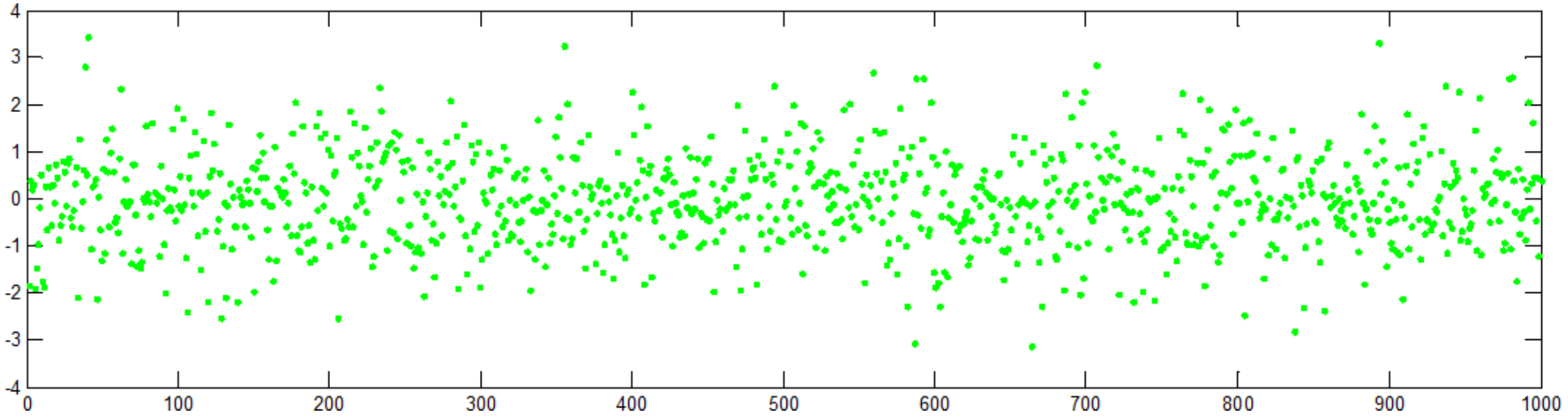
Also time requirement grows at each new observation

This is usually infeasible when working with data streams.

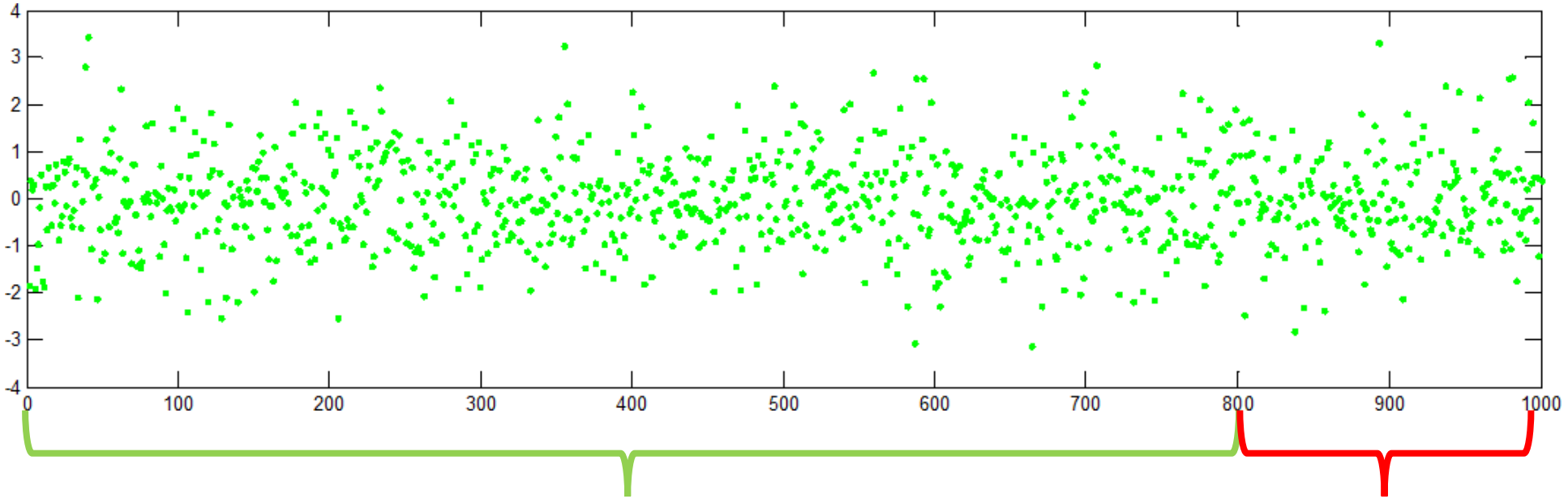
Solution: discretization of the older part of the stream

- Past data are stored in an histogram (ranks computed from quantized values)
- A window over the most recent data is kept to process these accurately
- Introduce an upper bound in memory and time requirements

Data Quantization



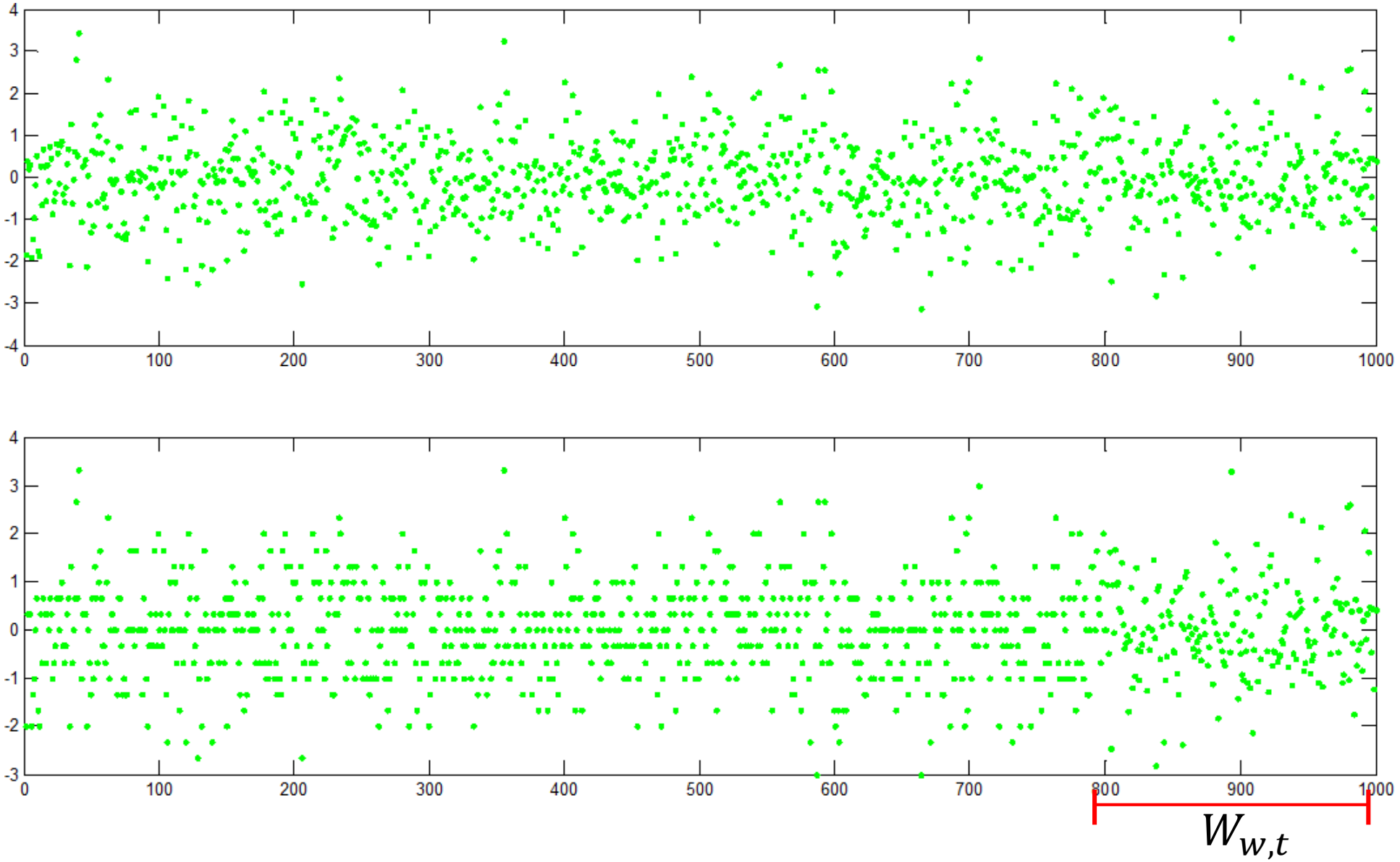
Data Quantization



Quantize past data in m values (range is defined from a training sequence)

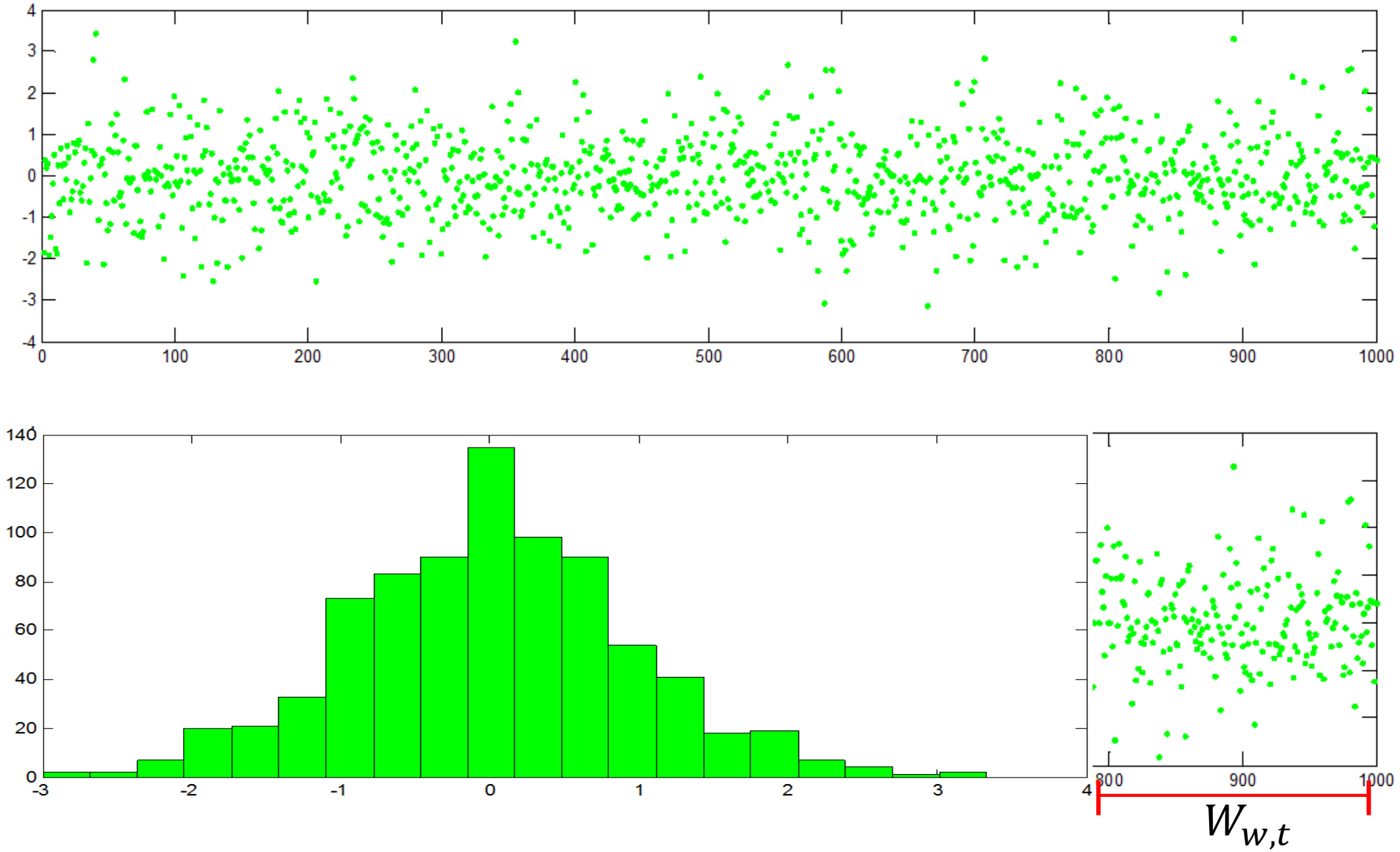
Sliding window $W_{w,t}$ over the stream

Data Quantization



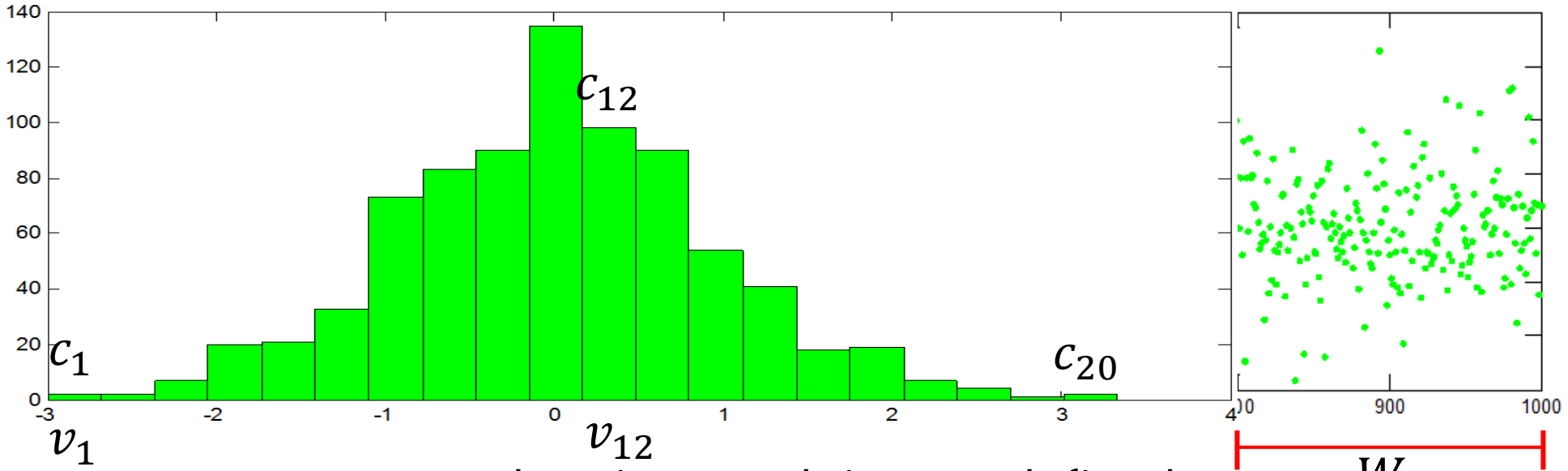
Boracchi, Trovò

Data Quantization



Boracchi, Trovò

Ranks Computation



Each point's rank is now defined as

$$r(x_t) = r_w(x_t) + \sum_{i=1}^{m} c_i \, I\left(x_t > v_j\right) - 1$$

the sum over W plus the rank w.r.t the histogram

Data Quantization

Pros: When windowing is used, the maximum number of operation performed becomes constant (when $t > w$)

Cons: loss of accuracy in rank computation (std adjustement)

Cons: No post-detection diagnosis possible when τ falls out of $W_{w,t}$

The change point outcomes is

$$\tau = \underset{t \in W_{w,t}}{\operatorname{argmax}} S_t$$

Change Detection Approaches

- The Change-Point Formulation
  - Parametric
  - Non-parametric
- Change-Detection by Monitoring Features / the Log-likelihood
- Change-Detection by Histograms

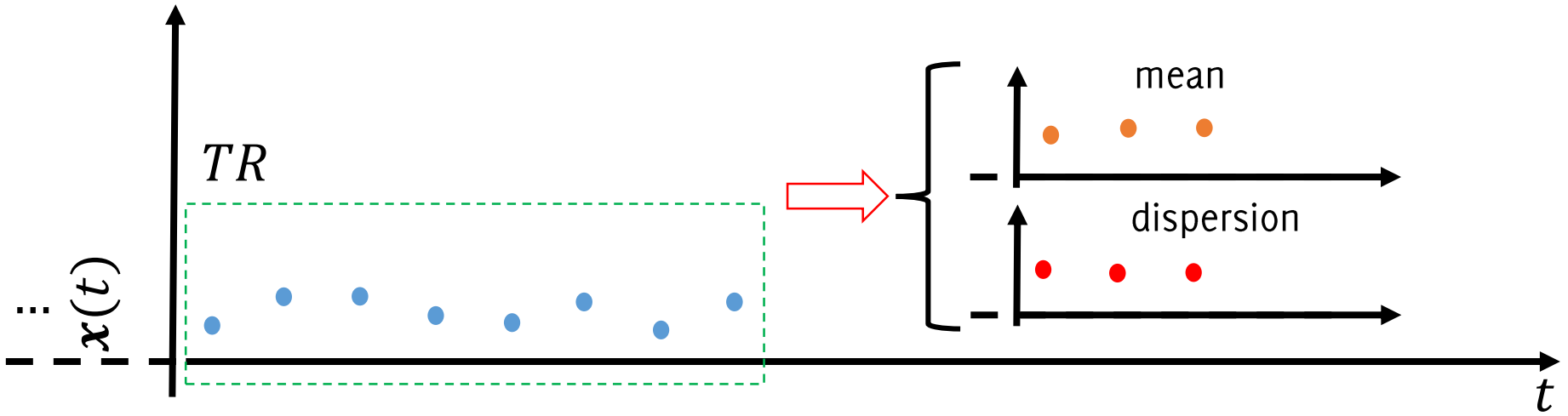
CMPs are nice, but statistics based on sorting holds for scalar sterams

Now we investigate solutions meant for multivariate data streams

Change Detection by Monitoring Features

Most often, **a training set** TR containing **stationary data** is provided, as in semi-supervised anomaly detection methods.

Extract indicators (features), which are **expected to change** when $\phi_0 \rightarrow \phi_1$ and which **distribution is known** under ϕ_0



Alippi, C., Boracchi, G., Roveri, M. *"Change detection tests using the ICI rule"* IJCNN 2010 (pp. 1-7).

Nonparametric settings: Sequential Monitoring

Examples of **decision rules** for features

- **CPM,** which can control the ARL_0
- **NP-CUSUM,** to detect changes in the data expectation
- **ICI rule**, to detect changes in the data expectation

Unfortunately **most** nonparametric statistics and the decision rules **do not apply to multivariate data.**

Different features are being monitored separately

Ross, G. J., Tasoulis, D. K., Adams, N. M. *"Nonparametric monitoring of data streams for changes in location and scale"* Technometrics, 53(4), 379-389, 2012.

Alippi, C., Boracchi, G., Roveri, M. *"Change detection tests using the ICI rule"* Proceedings of IJCNN 2010 (pp. 1-7).

Tartakovsky, A. G., Veeravalli, V. V. "*Change-point detection in multichannel and distributed systems*". Applied Sequential Methodologies: Real-World Examples with Data Analysis, 173, 339-370, 2004

Alippi C., Boracchi G. and Roveri M. *"Ensembles of Change-Point Methods to Estimate the Change Point in Residual Sequences"* Soft Computing, Springer, Volume 17, Issue 11 (2013)

Monitoring the Log-Likelihood: A Mainstream Change Detection Approach

Three ingredients

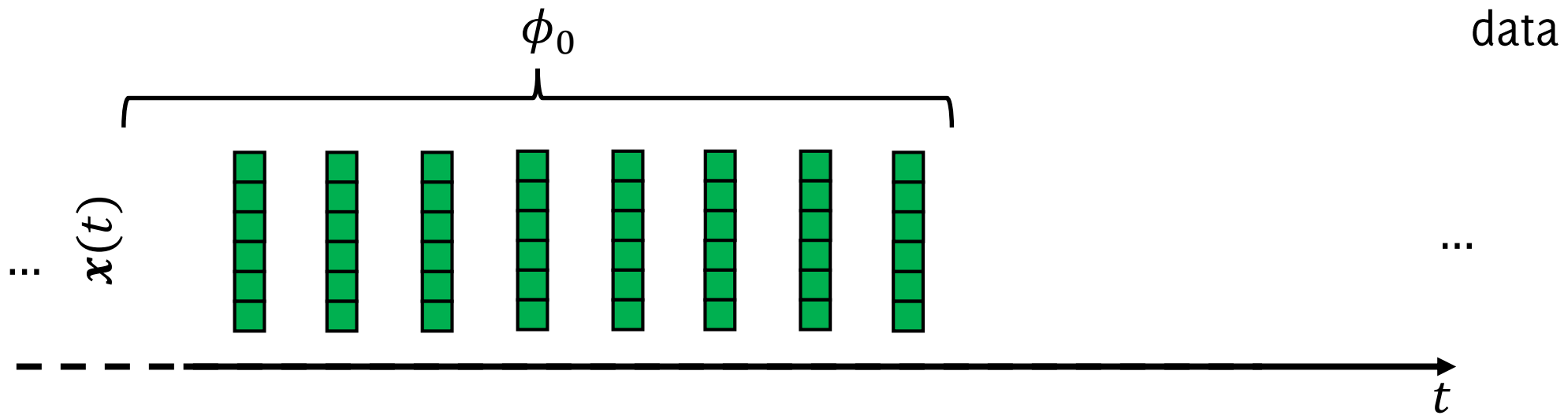
Most change-detection algorithm consists in

i. A model $\hat{\phi}_0$ describing ϕ_0

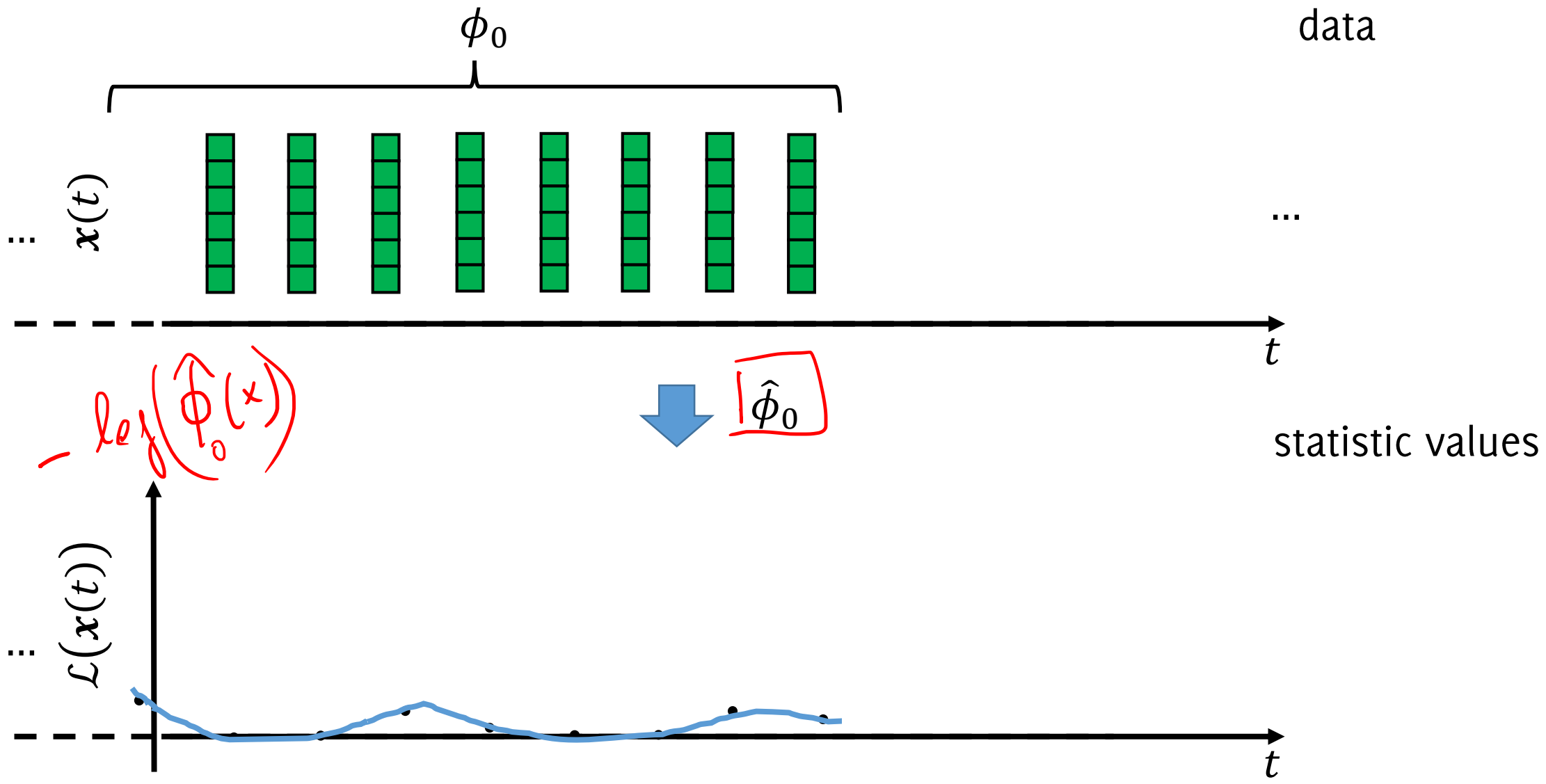
ii. A statistic $\mathcal{T}$ to test incoming data:

iii. A decision rule that monitors $\mathcal{T}$ to detect changes

Illustration

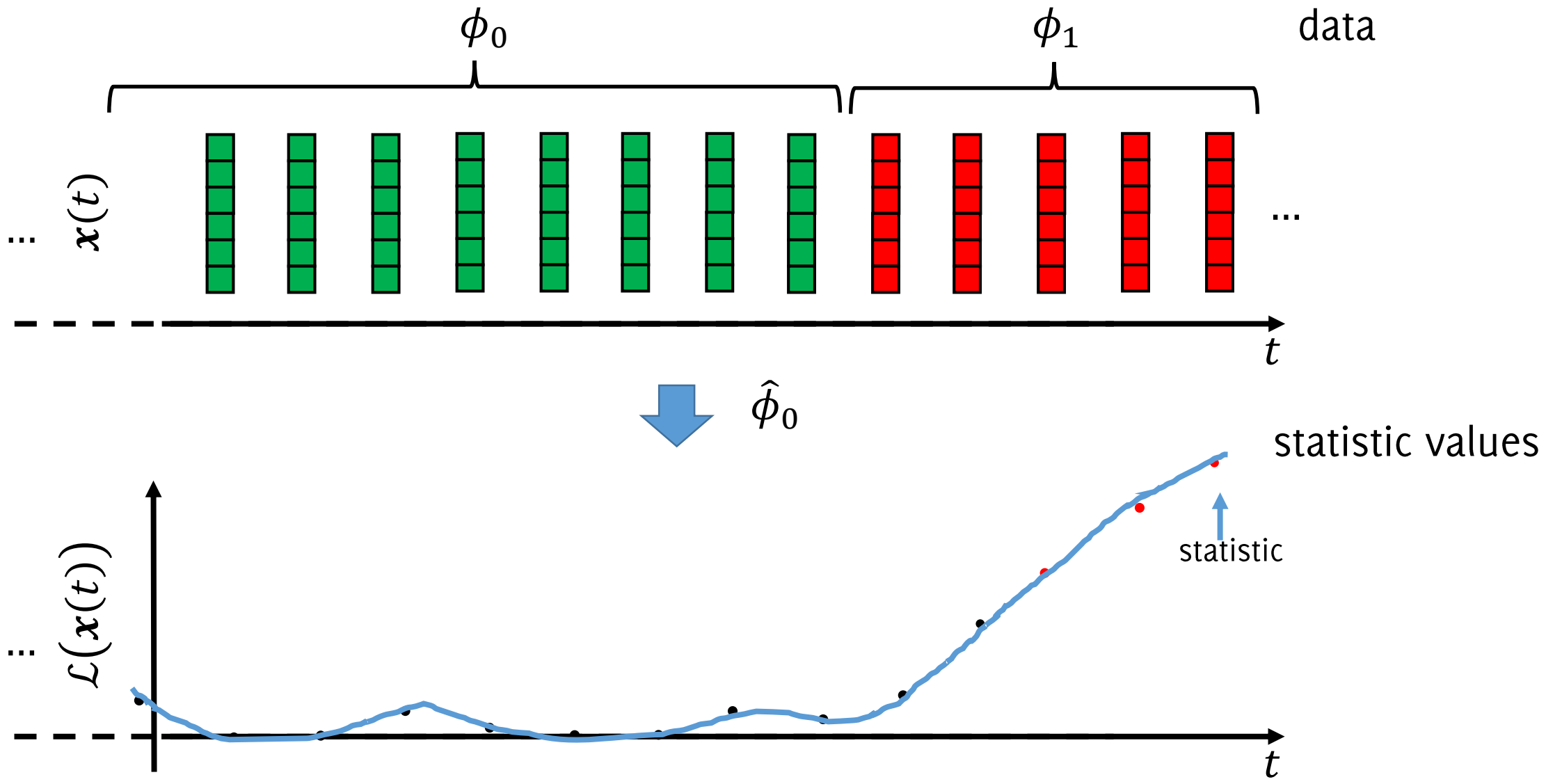


Illustration



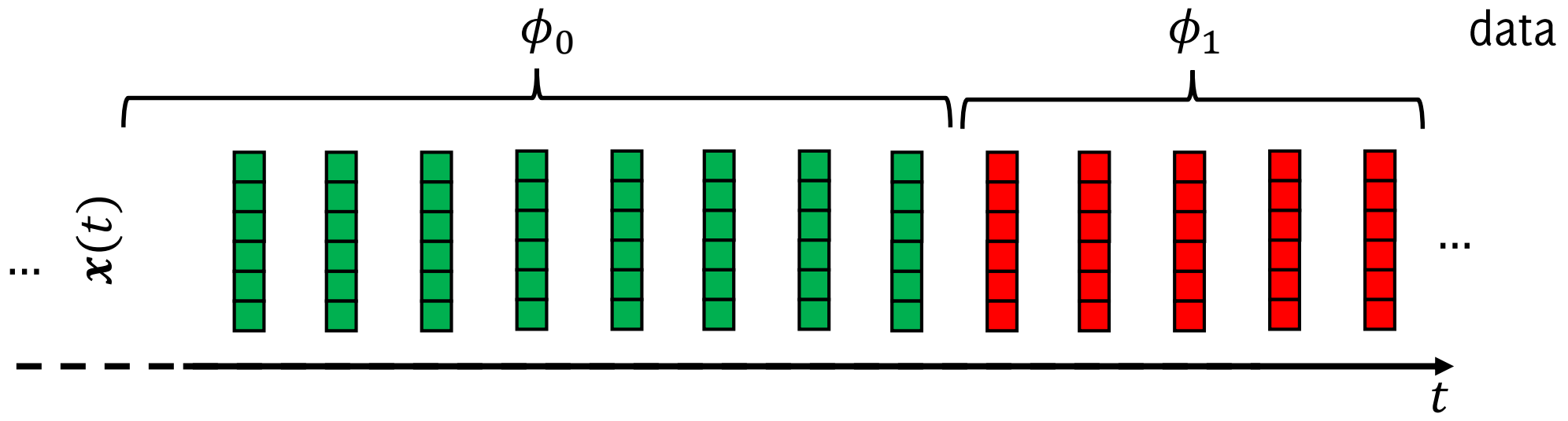
Boracchi, Trovò

Illustration



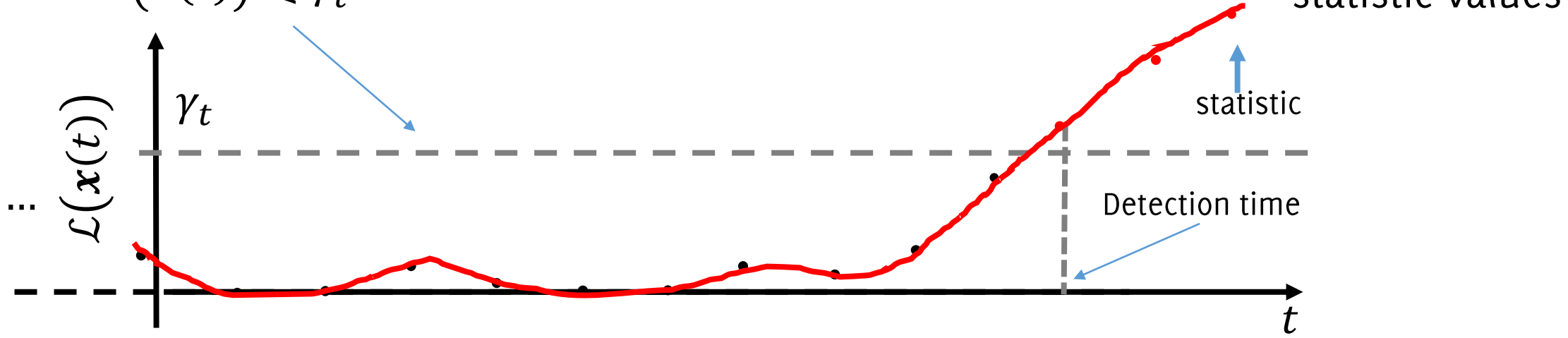
Boracchi, Trovò

Illustration



$\hat{\phi}_0$

decision rule $\mathcal{L}(\boldsymbol{x}(t)) \gtrless \gamma_t$



Boracchi, Trovò

Monitoring the log-likelihood

Fit a general density model $\hat{\phi}_0$ from TR

- Gaussian Mixtures
- Nonparametric Models (KDE)

Statistic to monitor:

$$\mathcal{L}\big(\boldsymbol{x}(t)\big) = -\log(\hat{\phi}_0(\boldsymbol{x}(t)))$$

Heuristic decision rule

$$\mathcal{L}\big(\boldsymbol{x}(t)\big) > \gamma$$

L. I. Kuncheva, *"Change detection in streaming multivariate data using likelihood detectors,"* IEEE Transactions on Knowledge and Data Engineering, 2013.

X. Song, M. Wu, C. Jermaine, and S. Ranka, *"Statistical change detection for multidimensional data,"* in Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), 2007.

J. H. Sullivan and W. H. Woodall, *"Change-point detection of mean vector or covariance matrix shifts using multivariate individual observations,"* IIE transactions, vol. 32, no. 6, 2000.

Monitoring the log-likelihood

Fit a general density model $\hat{\phi}_0$ from TR

- Gaussian Mixtures
- Nonparametric Models (KDE)

Statistic to monitor:

$$\mathcal{L}\big(\boldsymbol{x}(t)\big) = -\log(\hat{\phi}_0(\boldsymbol{x}(t)))$$

Heuristic decision rule

$$\mathcal{L}\big(\boldsymbol{x}(t)\big) > \gamma$$

Computing the log prevents numerical errors in case of Gaussian densities. For Gaussian mixtures this can be approximated

L. I. Kuncheva, *"Change detection in streaming multivariate data using likelihood detectors,"* IEEE Transactions on Knowledge and Data Engineering, 2013.

X. Song, M. Wu, C. Jermaine, and S. Ranka, *"Statistical change detection for multidimensional data,"* in Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), 2007.

J. H. Sullivan and W. H. Woodall, *"Change-point detection of mean vector or covariance matrix shifts using multivariate individual observations,"* IIE transactions, vol. 32, no. 6, 2000.

Sequential Monitoring the log-likelihood

Truly sequential monitoring:

1. Fit a general density model $\hat{\phi}_0$ from TR

$$\hat{\phi}_0 = \text{fit\_density\_model}(\{\boldsymbol{x}(t), t = 1, \dots, N\})$$

2. For each test sample $\boldsymbol{x}(t)$ compute the log-likeihood
3. Adopt a nonparametric CPM over the stream of likelihood values

$$L = \{-\log(\hat{\phi}_0(\boldsymbol{x}(t))), t = 1, \dots, \}$$

Batch-wise anomaly-detection in the log-likelihood

1. **Fit a general density model** $\hat{\phi}_0$ from TR and **compute the log likelihood** from the last portion of R **training samples** (which have not been used to fit the density model $\hat{\phi}_0$):

$$\hat{\phi}_0 = \text{fit\_density\_model}(\{\boldsymbol{x}(t), t = 1, \dots, N - R\})$$
$$TR_1 = \left\{-\log\left(\hat{\phi}_0\left(\boldsymbol{x}(t)\right)\right), t = N - R + 1, \dots, N\right\}$$

2. Divide the incoming stream in batches and **compute the likelihood over each batch** W_t

$$TS = \left\{-\log\left(\hat{\phi}_0\left(\boldsymbol{x}(t)\right)\right), t \in W_t\right\}$$

3. Detect anomalies as a **left-tailed two-sample t-test** comparing the distributions of likelihood values over TR_1 and TS

C. Alippi, G. Boracchi, D. Carrera, M. Roveri, "*Change Detection in Multivariate Datastreams: Likelihood and Detectability Loss*" IJCAI 2016,

CUSUM control chart (parametric case)

Fit a general density model $\hat{\phi}_0$ from TR

- Gaussian Mixtures
- Nonparametric Models (KDE)

- Statistic to monitor:

$$\mathcal{S}(t) = \left(\log\left(\frac{\hat{\phi}_1\big(\boldsymbol{x}(t)\big)}{\hat{\phi}_0\big(\boldsymbol{x}(t)\big)} \right) + \mathcal{T}(t-1) \right)^+$$

- Decision rule

$$\mathcal{S}(t) > \gamma$$

Histograms in Change Detection

Histograms

An histogram h^0 defined over the input domain $\mathcal{X} \subset \mathbb{R}^d$ is

$$h^0(\mathcal{X}) = \{(S_k, p_k^0)\}_{k=1,\dots,K}$$

Where $\{S_k\}_k$ is a disjoint covering of $\mathcal{X}$, namely $S_k \subset \mathcal{X}$

$$\bigcup_k S_k = \mathcal{X} \text{ and } S_j \cap S_i = \delta_{i,j}$$

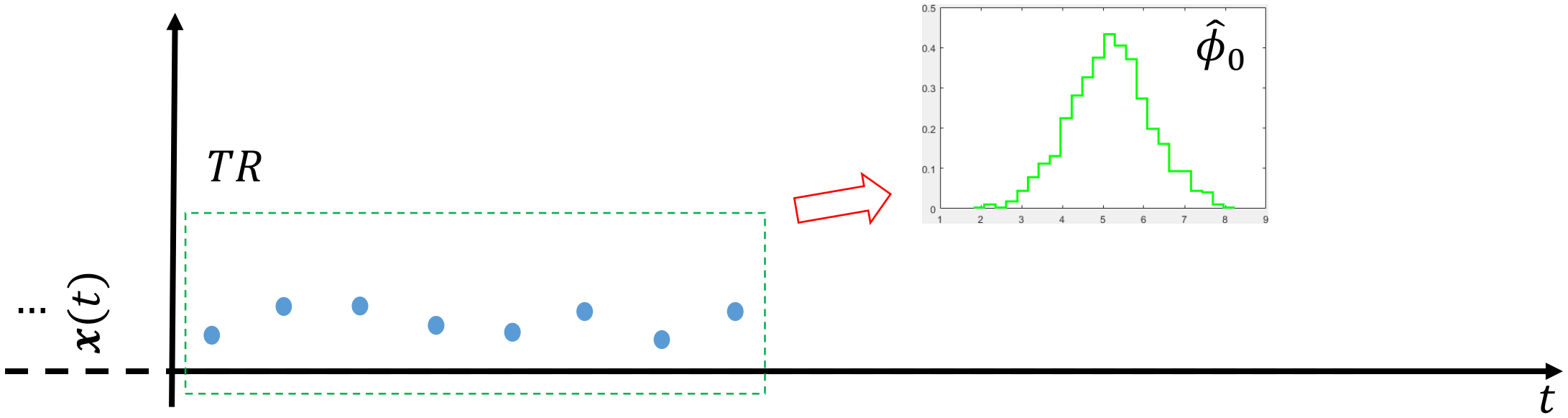
and $p_k^0 \in [0,1]$ is the probability (estimated from TR) for a sample drawn from ϕ_0 to fall inside S_k, i.e.

$$p_k^0 = \frac{m_k}{N}$$

and $N = \#TR$

Change Detection by Means of Histograms

The distribution of stationary data can be approximated by a histogram $\hat{\phi}_0$ estimated from a given training set TR containing stationary data



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

R. Sebastião, J. Gama, P. P. Rodrigues, and J. Bernardes, *"Monitoring incremental histogram distribution for change detection in data streams,"* Lecture Notes on Computer in Knowledge Discovery from Sensor Data, 2017.

Monitoring Approaches

Two **major monitoring approaches** using histograms:

- **Likelihood-based** methods
- **Distance-based** methods

whose applicability also depends on the partitioning scheme

Log-likelihood – Based Monitoring Scheme

As in density-based methods, $\hat{\phi}_0$ can be used to compute the log-likelihood, which can be then monitored by univariate CDT

1. During training, estimate $\hat{\phi}_0 = \{(S_k, p_k^0)\}_{k=1,\dots,K}$ from TR
2. During testing, compute
$$\mathcal{L}(\boldsymbol{x}(t)) = \hat{\phi}_0(\boldsymbol{x}(t))$$
3. Monitor $\{\mathcal{L}(\boldsymbol{x}(t)),\ t = 1, \dots\}$ which is now discrete

Log-likelihood – Based Monitoring Scheme

As in density-based methods, $\hat{\phi}_0$ can be used to compute the log-likelihood, which can be then monitored by univariate CDT

1. During training, estimate $\hat{\phi}_0 = \{(S_k, p_k^0)\}_{k=1,\dots,K}$ from TR
2. During testing, compute

$$\mathcal{L}(\boldsymbol{x}(t)) = \hat{\phi}_0(\boldsymbol{x}(t)) = p_k^0 \text{ s.t. } \boldsymbol{x}(t) \in S_k$$

3. Monitor $\{\mathcal{L}(\boldsymbol{x}(t)),\ t = 1, \dots\}$ which is now discrete

This is the problem of associating each incoming sample to the corresponding bin

Log-likelihood – Based Monitoring Scheme

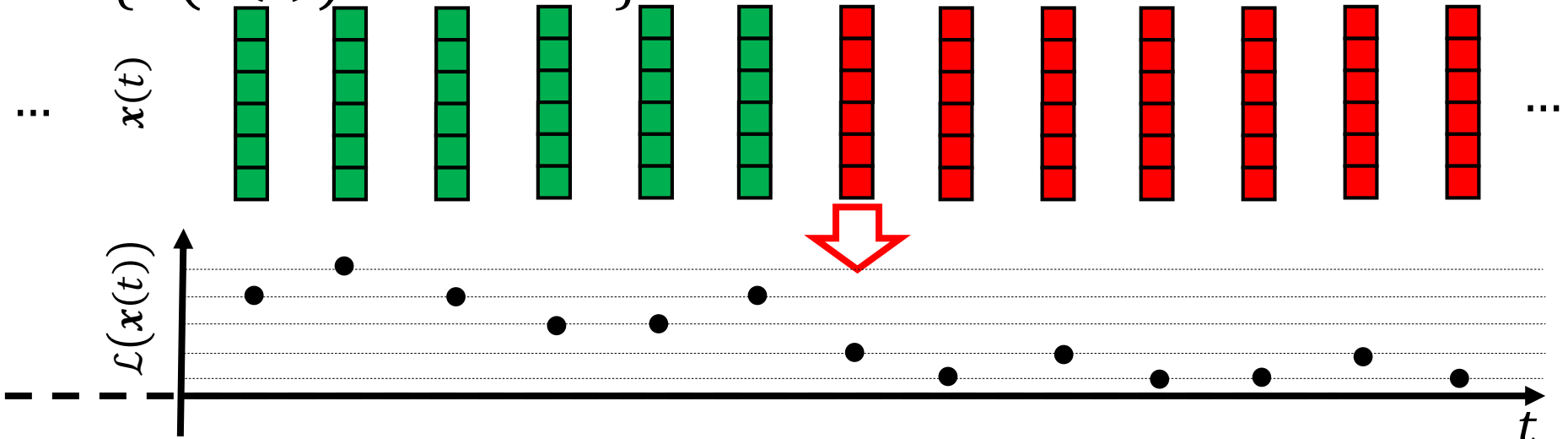
As in density-based methods, $\hat{\phi}_0$ can be used to compute the log-likelihood, which can be then monitored by univariate CDT

1. During training, estimate $\hat{\phi}_0 = \{(S_k, p_k^0)\}_{k=1,\dots,K}$ from TR

2. During testing, compute

$$\mathcal{L}(\boldsymbol{x}(t)) = \hat{\phi}_0(\boldsymbol{x}(t)) = p_k^0 \text{ s.t. } \boldsymbol{x}(t) \in S_k$$

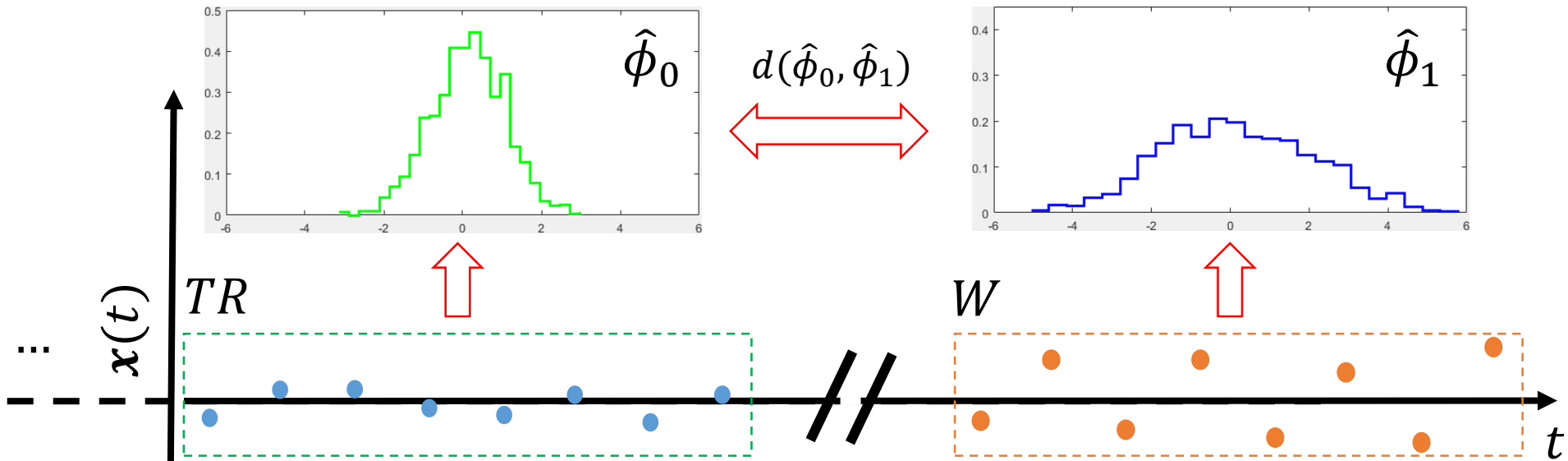
3. Monitor $\{\mathcal{L}(\boldsymbol{x}(t)),\ t = 1, \dots\}$ which is now discrete



Distance-Based (or batch) Monitoring Scheme

$\hat{\phi}_0$ can be used to monitor the datastream window-wise:

- During training, estimate $\hat{\phi}_0 = \{(S_k, p_k^0)\}_{k=1,\dots,K}$ from TR
- Crop a window W over the most recent data
- Estimate $\hat{\phi}_1 = \{(S_k, p_k^1)\}_{k=1,\dots,K}$ from W
- Compare $\hat{\phi}_0$ and $\hat{\phi}_1$ by a distance d between distributions
- Monitor $d(\hat{\phi}_0, \hat{\phi}_1)$



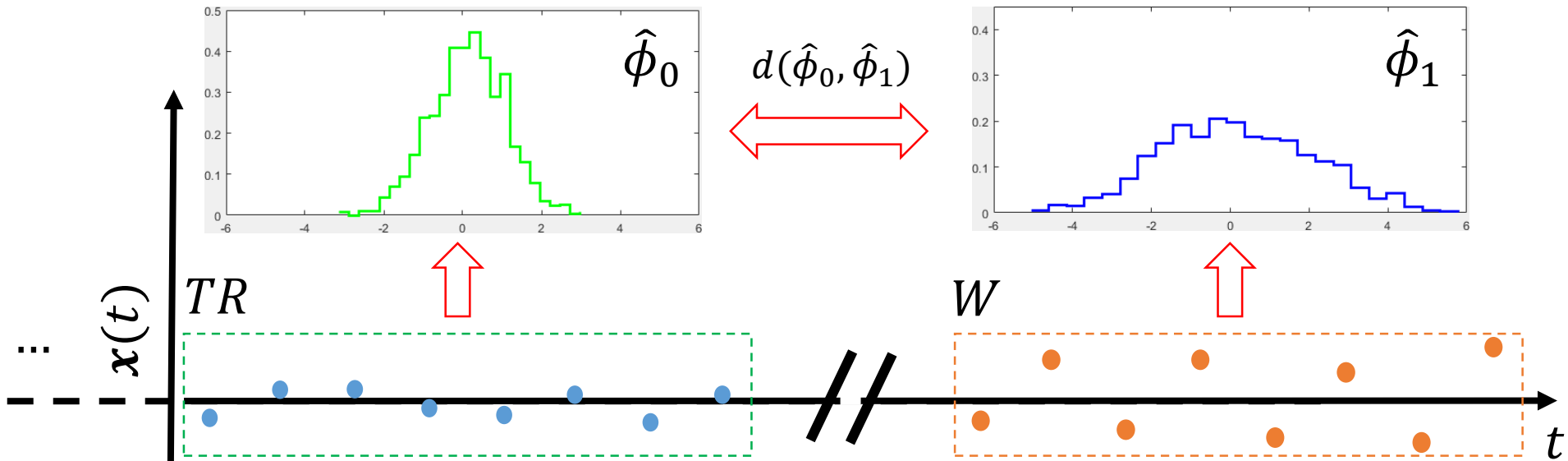
Boracchi, Trovò

Distance-Based (or batch) Monitoring Scheme

$\hat{\phi}_0$ can be used to monitor the datastream window-wise:

- During training, estimate $\hat{\phi}_0 = \{(S_k, p_k^0)\}_{k=1,\dots,K}$ from TR
- Crop a window W over the most recent data
- Estimate $\hat{\phi}_1 = \{(S_k, p_k^1)\}_{k=1,\dots,K}$ from W
- Compare $\hat{\phi}_0$ and $\hat{\phi}_1$ by a distance d between distributions
- Monitor $d(\hat{\phi}_0, \hat{\phi}_1)$

Here bins are defined by $\hat{\phi}_0$, we just have to associate each sample to the corresponding bin



Boracchi, Trovò

Distance-Based Monitoring scheme: Stopping Rule

Thresholding the distance is the typical stopping rule.

$$d(\hat{\phi}_0, \hat{\phi}_1) \gtrless \gamma$$

- γ defined from the empirical distribution of $d(\hat{\phi}_0, \hat{\phi}_1)$, which is computed through a **Bootstrap procedure**.
- γ given from **approximation of the statistic**, which typically **holds asymptotically**, as in case the of Pearson

Similar approaches can be used to compare features extracted in different data-windows.

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

Ditzler G., Polikar R., *"Hellinger distance based drift detection for nonstationary environments"*, IEEE SSCI 2011.

Boracchi G., Cervellera C., and Maccio D. *"Uniform Histograms for Change Detection in Multivariate Data"* IJCNN 2017

Sebastião R., Gama J. Mendonça T. *"Fading histograms in detecting distribution and concept changes"* IJDSA, 2017

Bu L., Alippi C., Zhao D. *"A pdf-free change detection test based on density difference estimation"* TNNLS 2016

S. Liu, M. Yamada, N. Collier, and M. Sugiyama, *"Change-point detection in time-series data by relative density-ratio estimation,"* Neural Networks, 2013

An example of distance-based monitoring scheme

1. Compute the probabilities for an incoming batch W over $\{S_k\}$

$$p_k^W = \frac{\#\{\boldsymbol{x}_i \in S_k \cap W\}}{\nu}$$

2. Compare h^0 and h^W by a suitable distance, e.g.

$$d_{TV}(h^0, h^W) = \frac{1}{2}\sum_k |p_k^0 - p_k^W| \quad \text{(total variation)}$$

or

$$d_{PS}(h^0, h^W) = \nu \sum_k \frac{\left(p_k^0 - p_k^W\right)^2}{p_k^0} \quad \text{(Pearson)}$$

3. Run an HT on d_{TV} (having estimated its p-values empirically) or d_P (this follows a χ-square distribution)

Pros and Cons of using histograms

Pros:

- Histograms are very **general and flexible models**.
- Some partitioning schemes can be associated with **a tree having splits along a single component (kd-trees, quantTrees)**. This enable very fast searches through the histogram.

Cons:

- When d increases, grids are not a viable option, since they require q^d bins.
- In general, the distribution of test statistic is unknown

Pros and Cons of using histograms

Pros:

- Histograms are very **general and flexible models**.
- Some partitioning schemes can be associated with **a tree having splits along a single component (kd-trees, quantTrees)**. This enable very fast searches through the histogram.

Cons:

- When d increases, grids are not a viable option, since they require q^d bins.
- In general, the distribution of test statistic is unknown

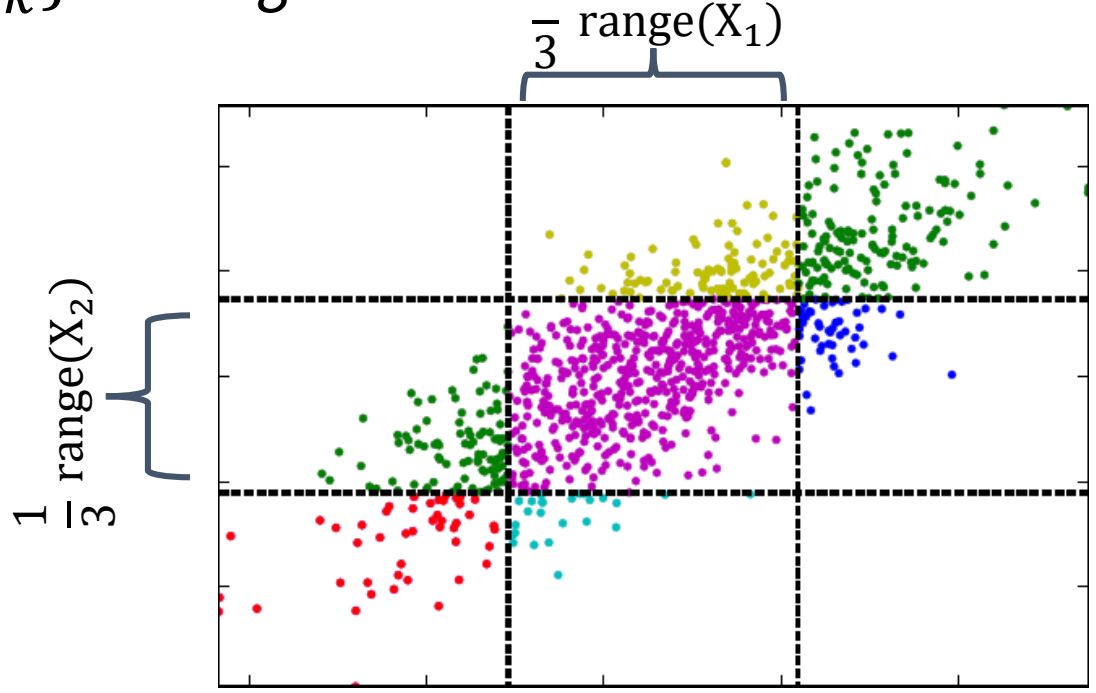
However, there is quite a lot of freedom in designing $\{S_k\}_k$

Histograms yielding uniform volume

"grids": the most common way of constructing histograms.

Build a tessellation of $\mathrm{supp}(TR)$ by splitting each component in q equally sized parts.

This yields q^d hyper-rectangles $\{S_k\}$ having the **same volume**



An example of 2D histogram $q = 1/3$

Histograms yielding uniform volume

"grids": the most common way of constructing histograms.

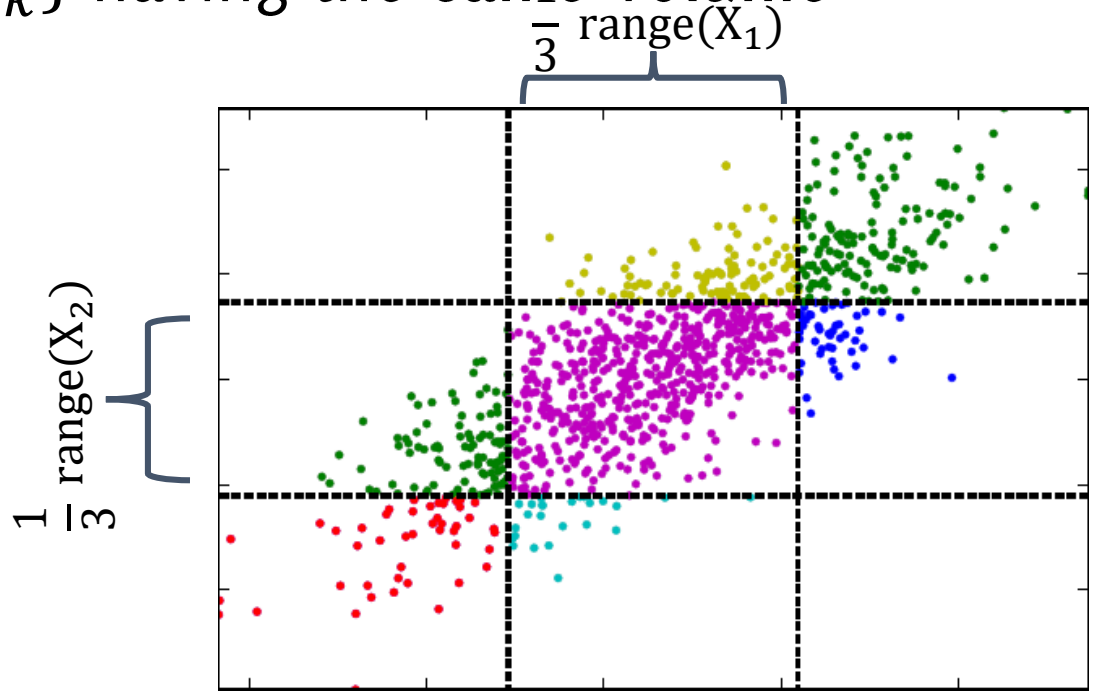
Build a tessellation of $\mathrm{supp}(TR)$ by splitting each component in q equally sized parts.

This yields q^d hyper-rectangles $\{S_k\}$ having the **same volume**

Add to the histogram a region to gather points that during operation, won't fall in $\mathrm{supp}(TR)$

$$S_K = \overline{TR}, p_K^0 = 0$$

being $K = q^d + 1$



An example of 2D histogram $q = 1/3$

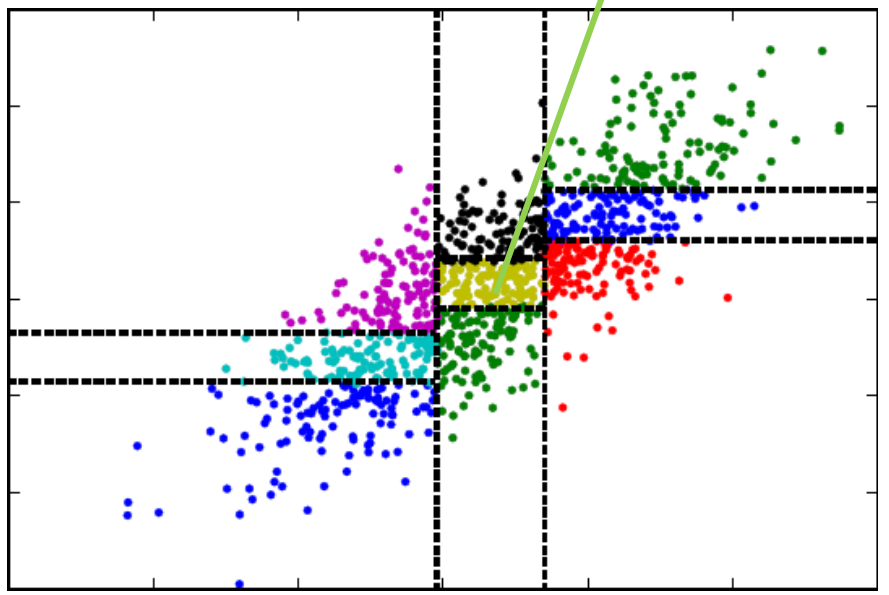
Histograms yielding uniform density

Define the partition $\{S_k\}_k$ in such a way that all the subsets have the **uniform density**, i.e.,

$$p_k^0 \approx \frac{1}{K}, k = 1, .., K$$

Such that each of the q^d hyper-rectangles contains the same number of points

No need to consider a bin for $\bar{X}$



An example of 2D histogram $q = 1/3$

Boracchi G., Cervellera C., and Maccio D. *"Uniform Histograms for Change Detection in Multivariate Data"* IJCNN 2017

Histograms yielding uniform density

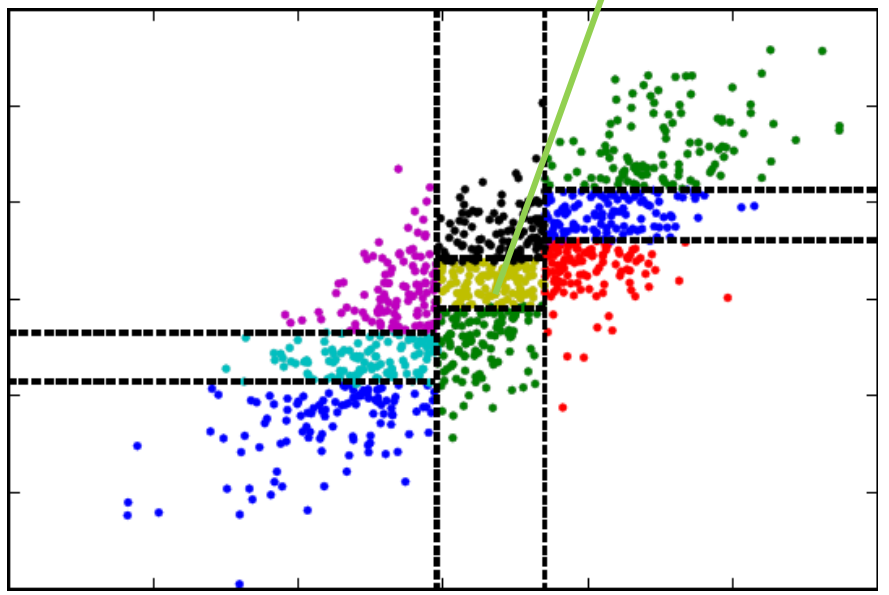
Define the partition $\{S_k\}_k$ in such a way that all the subsets have the **uniform density**, i.e.,

$$p_k^0 \approx \frac{1}{K}, k = 1, .., K$$

Such that each of the q^d hyper-rectangles contains the same number of points

No need to consider a bin for $\bar{X}$

This is an example of k-d tree, there are many alternatives…



An example of 2D histogram $q = 1/3$

Boracchi G., Cervellera C., and Maccio D. *"Uniform Histograms for Change Detection in Multivariate Data"* IJCNN 2017

Performance Measures in Change Detection

How to assess performance of change/anomaly detection algorithms

Boracchi, Trovò

Anomaly-Detection Performance

Anomaly detection performance:

- True positive rate: $TPR = \frac{\#\{\text{anomalies detected}\}}{\#\{\text{anomalies}\}}$
- False positive rate: $FPR = \frac{\#\{\text{normal samples detected}\}}{\#\{\text{normal samples}\}}$

You have probably also heard of

- False negative rate (or miss-rate): $FNR = 1 - TPR$
- True negative rate (or specificity): $TNR = 1 - FPR$
- Precision on anomalies: $\frac{\#\{\text{anomalies detected}\}}{\#\{\text{detections}\}}$
- Recall on anomalies (or sensitivity, hit-rate): TPR

Anomaly-Detection Performance

There is always a **trade-off between $\boldsymbol{TPR}$ and $\boldsymbol{FPR}$** (and similarly for derived quantities), which is ruled by algorithm parameters

Anomaly-Detection Performance

There is always a **trade-off between TPR and FPR** (and similarly for derived quantities), which is ruled by algorithm parameters

Thus, to correctly assess performance it is necessary to consider at least **two indicators** (e.g., TPR, FPR)

Indicators combining both TPR and FPR:

$$\text{Accuracy} = \frac{\#\{\text{anomalies detected}\} + \#\{\text{normal samples not detected}\}}{\#\{\text{samples}\}}$$

$$\text{F1 score} = \frac{2\#\{\text{anomalies detected}\}}{\#\{\text{detections}\} + \#\{\text{anomalies}\}}$$

These equal 1 in case of "ideal detector" which detects all the anomalies and has no false positives

Anomaly-Detection Performance

Comparing different methods might be tricky since we have to make sure that both have been configured in their best conditions

Testing a large number of parameters lead to the **ROC** (receiver operating characteristic) **curve**

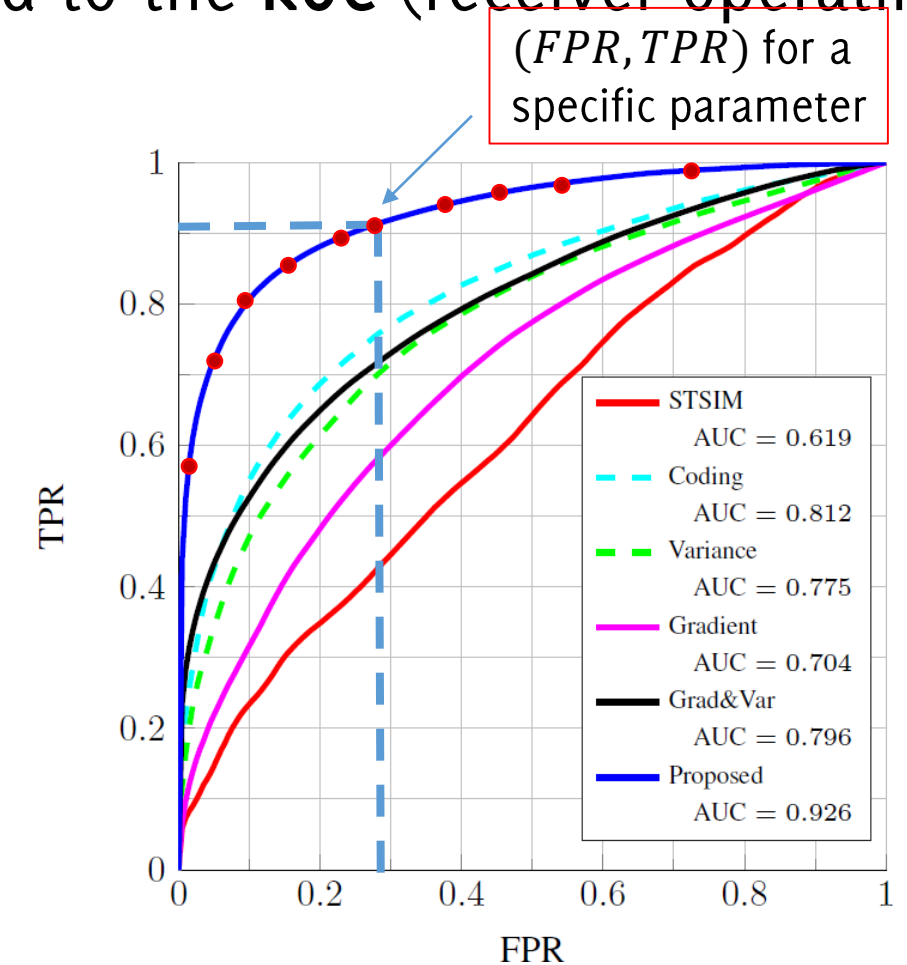
The ideal detector would achieve:

- $FPR = 0\%$,
- $TPR = 100\%$

Thus, the closer to (0,1) the better

The largest the **Area Under the Curve** (AUC), the better

The optimal parameter is the one yielding the point closest to (0,1)



racchi, Trovò

Change-Detection Performance

In a sequential monitoring scenarios, performance are assessed in terms of the Average Run Length.

In particular, we denote by $\hat{T}$ the detection time and define

$$ARL_0 = \mathop{\mathrm{E}}_{x}\left[\hat{T} \mid \phi_0\right]$$

which is the **expected number of samples before a false alarm** and

$$ARL_1 = \mathop{\mathrm{E}}_{x}\left[\hat{T} \mid \phi_1\right]$$

which is the **expected delay for a detection of the change** $\phi_0 \rightarrow \phi_1$

ARL_0 and ARL_1 still depend on the algorithm parameters.

In particular, one configures the CDT to operate at a given ARL_0

M. Basseville, I. V. Nikiforov, "*Detection of Abrupt Changes - Theory and Application*", Prentice-Hall, Inc. April 1993

Ross, G. J., Tasoulis, D. K., Adams, N. M. *"Nonparametric monitoring of data streams for changes in location and scale"* Technometrics, 53(4), 379-389, 2012.

Change-Detection Performance

Unfortunately, it is not always possible to compute ARL_0 and/or ARL_1, in particular for nonparametric CDTs.

Then, one resorts to **performing several simulations** on finite sequences with a change at a known location τ, and computing

The **detection delay**,

$$DD = \mathop{\mathrm{E}}_{x}\left[\hat{T} - \tau \mid \hat{T} \geq \tau, \phi_1\right]$$

and

- $FPR = \frac{\#\{\text{normal sequences where a change was detected}\}}{\#\{\text{normal sequences}\}}$
- $FNR = \frac{\#\{\text{sequences where change was not detected }\}}{\#\{\text{changed sequences}\}}$

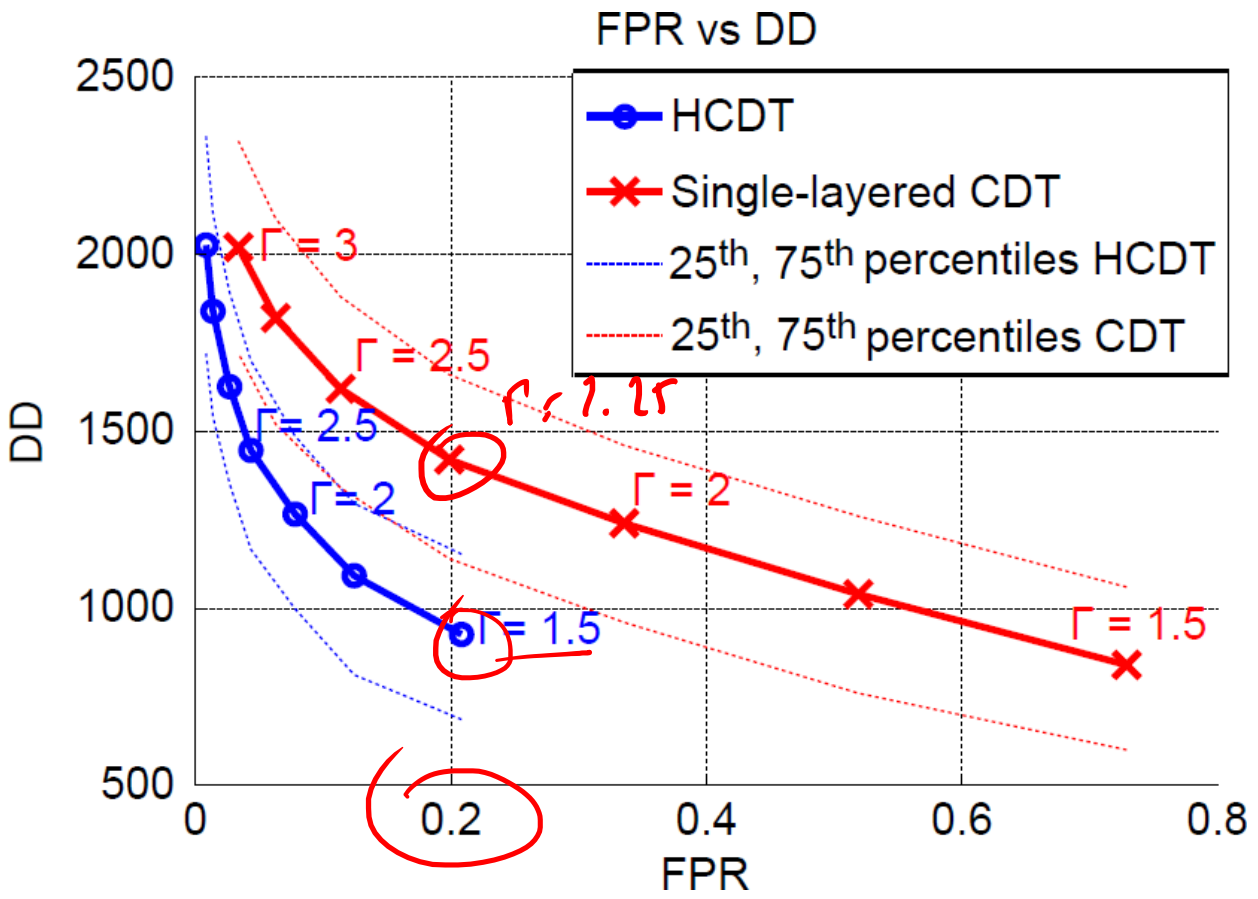
which are defined as in the anomaly detection case, and here depend on the sequence length

Change-Detection Performance

These figures of merit also depend on algorithm parameters.

To perform a fair comparison among different methods one can:

- Generate long enough sequences to have $FNR = 0\%$
- Consider few values of thresholds/parameters for the CDT
- Draw FPR-DD curves (similar to ROC): the lower the better



Third Matlab Assignment

Boracchi, Trovò

Part 1: get the third matlab snippet

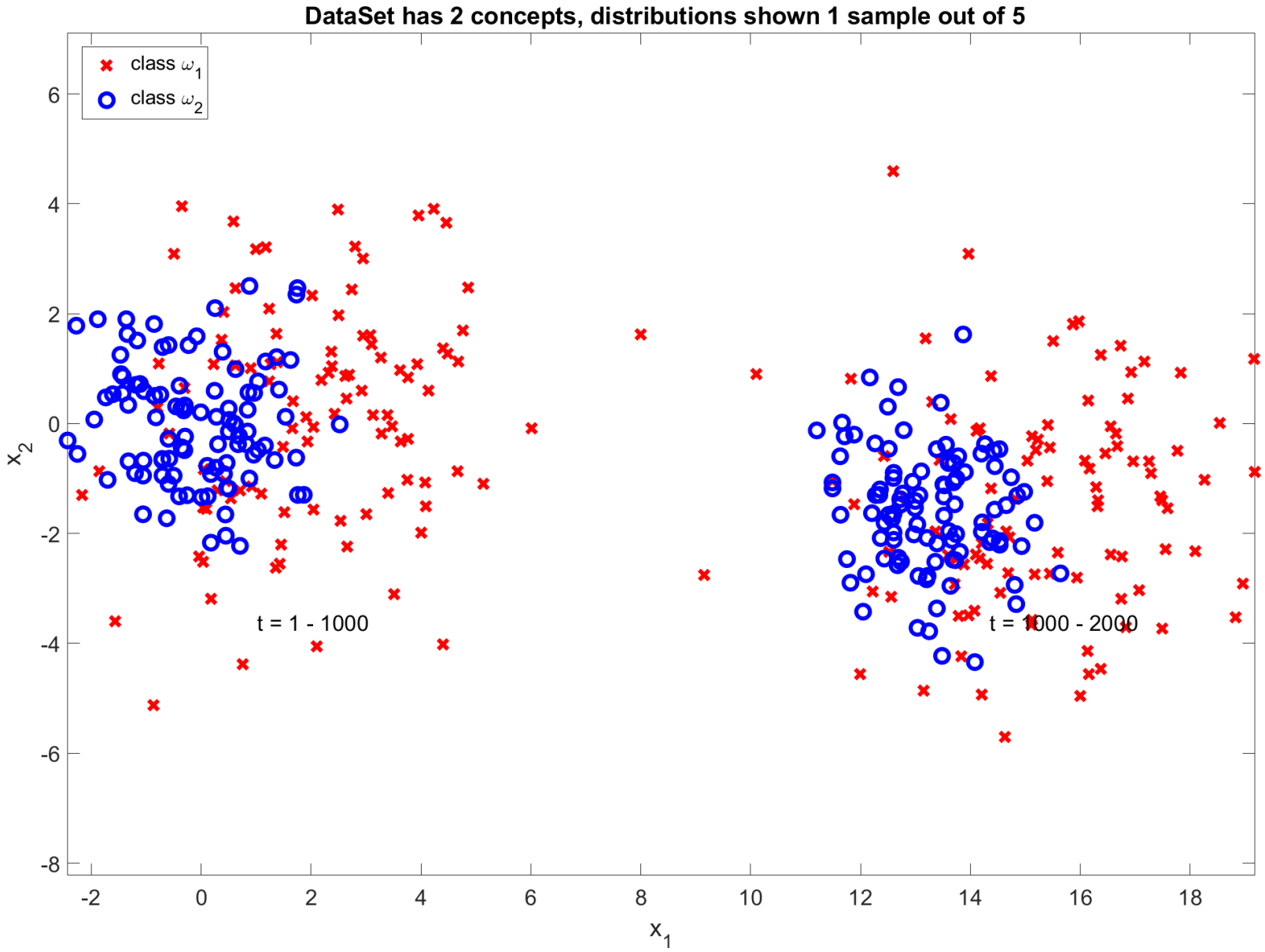
And develop a monitoring scheme that

- **Estimates a Gaussian density model** $\hat{\phi}_0$ from TR_0, a portion of $TR = TR_0 \cup TR_1$ and uses TR_1 to compute the likelihood values from the inital concept
- As soon as a new bacth comes in, **detects distribution changes** by a two-sample t-test, monitoring only for an increase in $-\log(\hat{\phi}_0(\boldsymbol{x}(t)))$
- **Updates the classifier accordingly**, by setting after each detection, the starting point for the new concept the left end of the latest batch

Compare the performance with a classifier that is never updated

Display the likelihood values before and after the change.

DataSet (change in covariates)

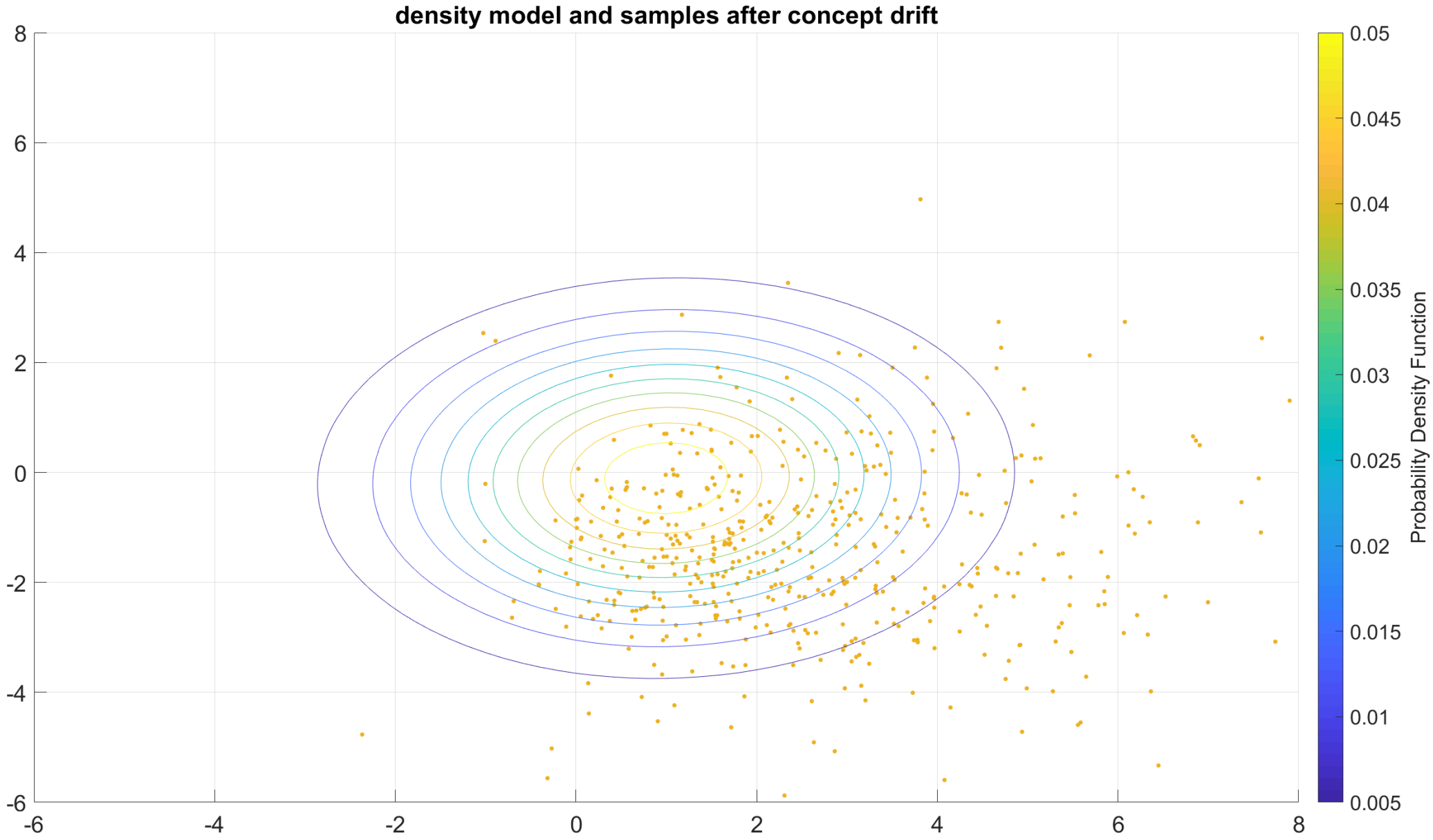


Boracchi, Trovò

Training Samples and $\hat{\phi}_0$

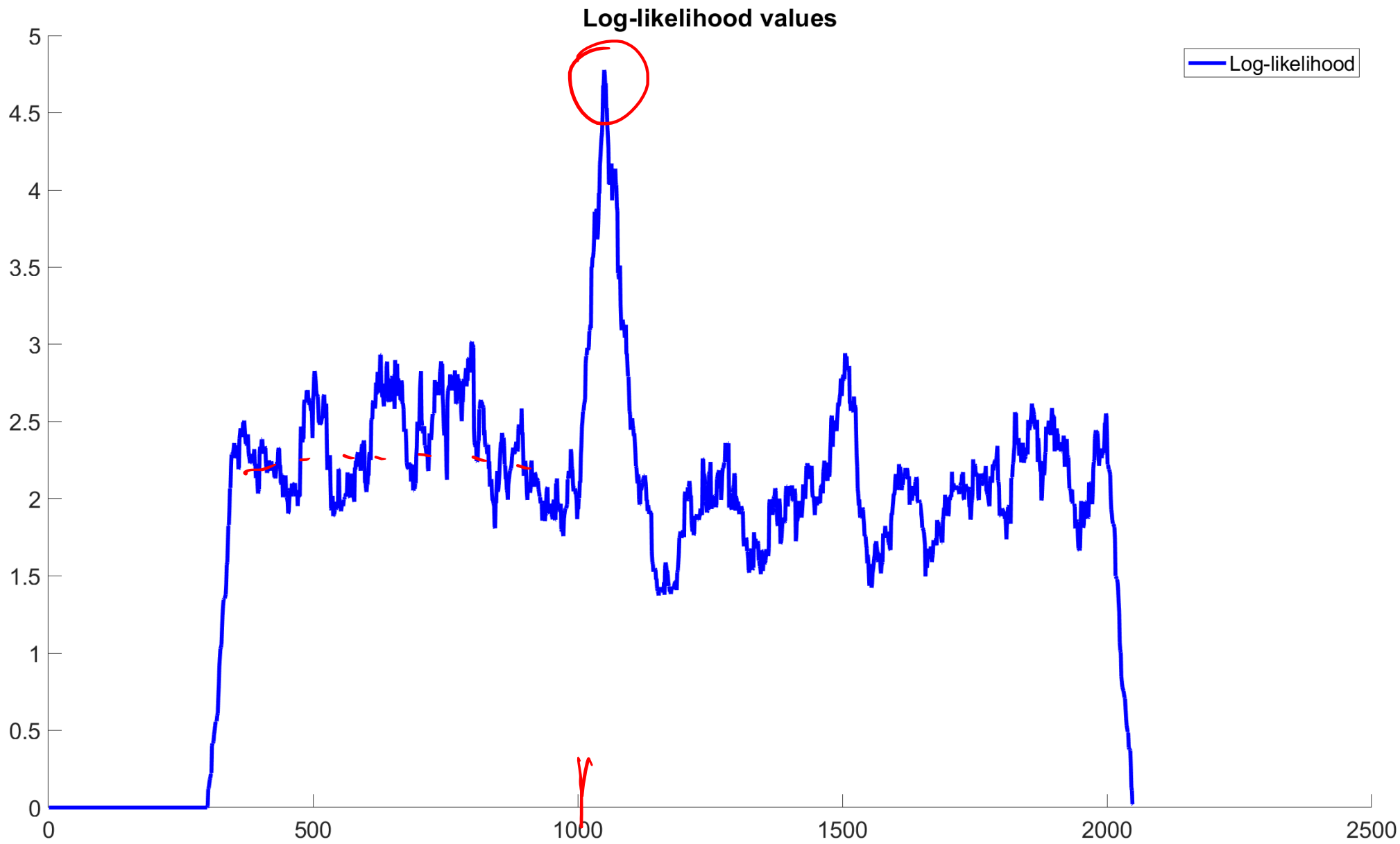
Boracchi, Trovò

$\hat{\phi}_0$ under concept drift



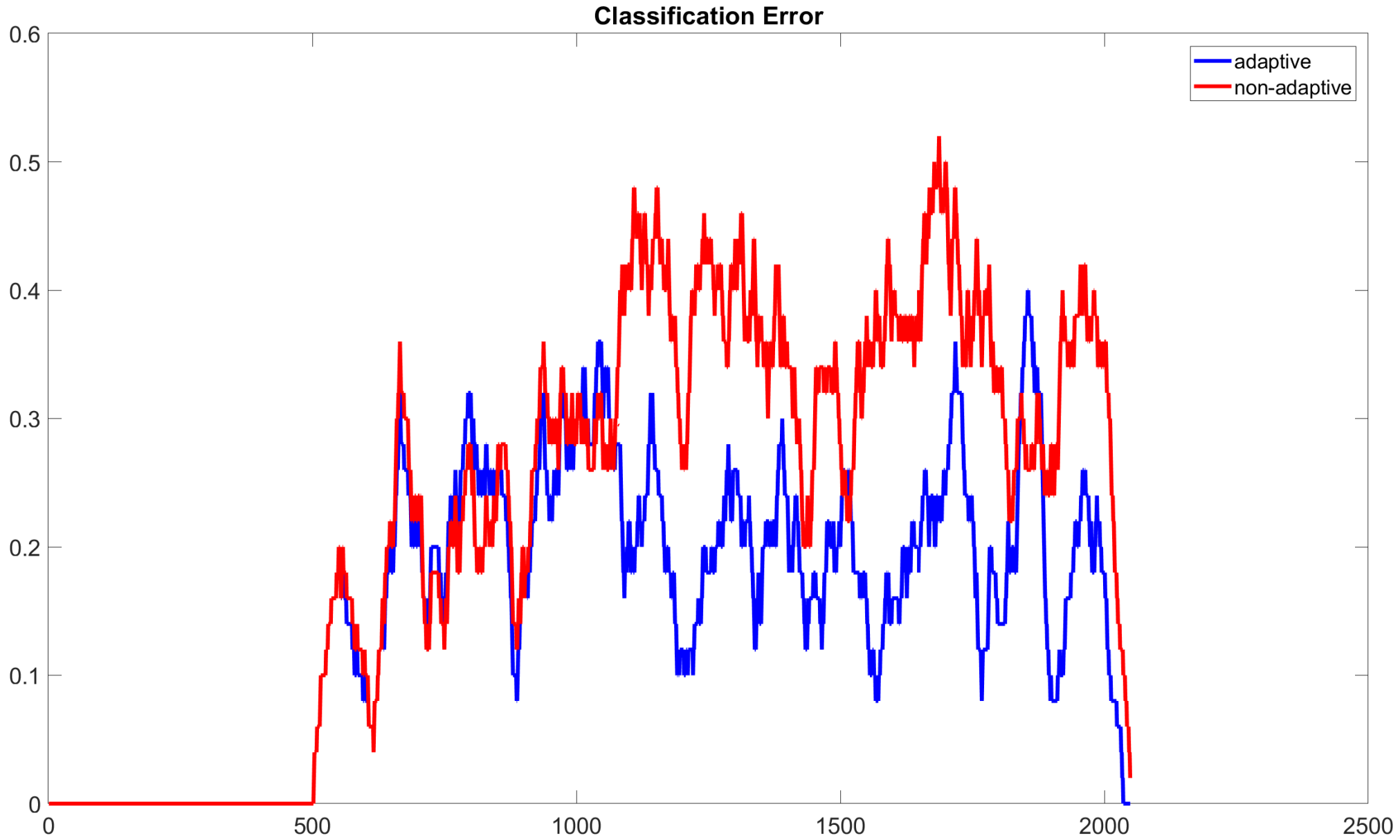
Boracchi, Trovò

Let's have a look at the log-likelihood



Boracchi, Trovò

And the resulting classification error



Boracchi, Trovò

Part2: once done Part1,

Modify the script to:

- Try different setup for the data-generating process. In particular, change the concept drift type, possibly modifying the function **`defineExperimentParameters,`** as well as the frequency of supervised samples **`m.`**
- Implement an adaptive classifier that leverages both error- and input-monitoring schemes to detect concept drift and adapts accordingly
- Try your classifier on longer sequences with multiple changes